Information and Computation **145**, 153–190 (1998)
Article No. IC982724

Uniform Closures: Order-Theoretically Reconstructing Logic Program Semantics and Abstract Domain Refinements

Roberto Giacobazzi

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy
E-mail: giaco@di.unipi.it

and

Francesco Ranzato

Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova, Italy
E-mail: franz@math.unipd.it

The notion of uniform closure operator is introduced, and it is shown how this concept surfaces in two different areas of application of abstract interpretation, notably in semantics design for logic programs and in the theory of abstract domain refinements. In logic programming, uniform closures permit generalization, from an order-theoretic perspective, of the standard hierarchy of declarative semantics. In particular, we show how to reconstruct the model-theoretic characterization of the well-known s-semantics using pure order-theoretic concepts only. As far as the systematic refinement operators on abstract domains are concerned, we show that uniform closures capture precisely the property of a refinement of being invertible, namely of admitting a related operator that simplifies as much as possible a given abstract domain of input for that refinement. Exploiting the same argument used to reconstruct the s-semantics of logic programming, we yield a precise relationship between refinements and their inverse operators: we demonstrate that they form an adjunction with respect to a conveniently modified complete order among abstract domains. © 1998 Academic Press

Key Words: uniform closure, abstract interpretation, logic program semantics, abstract domain refinement.

1. INTRODUCTION

Abstract interpretation [Cousot and Cousot 1977, 1979] is a well-establised theory for program analysis specification, which is gradually gaining ground also as a formal basis for the comparative study and the design of programming language semantics at different levels of abstraction [Cousot 1996, 1997a, 1997b; Cousot and Cousot 1992b, 1995; Giacobazzi 1996; Giacobazzi and Ranzato 1996]. Abstract interpretation theory provides the right mathematical tools to relate in a precise way semantic definitions and to systematically design new semantics, e.g., by approximation or refinement of existing ones. In POPL'92, Patrick and Radhia Cousot present the basis for this new range of applications, by studying the abstract interpretation of generic inductive definitions and by relating trace-based, relational, and denotational semantics.

This work shows that some relevant constructions and results known in semantics of logic programming languages can be generalized from an order-theoretic perspective, and then applied in abstract interpretation theory, providing new and unexpected results in both areas. Our research starts by the attempt to reconstruct the standard hierarchy of declarative semantics for logic programs [Falaschi *et al.* 1989, 1993] in a purely order-theoretic fashion, i.e., independently from the properties peculiar to the objects manipulated by logic programs, namely atoms, clauses and substitutions. Surprisingly, this generalization, which relies on typical abstract interpretation tools like closure operators and Galois connections, while providing the possibility to extend some standard and well-known results in the field of logic programming semantics to other programming languages and semantics, also surfaces in other, completely different, applications of abstract interpretation theory, that is in the systematic design of abstract domains. In a sense, this is a good example where the traditional theories of programming language semantics and abstract interpretation may both benefit from a cross fertilization.

Within the standard Cousot and Cousot [1977, 1979] framework, (upper) closure operators capture the "essence" of the process of abstraction; namely, they play the role of approximating operators. Given a concrete domain C, i.e., a complete lattice where the underlying ordering encodes the relation of approximation between objects (the top element represents no information, i.e., the meaning of the order is dual to the standard one used in classical domain theory), a closure operator $\rho: C \rightarrow C$ is monotone, idempotent and extensive (i.e., $c \leqslant \rho(c)$). The intuition is quite simple. Monotonicity ensures that the abstraction monotonically approximates domain objects, idempotency means that the process of approximation is performed all at once, while extensivity captures precisely the intended meaning of approximation, i.e., an approximation $\rho(c)$ "contains" less information than its source c. By this approach, the complete lattice $uco(C)$ of all closure operators on C is identified with the so-called lattice of abstract interpretations of C [Cousot and Cousot 1977, 1979], i.e., the complete lattice of all possible abstract domains (modulo isomorphic representation of their objects) of the concrete domain C—where the bottom, i.e., the straightforward abstraction, is C itself.

The order-theoretic reconstruction of the hierarchy of logic program semantics leads naturally to the concept of *uniform closure*, specialized by duality to meet- and join-uniformity, which is the main novel lattice-theoretic notion of the paper. A closure operator ρ on a complete lattice C is meet-uniform when for any non-empty subset $Y \subseteq C$, if all the elements of Y are mapped by ρ to some c then also the infimum $\wedge Y$ is mapped by ρ to c. Let us give a simple example. Consider the classical domain *Sign* depicted in Fig. 1, typically used for sign analysis of integer variables. The abstract domains $A^{\pm 0}$ and $A^{\pm}$ both depicted in Fig. 1 are two proper abstractions of *Sign* (actually, $A^{\pm}$ is in turn an abstraction of $A^{\pm 0}$). These domains correspond to the following closure operators $\rho_{A^{\pm 0}}$ and $\rho_{A^{\pm}}$ defined over *Sign*:

$$\rho_{A^{\pm 0}}(x) = \begin{cases} \mathbb{Z} & \text{if } \quad x = \mathbb{Z}, 0+, 0- \\ x & \text{otherwise;} \end{cases} \qquad \rho_{A^{\pm}}(x) = \begin{cases} \mathbb{Z} & \text{if } \quad x = \mathbb{Z}, 0, 0+, 0- \\ x & \text{otherwise.} \end{cases}$$

It turns out that $\rho_{A^{\pm 0}}$ is not meet-uniform, because $\{x \in Sign \mid \rho_{A^{\pm 0}}(x) = \mathbb{Z}\}$ does not contain its infimum 0, while it is immediate to check that $\rho_{A^{\pm}}$ is meet-uniform, since in this case $\{x \in Sign \mid \rho_{A^{\pm}}(x) = \mathbb{Z}\}$ contains its infimum.

Closure operators are, in general, neither additive nor co-additive (namely, they do not preserve sups or infs), and meet-uniformity is, in general, weaker than co-additivity. For instance, in the example above, although being meet-uniform, $\rho_{A^{\pm}}$ is not co-additive: $\varnothing = \rho_{A^{\pm}}(0+ \ \wedge \ -) \neq \rho_{A^{\pm}}(0+) \wedge \rho_{A^{\pm}}(-) = -$. We prove that given a meet-uniform closure ρ on C, meet-uniformity provides a systematic way for lifting the complete order of C, yet maintaining a complete lattice structure, and achieving co-additivity for ρ relatively to the lifted complete order. The definition of such lifted partial order is obtained by generalizing a construction by Falaschi *et al.* [1993], proposed to relate semantic interpretations for logic programs. Their definition is obtained by lifting the Hoare powerdomain preorder between nonground interpretations (i.e., sets of atoms), based on the relation of instantiation between atoms, so that it becomes a partial order (actually, a complete order). The relevant point of this construction is that it allows one to keep track both of the degree of instantiation and of set inclusion between interpretations. The generalization of that idea is quite simple. Given a complete lattice C and any operator $\rho: C \to C$, for all $x, y \in C$, x is smaller than y in the lifted order for ρ, whenever $\rho(x)$ is smaller than $\rho(y)$ in the original order of C. This would lead in general to a preorder relation, unless ρ is injective. Thus, when two elements are mapped by ρ to the same value, the original partial order is considered: whenever $\rho(x) = \rho(y)$, if x is smaller than y in the original order of C, then x is smaller than y in the lifted order as well. The link with the notion of meet-uniformity is given by the relevant consequences of this definition when ρ is a meet-uniform closure operator. The approach of Falaschi *et al.* [1993] can be then reformulated by means of the closure under instantiation over interpretations, which actually results to be meet-uniform. In the simple example above, the lifted order for *Sign* with respect to the meet-uniform closure $\rho_{A^{\pm}}$ gives rise to the lattice depicted in Fig. 2,

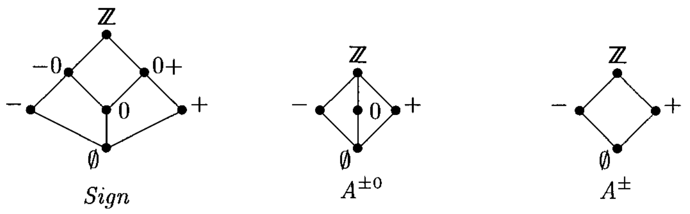


FIG. 1. The domain *Sign* and two its abstractions $A^{\pm 0}$ and $A^{\pm}$.

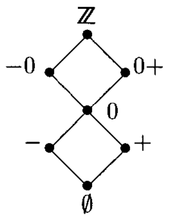


FIG. 2. *Sign* equipped with the lifted order for $\rho_{A^{\pm}}$.

where, e.g., $+$ becomes smaller than -0. Moreover, it is easily seen that for this lifted order, the closure $\rho_{A^{\pm}}$ actually becomes co-additive.

Let us now illustrate in more detail how the general theory of uniform closure operators will be applied in the fields of logic program semantics and abstract domain design.

Order-theoretic reconstruction of the s-semantics of logic programs. The declarative *s-semantics* has been proposed by Falaschi *et al.* [1989] (for a survey see [Bossi *et al.* 1994]) as a semantics modeling more adequately than the canonical least Herbrand model semantics [van Emden and Kowalski 1976] the operational behavior of a logic program as defined by SLD-resolution. In particular, stronger soundness and completeness results hold for the s-semantics, and this allows one to characterize precisely the key observable operational property of computed answer substitutions. Due to these features, the s-semantics has been widely and successfully applied in the area of semantics-based program transformation and analysis (see, e.g., [Bossi and Cocco 1993; Barbuti *et al.* 1993; Codish *e tal.* 1994]). One of the key points of this approach is that the denotations (or interpretations) for a logic program are equivalence classes (w.r.t. renaming of variables) of sets of possibly nonground atoms. However, the original model-theoretic view of s-semantics was unsatisfactory, being based on ad hoc notions of s-truth and s-model [Falaschi *et al.* 1989, Section 4]. These problems have been solved by Falaschi *et al.* [1993], where a nonground interpretation is defined to be a model whenever the corresponding Herbrand interpretation, given by the closure under ground instances, is a model in the standard sense. However, as observed by Falaschi *et al.* [1993], the model intersection property, which allows one to associate with each logic program a canonical model, does not hold in general for these latter models. This is basically due to the fact that plain set inclusion does not adequately reflect this intended meaning of nonground interpretations as models, and instantiation between atoms should be also taken into account. We show that the construction proposed by Falaschi *et al.* [1993] to overcome these problems can be made fully independent from any notion peculiar to logic programming. Actually, we prove that many results given by Falaschi *et al.* [1993] can be obtained as an instance of a general framework which completely reconstructs their approach using pure order-theoretic concepts only.

There are three key observations that led to our construction: (i) the semantics involved in [Falaschi *et al.* 1993] are related each other by abstract interpretation; (ii) the semantics of computed answers is related to that of correct answers by a meet-uniform closure, which is an instance of a generic *downward closure* later discussed in Sections 3 and 4; (iii) the lifted order induced by the downward closure generalizes exactly the new ordering relation introduced by Falaschi *et al.* [1993]. Thus, in our order-theoretic approach, the downward closure becomes co-additive with respect to the lifted order, and, as a consequence, this ensures the existence of the least canonical model. Being independent on specific semantic objects, our results are applicable to other semantics for logic programming in the style of s-semantics, and, more in general, to programming language semantics specified as inductive definitions. For instance, as far as logic programming is concerned, it should not be too hard to generalize many results of the model-theoretic compositional semantics of Bossi *et al.* [1994, Section 5], following the lines of our approach.

It is worth remarking that logic programming is probably the programming paradigm where abstract interpretation ideas have been mostly successful in the study of semantics, as the growing literature on this topic shows (e.g., see [Amato and Levi 1997; Comini and Levi 1994]; Commi *et al.* 1995; Fages and Gori 1996; Giacobazzi 1996; Giacobazzi and Ranzato 1995, 1996]), and therefore our results fit well along this trend.

Order-theoretic foundations of abstract domain refinements. The idea of domain refinement is recurrent in abstract interpretation. Relevant examples include the disjunctive completion [Cousot and Cousot 1979, 1994; Filé and Ranzato 1998; Giacobazzi and Ranzato 1998; Jensen 1997] and the reduced product [Cousot and Cousot 1979], to cite the most known ones. The basic idea is that more expressive abstract domains can be obtained by combining simpler ones or by lifting them by systematically adding new information. A systematic treatment of abstract domain refinements has been given in [Filé *et al.* 1996; Giacobazzi and Ranzato 1997], where a generic refinement is defined to be a lower closure operator—that is, extensivity is replaced by the dual reductivity—on the lattice of abstract interpretations of a given concrete domain. The intuition is still the most natural. Monotonicity of the refinement preserves the relative precision between abstract domains, idempotency ensures that the refinement is performed all at once, and reductivity captures the action of refinement. These kind of operators on abstract domains provide high-level facilities to tune a program analysis in accuracy and cost, and have been included as tools for design aid in modern systems for program analysis, for instance in System Z [Yi and Harrison 1993], in PLAI [Codish *et al.* 1995], and in GAIA [Cortesi *et al.* 1994].

Recently, much attention has been devoted to "invert" abstract domain refinements. For a given refinement $\Re \in lco(uco(C))$, this corresponds to define an operator which simplifies an abstract domain A of input for $\Re$, by returning the domain (if any) which contains the least amount of information required by $\Re$ to get the same expressiveness obtainable from A. Thus, the inverse operator of $\Re$ exists on a class $\mathbb{K} \subseteq uco(C)$ of abstract domains when, for any $A \in \mathbb{K}$, there exists the least (w.r.t. the ordering of $uco(C)$) common abstraction of all the domains D such that $\Re(D) = \Re(A)$. Intuitively, this somehow resembles what the operation of compression does on files. The problem of inverting in the above sense a refinement can be quite hard to solve in a satisfactory way. Moreover, Filé *et al.* [1996] observed that not all refinements can be inverted on a significant class of abstract domains. The problem of inverting the reduced product and disjunctive completion has been solved, introducing, respectively, the notions of complementation [Cortesi *et al.* 1995] and least disjunctive basis [Giacobazzi and Ranzato 1998].

We observe here that the the problem of inverting an abstract domain refinement is closely bound to the concept of uniformity. In fact, since the least common abstraction on the lattice of abstract interpretations coincides with the least upper bound operation, it turns out that a refinement $\mathfrak{R}$ is invertible if and only if $\mathfrak{R}$ is join-uniform. Using a straightforward extension of the notion of uniformity, it can be stated more precisely that $\mathfrak{R}$ is invertible on a class $\mathbb{K}$ of abstract domains if and only if $\mathfrak{R}$ is join-uniform on $\mathbb{K}$. The situation is therefore dual to that above for logic program semantics, i.e., join-uniformity replaces meet-uniformity, and hence it is possible to lift the complete order between abstract domains with respect to an invertible refinement. We argue that this novel order reflects more adequately than the standard one the relation of precision between abstract domains relatively to a given invertible refinement. Moreover, we demonstrate that an invertible refinement and its inverse operator give rise to an adjunction with respect to the lifted order, and this justifies once more the use of the term "inversion" in this context.

Structure of the paper. In Section 2, we recall the basic notations and notions of lattice theory and logic programming used in the paper, and we present a succinct overview of abstract domain theory. In Section 3, motivated from the approach in [Falaschi *et al.* 1993], we introduce the main notion of uniformity, and in Section 4, we study the meet-uniformity for closure operators on complete lattices. In Section 5, we define the lifting of a complete order via a meet-uniform closure operator, and we prove that this process preserves the complete lattice structure and allows the meet-uniform closure to become co-additivity. Using the previous results, in Section 3 we define our order-theoretic generalized semantics, which generalizes the results in [Falaschi *et al.* 1993]. In particular, the order-theoretic generalizations of the notions of model and least model of a logic program are presented, respectively, in Sections 6.1 and 6.2. Section 7 presents the application to the theory of abstract domain refinements. Section 8 concludes, also by sketching some further research directions.

2. PRELIMINRAIES

In this section, we briefly introduce the notation used throughout the paper and summarize some definitions and well-known properties concerning closure operators (for more details see [Birkhoff 1967; Morgado 1960; Ward 1942]), abstract interpretation (see [Cousot and Cousot 1977, 1979]), and logic programming (e.g., see [Apt 1990]).

2.1. Basic Notation

Let C and D be sets. The powerset of C is denoted by $\wp(C)$, and its cardinality by $|C|$. The set-difference between C and D is denoted by $C\setminus D$. If f is a function defined on C and $D \subseteq C$ then $f(D) = \{f(x) \mid x \in D\}$. Functions will be sometimes denoted by Church's lambda notation. By $g \circ f$ we denote the composition of the functions f and g, i.e., $g \circ f = \lambda x.g(f(x))$. The set C equipped with a partial order $\leqslant$ is denoted by $\langle C, \leqslant \rangle$ or simply by $C_\leqslant$. If C is a poset, we usually denote by $\leqslant_C$ the corresponding partial order. A complete lattice C with partial order $\leqslant$, least upper bound (*lub*) $\bigvee$, greatest lower bound (*glb*) $\bigwedge$, top element $\top = \bigwedge \varnothing = \bigvee C$, and bottom element $\bot = \bigvee \varnothing = \bigwedge C$, is denoted by $\langle C, \leqslant, \vee, \wedge, \top, \bot \rangle$. When C is a lattice, $\bigvee_C$, $\bigwedge_C$, $\top_C$ and $\bot_C$ denote the corresponding basic operators and elements. Often, we will slightly abuse notation by denoting lattices with their poset notation. We use $C \cong D$ to denote that the ordered structures C and D are isomorphic. A function $f: C \to D$ between complete lattices is *additive* if for any $Y \subseteq C$, $f(\bigvee_C Y) = \bigvee_D f(Y)$. *Co-additivity* is dually defined.

2.2. Closure Operators

An (*upper*) *closure operator* (or simply *closure*) on a poset $C_\leqslant$ is an operator $\rho: C \to C$ monotone, idempotent and extensive (i.e., $\forall x \in C. x \leqslant \rho(x)$). We denote by $uco(C)$ the set of all closure operators on the poset C. If C is a complete lattice then each closure operator $\rho \in uco(C)$ is uniquely determined by the set of its fixpoints, which is its image $\rho(C)$. A subset $Y \subseteq C$ is the set of fixpoints of a closure operator iff Y is a *Moore-family* of C, i.e., $Y = \{\bigwedge X \mid X \subseteq Y\}$ (where $\top = \bigwedge \varnothing \in Y$). In this case, $\rho_Y = \lambda x. \bigwedge \{y \in Y \mid x \leqslant y\}$ is the corresponding closure on C. The set $\rho(C)$ of fixpoints of $\rho \in uco(C)$ is a complete lattice (with respect to the order $\leqslant$ of C), and more precisely is a complete meet subsemilattice of C (namely, the *glb*'s in C and $\rho(C)$ coincide). However, in general, $\rho(C)$ is not a complete sublattice of C, since the *lub* in $\rho(C)$ might be different from that in C: in fact, $\rho(C)$ is a complete sublattice of C iff ρ is additive. In view of the above equivalence, often we will find it particularly convenient to identify closure operators with their sets of fixpoints, using as notation capital Latin letters; instead, when viewing closures as functions, we will use Greek letters to denote them. In the following, we will keep this soft ambiguity by using both notations and leave to the reader to distinguish their use as functions or sets, according to the context. We denote by $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x \rangle$ the complete lattice of all closure operators on the complete lattice C, where for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i \in I} \subseteq uco(C)$ and $x \in L$:

- $\rho \sqsubseteq \eta$ iff $\forall x \in C. \rho(x) \leqslant \eta(x)$, or equivalently, $\rho \sqsubseteq \eta$ iff $\eta(C) \subseteq \rho(L)$;
- $(\bigsqcup_{i \in I} \rho_i)(x) = x \Leftrightarrow \forall i \in I. \rho_i(x) = x$;
- $(\bigsqcap_{i \in I} \rho_i)(x) = \bigwedge_{i \in I} \rho_i(x)$;
- $\lambda x.\top$ is the top element in $uco(C)$, whereas $\lambda x.x$ is the bottom element.

By $uco^a(C)$ and $uco^{ca}(C)$, we denote, respectively, the subsets of $uco(C)$ consisting of all additive and co-additive closures on C. For a closure operator $\rho \in uco(C)$ and $Y \subseteq C$, the following two properties hold:

(i) $\rho(\bigwedge \rho(Y)) = \bigwedge \rho(Y)$;

(ii) $\rho(\bigvee Y) = \rho(\bigvee \rho(Y))$.

It is known that the complete lattice *uco*(C) is *dual-atomic*, namely each closure different from the top $\lambda x.\top$ is the *glb* of the set of dual-atoms following it (where a dual-atom is an element covered by the top).

Lower closure operators are dually defined; that is, extensivity is replaced by reductivity: $\forall x \in C.\rho(x) \leqslant x$. The set of all lower closure operators on C is denoted by *lco*(C). All the properties of lower closure operators can be derived by duality from those above for upper closures (in particular, *uco*(C) and *lco*(C) are dually isomorphic). Often, upper and lower closures will be called simply closures, and we will leave to the reader to distinguish them according to the context.

2.3. *Galois Connections and Abstract Domains*

If C and A are posets and $\alpha\colon C \to A$, $\gamma\colon A \to C$ are monotone functions such that $\forall c \in C.c \leqslant_C \gamma(\alpha(c))$ and $\forall a \in A.\alpha(\gamma(a)) \leqslant_A a$, then the quadruple (α, C, A, γ) is a *Galois connection* (G.c. for short) between C and A. If in addition $\forall a \in A.\alpha(\gamma(a)) = a$, then (α, C, A, γ) is a *Galois insertion* (G.i. for short) of A in C. In a G.i. (α, C, A, γ), α is onto and γ is 1–1. We also recall that the above definition of Galois connection is equivalent to that of adjunction: if $\alpha\colon C \to A$ and $\gamma\colon A \to C$ then (α, C, A, γ) is a G.c. iff $\forall c \in C.\forall a \in A.\alpha(c) \leqslant_A a \Leftrightarrow c \leqslant_C \gamma(a)$. The map α (γ) is called the *left-adjoint* (*right-adjoint*) to γ (α). This terminology is justified by the well-known fact that one mapping uniquely determines the other. In particular, when C and A are complete lattices, if α is additive, or γ is co-additive, then it determines a Galois connection, where:

(i) $\forall a \in A.\gamma(a) = \bigvee_C \{c \in C \mid \alpha(c) \leqslant_A a\}$;

(ii) $\forall c \in C.\alpha(c) = \bigwedge_A \{a \in A \mid c \leqslant_C \gamma(a)\}$.

For a function f, we denote by f^r (f^l) the corresponding right-adjoint (left-adjoint) function, whenever it exists.

Within the standard Cousot and Cousot [1977, 1979] abstract interpretation framework, a nonstandard program semantics is obtained from the standard one by substituting its domain of computation, called *concrete* (and the basic operations on it), with an *abstract domain* (and corresponding abstract operations). The concrete and abstract domains are complete lattices, where the ordering relations describe the relative precision of the denotations—the top elements representing no information. The concrete domain C and the abstract domain A are related by a Galois connection (α, C, A, γ), where α and γ are called the *abstraction* and *concretization* maps, respectively. Also, A is called an *abstraction* of C. The intuition is that the concretization map gives the concrete value corresponding to an abstract denotation (i.e., its semantics), whereas for a concrete value the abstraction map gives its best (with respect to the ordering of A) abstract approximation. Thus, an abstract value $a \in A$ approximates a concrete value $c \in C$ if $c \leqslant_C \gamma(a)$, or equivalently (by adjunction), if $\alpha(c) \leqslant_A a$. If (α, C, A, γ) is a G.i., each value of the abstract domain is useful in the representation of the concrete domain, because all the elements of A represent distinct members of C, being γ 1–1. It is known that any G.c. may be lifted to a G.i. identifying in an equivalence class those values of the abstract domain with the same concrete meaning. This process is known as *reduction* of the abstract domain.

It has been well-known since [Cousot and Cousot 1979] that abstract domains can be equivalently specified either as Galois insertions or as (sets of fixpoints of) upper closures on the concrete domain. These two approaches are completely equivalent: if $\rho \in uco(C)$ and $A \cong \rho(C)$ (with $\imath: \rho(C) \to A$ and $\imath^{-1}: A \to \rho(C)$ being the isomorphism) then $(\imath \circ \rho, C, A, \imath^{-1})$ is a G.i.; if (α, C, A, γ) is a G.i. then $\rho_A = \gamma \circ \alpha \in uco(C)$ is the closure associated with A such that $\rho_A(C) \cong A$. Actually, an abstract domain A specified by a G.i. (α, C, A, γ) is just a "computer representation" of its logical meaning, namely its image in the concrete domain C, and therefore the essence of A lies with the corresponding closure operator ρ_A. By the above equivalence, it is not restrictive, and often more convenient, to use the closure operator approach to reason about abstract domain properties independently from the representation of the objects. Thus, whenever we will introduce closure operators on some complete lattice C, we will be actually doing a step of abstract interpretation on C. Consequently, we will identify $uco(C)$ with the so-called *lattice of abstract interpretations* of C (cf. [Cousot and Cousot 1977, Section 7] and [Cousot and Cousot 1979, Section 8]), i.e., the complete lattice of all possible abstract domains (modulo isomorphic representation of their objects) of the concrete domain C. The ordering on $uco(C)$ corresponds precisely to the standard order used to compare abstract domains with regard to their precision: A_1 is *more precise* than A_2 (or A_2 is an abstraction of A_1) iff $A_1 \sqsubseteq A_2$ in $uco(C)$. The *lub* and *glb* on $uco(C)$ have therefore the following meaning as operators on domains. Let $\{A_i\}_{i \in I} \subseteq uco(C)$: (i) $\bigsqcup_{i \in I} A_i$ is the most concrete among the domains which are abstractions of all the A_i's, i.e., $\bigsqcup_{i \in I} A_i$ is the *least common abstraction* of all the A_i's; (ii) $\bigsqcap_{i \in I} A_i$ is (isomorphic to) the well-known *reduced product* (basically cartesian product plus reduction) of all the A_i's, or, equivalently, it is the most abstract among the domains (abstracting C) which are more concrete than every A_i.

2.4. Logic Programming Notation

Throughout the paper, *Atom* will denote the set of atoms built over a given first-order language $\mathscr{L}$ (where *Var* denotes the set of variables). A syntactic object is termed ground if it does not contain occurrences of variables. The set of all substitutions (built on $\mathscr{L}$) is denoted by *Sub*. The application of a substitution σ to a syntactic object s is denoted by $s\sigma$. A variable renaming is a substitution which is a bijection on *Var*. A syntactic object t' is more instantiated than t, denoted $t' \preccurlyeq t$, iff there exists $\sigma \in Sub$ such that $t' = t\sigma$. The relation $\preccurlyeq$ is a pre-order on *Atom*. Syntactic objects t_1 and t_2 are equivalent up to renaming, denoted $t_1 \sim t_2$, iff $t_1 \preccurlyeq t_2$ and $t_2 \preccurlyeq t_1$. For the sake of simplicity, we will let a syntactic object denote its equivalence class by renaming. The quotient $Atom_{/\sim}$ becomes partially ordered with respect to $\preccurlyeq$. With abuse of notation, it is still denoted by *Atom*, and it is also called the *nonground Herbrand base*. The subset of *Atom* given by ground atoms is denoted by $Atom_{\varnothing}$, and called the (*ground* or *standard*) *Herbrand base*.

3. DOMAINS OF INTERPRETATIONS AND MEET-UNIFORMITY

In this section, we introduce the concept of (meet-)uniform function defined on complete lattices, which provides the key notion both for order-theoretically reconstructing the model-theoretic semantics of logic programs and for studying abstract domain refinements.

Motivations from logic programming. We recall some of the notions involved in the construction proposed by Falaschi *et al.* [1993]. This is obtained by presenting the different logic program semantics as related by abstract interpretation, similarly to the approaches of Comini and Levi [1994] and Giacobazzi [1996]. In the following, let us assume that a fixed first-order language $\mathscr{L}$ is given; all the subsequent notions are then given with respect to $\mathscr{L}$.

In [Falaschi *et al.* 1993], an *interpretation* is defined to be any subset of the nonground Herbrand base *Atom*. Hence, the domain of interpretations is fixed as the powerset $\langle \wp(Atom), \subseteq \rangle$ of the nonground Herbrand base, ordered by subset inclusion. Some operators on interpretations are defined as follows. Suppose that $I \in \wp(Atom)$.

(i) Closure by instantiation: $\ulcorner I \urcorner = \{A \in Atom \mid \exists B \in I . A \preccurlyeq B\}$;

(ii) Ground elements: $\llcorner I \lrcorner = \{A \in I \mid A \text{ is ground}\}$;

(iii) Ground instances: $gr(I) = \llcorner \ulcorner I \urcorner \lrcorner I = \ulcorner I \urcorner \cap Atom_{\varnothing}$.

Using these operators, we now build a hierarchy of the various domains of interpretations introduced by Falaschi *et al.* [1993]. The operator $\ulcorner \cdot \urcorner$ defines the subset $\wp^{\downarrow}(Atom)$ of $\wp(Atom)$ given by the interpretations closed by instantiation: $\wp^{\downarrow}(Atom) = \{I \in \wp(Atom) \mid I = \ulcorner I \urcorner\}$. Moreover, the set of ground Herbrand interpretations $\wp(Atom_{\varnothing})$ can be in turn defined as the subset of $\wp^{\downarrow}(Atom)$ which is image of the operator *gr* (or $\llcorner \cdot \lrcorner$): $\wp(Atom_{\varnothing}) = \{I \in \wp^{\downarrow}(Atom) \mid I = gr(I)\ (= \llcorner I \lrcorner)\}$.

It is immediate to check that $\ulcorner \cdot \urcorner : \wp(Atom) \to \wp(Atom)$ is an additive closure operator. Hence, its set of fixpoints, equipped with the subset ordering, is a complete sublattice of $\wp(Atom)$, namely $\langle \wp^{\downarrow}(Atom), \subseteq \rangle$ is a complete lattice, where *lub* and *glb* are, respectively, union and intersection. According to the abstract interpretation viewpoint, this means that the domain of interpretations closed by instantiation actually is an abstraction of the basic domain of (nonground) interpretations. This observation makes explicit in the framework of abstract interpretation the intuition that, by considering as domain of interpretations $\wp^{\downarrow}(Atom)$ instead of $\wp(Atom)$, we are actually disregarding some of the information that $\wp(Atom)$ is able to represent.

On the other hand, it is also clear that $gr : \wp^{\downarrow}(Atom) \to \wp^{\downarrow}(Atom)$ is additive and co-additive. Therefore, its image $\langle \wp(Atom_{\varnothing}), \subseteq \rangle$, equipped with the subset ordering, is a complete sublattice of $\langle \wp^{\downarrow}(Atom), \subseteq \rangle$, and hence *lub* and *glb* are union and intersection, respectively. It is worth noting that since *gr* is additive, it is the left-adjoint of the Galois insertion $(gr, \wp^{\downarrow}(Atom), \wp(Atom_{\varnothing}), gr^{r})$, where the right-adjoint $gr^r\colon \wp(Atom_{\varnothing}) \to \wp^{\downarrow}(Atom)$ is defined as $gr^r(I) = \bigcup \{J \in \wp^{\downarrow}(Atom) \mid gr(J) \subseteq I\}$. Hence, this Galois insertion induces the closure operator $cgr = gr^r \circ gr \in uco(\wp^{\downarrow}(Atom))$, and its image $\langle cgr(\wp^{\downarrow}(Atom)), \subseteq\rangle$ is a complete lattice isomorphic to $\langle \wp(Atom_{\varnothing}), \subseteq\rangle$. Moreover, since right-adjoints are always co-additive and the composition of co-additive functions is co-additive, the co-additivity of gr induces the co-additivity of the corresponding closure cgr. Here again, these remarks say that $\wp(Atom_{\varnothing})$ is in turn an abstraction of $\wp^{\downarrow}(Atom)$.

Thus, from the perspective of abstract interpretation, we deal with a hierarchy of abstract domains, as depicted by Fig. 3.

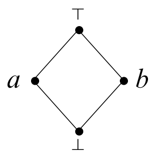
Order-theoretic generalization. From the order-theoretic point of view, we assume that the basic domain of interpretations is merely any complete lattice $\langle C, \leqslant\rangle$. To generalize the syntactic operators defined above from an order-theoretic perspective, it is necessary to assume that the domain of interpretations is a powerset of a poset, ordered by subset inclusion, namely $\langle C, \leqslant\rangle = \langle \wp(Q), \subseteq\rangle$, where $\langle Q, \leqslant_Q\rangle$ is any poset—nevertheless, it should be remarked that this hypothesis will not be necessary for our generalized order-theoretic construction, as we will see later. The generalization is then immediate, since it involves standard and well-known operators used in basic lattice theory. Let $G \subseteq Q$ be fixed (G stands for the ground Herbrand base), and let $I \in \wp(Q)$.

(i) $\lceil I \rceil = {\downarrow} I = \{x \in Q \mid \exists y \in I. x \leqslant_Q y\}$;

(ii) $\lfloor I \rfloor = I \cap G$;

(iii) $gr(I) = \lfloor \lceil I \rceil \rfloor I = ({\downarrow} I) \cap G$.

It is immediate to note that ${\downarrow} \in uco(\wp(Q))$, and, furthermore, it is additive. This closure operator is generally known as the *downward closure*. Moreover, it is worthwhile to observe that $\downarrow$ is not co-additive: in fact, for the lattice L depicted below, we have that ${\downarrow}(\{a\} \cap \{b\}) = {\downarrow}\varnothing = \varnothing$, while ${\downarrow}\{a\} \cap {\downarrow}\{b\} = \{\bot\}$.



As far as $\lfloor \cdot \rfloor\colon \wp(Q) \to \wp(Q)$ is concerned, it is immediate to note that $\lfloor \cdot \rfloor$ is both additive and co-additive.

$$\begin{tikzcd}
\langle \wp(Atom), \subseteq, \cup, \cap, Atom, \emptyset\rangle \arrow[d, "{\lceil \cdot \rceil \,\in\, uco(\wp(Atom)) \text{ additive}}"] \\
\langle \wp^{\downarrow}(Atom), \subseteq, \cup, \cap, Atom, \emptyset\rangle \arrow[d, "{gr \,:\, \wp^{\downarrow}(Atom) \to \wp^{\downarrow}(Atom) \text{ additive and co-additive}}"] \\
\langle \wp(Atom_{\emptyset}), \subseteq, \cup, \cap, Atom_{\emptyset}, \emptyset\rangle
\end{tikzcd}$$

FIG. 3. The hierarchy of domains of interpretations.

We propose a generalization of the hierarchy of domains of interpretations recalled above which only takes into consideration some of the order-theoretic properties of the functions involved in that construction. To mimic the closure by instantiation relating $\wp(Atom)$ and $\wp^{\downarrow}(Atom)$, we assume that a closure operator $\rho \in uco(C)$ is defined on the generalized domain (i.e., any complete lattice) $\langle C, \leqslant \rangle$. Thus, the closure ρ formalizes a step of abstraction on the domain C. In contrast with $\ulcorner \cdot \urcorner$ and its immediate generalization $\downarrow$ above, we do not require the additivity of ρ. Instead, we focus on a peculiar and somehow hidden order-theoretic property of the downward closure $\downarrow \in uco(\wp(P))$, which we call *meet-uniformity*. To the best of our knowledge, this property of functions has not been previously considered in the literature.[1] Let C be a complete lattice and S be any set.

DEFINITION 3.1. A function $f\colon C \to S$ is *meet-uniform* if for all $x \in C$ and $Y \subseteq C$, such that $Y \neq \varnothing$,

$$(\forall y \in Y. f(y) = f(x)) \Rightarrow f\left(\bigwedge_C Y\right) = f(x)$$

In other terms, if S is thought of as a lattice, then a mapping f from C to S enjoys the property of meet-uniformity if it is co-additive for any family of elements for which f is constant. Note that the above condition is vacuously satisfied for Y singleton, and therefore, in the following proofs of meet-uniformity, we will assume $|Y| > 1$. Further, notice that if f is 1–1 then it is trivially meet-uniform. It is also worth noting that in the above definition the complete lattice C can be easily generalized to any algebra equipped with a finitary or infinitary operation which replaces the role played by the *glb* of C. However, this generality does not contribute significantly to our aims. As far as closures are concerned, we denote by $uco^*(C) = \{\rho \in uco(C) \mid \rho \text{ is meet-uniform}\}$ the set of all meet-uniform closure operators on the complete lattice C.

As announced above, although not being co-additive, the downward closure is meet-uniform, whenever the poset Q satisfies the ascending chain condition (ACC for short, i.e., it does not contain infinite strictly increasing chains).

THEOREM 3.2. *If Q satisfies the ACC then $\downarrow$ is meet-uniform.*

To demonstrate this result, we need some notation and a preliminary lemma. The operator $max\colon \wp(Q) \to \wp(Q)$, giving the maximal elements of any subset S of Q, is defined as $max(S) = \{x \in S \mid \forall y \in S. x \leqslant_Q y \Rightarrow x = y\}$.

[1] The only account we found on a similar property for closure operators is in [Adaricheva and Gorbunov 1990]. The authors introduced in Definition 1.1 the notion of equational closure on a complete lattice L, as a strict meet-uniform upper closure operator ρ on L, where, in addition, closed elements distribute in L and $\rho(L)$ is join-generated by the dual-compact elements of L. The lattice-theoretic structure of equational closures is studied by Adaricheva and Gorbunov [1990] in Section 5. However, because meet-uniformity is in general weaker than equationality, their results are not applicable to study the properties of generic meet-uniform closure operators on complete lattices.

LEMMA 3.3. *Let Q be a poset.*

(i) *For any $S \in \wp(Q)$, $max(S) = max(\downarrow S)$.*

(ii) *If Q satisfies the ACC then, for all $S \subseteq Q$, $\downarrow S = \downarrow max(S)$.*

(iii) *If Q satisfies the ACC then, for all $S, T \subseteq Q$, $\downarrow S = \downarrow T \Leftrightarrow max(S) = max(T)$.*

Proof. (i) ($\subseteq$) Let $x \in max(S)$ and consider any $y \in \downarrow S$ such that $x \leqslant_Q y$. Hence, there exists $z \in S$ such that $y \leqslant_Q z$. Thus, $x \leqslant_Q z$, from which $x = z$. Therefore, $x = y$, and $x \in max(\downarrow S)$.

($\supseteq$) Let $x \in max(\downarrow S)$. If $x \in S$ then $x \in max(\downarrow S)$, since $S \subseteq \downarrow S$. Otherwise, $x \in \downarrow S \backslash S$. Thus, there exists $y \in S$ such that $x <_Q y$, which is a contradiction.

(ii) ($\supseteq$) From $max(S) \subseteq S$ one get $\downarrow max(S) \subseteq \downarrow S$.

($\subseteq$) Let $x \in \downarrow S$. Then, there exists $y_0 \in S$ such that $x \leqslant_Q y_0$. If $y_0 \in max(S)$ then $x \in \downarrow max(S)$. Otherwise, there exists $y_1 \in S$ such that $y_0 \leqslant_Q y_1$ and $y_0 \neq y_1$, i.e., $y_0 <_Q y_1$. If $y_1 \in max(S)$ then $x \in \downarrow max(S)$, otherwise, as before, we pick out another $y_2 \in S$ such that $y_0 <_Q y_1 <_Q y_2$. Iterating this constructive process, we would get an infinite strictly increasing chain, namely, a contradiction. Thus, there exists $k \in \mathbb{N}$ such that $y_k \in max(S)$ and $x \leqslant_Q y_k$, i.e., $x \in \downarrow max(S)$.

(iii) ($\Rightarrow$) $max(S) = (\text{by (i)}) = max(\downarrow S) = max(\downarrow T) = (\text{by (i)}) = max(T)$.

($\Leftarrow$) $\downarrow S = (\text{by (ii)}) = \downarrow max(S) = \downarrow max(T) = (\text{by (ii)}) = \downarrow T$. ∎

Proof of Theorem 3.2. Consider any family $\{S_i\}_{i \in I} \subseteq \wp(A)$ (where $|I| > 1$) and $T \in \wp(A)$ such that $\forall i \in I. \downarrow S_i = \downarrow T$. Let us prove that $\downarrow(\bigcap_{i \in I} S_i) = \downarrow T$.

($\subseteq$) By monotonicity of $\downarrow$.

($\supseteq$) Let $x \in \downarrow T$. By Lemma 3.3 (ii), $\downarrow T = \downarrow max(T)$, and, therefore, there exists $y \in max(T)$ such that $x \leqslant_Q y$. By Lemma 3.3 (iii), for all $i \in I$, $max(S_i) = max(T)$, and hence, $max(T) \subseteq S_i$. Thus, $max(T) \subseteq \bigcap_{i \in I} S_i$, and $y \in \bigcap_{i \in I} S_i$, i.e., $x \in \downarrow(\bigcap_{i \in I} S_i)$. ∎

It is worth remarking that when Q does not satisfy the ACC, Theorem 3.2 in general does not hold, as the following example shows.

EXAMPLE 3.4. Consider the poset $\mathbb{N}$ of natural numbers, equipped with the standard ordering, which does not satisfy the ACC, and the subsets O and E of, respectively, odd and even numbers. Then, $\downarrow O = \downarrow E = \mathbb{N}$, whilst $\downarrow(O \cap E) = \downarrow \varnothing = \varnothing$, i.e., $\downarrow$ is not meet-uniform.

Since *Atom*, equipped with the partial order of instantiation $\preccurlyeq$, is evidently a poset satisfying the ascending chain condition, as a consequence of Theorem 3.2, we get that the closure by instantiation $\ulcorner \cdot \urcorner : \wp(Atom) \to \wp(Atom)$ is meet-uniform. We will see later that meet-uniformity is the key order-theoretic property that allows us to generalize the results in [Falaschi *et al.* 1993]. Then, in our generalized hierarchy, we assume that the first step of abstraction is given by a meet-uniform closure operator $\rho \in uco^*(C)$. The next step consists in generalizing the grounding operator defined on $\wp^{\downarrow}(Atom)$. As observed above, gr and $\llcorner \cdot \lrcorner$ coincide on $\wp^{\downarrow}(Atom)$, and $gr : \wp^{\downarrow}(Atom) \to \wp^{\downarrow}(Atom)$ is both additive and co-additive. Our

$$\begin{tikzcd}
\langle C, \leq, \vee, \wedge, \top_C, \bot_C \rangle \arrow[d, "\rho \in uco^*(C)"] \\
\langle \rho(C), \leq, \vee_{\rho(C)}, \wedge, \top_C, \rho(\bot_C) \rangle \arrow[d, "g : \rho(C) \to \rho(C) \text{ co-additive}"] \\
\langle g(\rho(C)), \leq, \vee_{g(\rho(C))}, \wedge, \top_C, g(\rho(\bot_C)) \rangle
\end{tikzcd}$$

FIG. 4. The generalized order-theoretic hierarchy of domains.

generalization simply considers any co-additive map defined on the image $\langle \rho(C), \leqslant \rangle$, namely any $g: \rho(C) \to \rho(C)$ which is co-additive. Clearly, being co-additive, g is also monotone. Hence, in this case the key order-theoretic property of the grounding operator relating the domain of interpretations closed by instantiation and the domain of ground interpretations is its co-additivity. As we noted above, this actually is an abstract interpretation step. However, for our purposes, we only need to consider a co-additive map. Also, notice that the image $\langle g(\rho(C)), \leqslant \rangle$ of the co-additive function g, is a complete lattice w.r.t. the induced order $\leqslant$, where the *glb* coincides with that of $\rho(C)$ (which in turn coincides with that in C), i.e., $\langle g(\rho(C)), \leqslant \rangle$ is a Moore-family of both $\langle C, \leqslant \rangle$ and $\langle \rho(C), \leqslant \rangle$. Our order-theoretic generalized hierarchy is summarized by Fig. 4.

4. MEET-UNIFORM CLOSURE OPERATORS

In this section, we mainly concentrate on studying the properties of *meet-uniform closure operators* on complete lattices. In the following, let us assume that $\langle C, \leqslant \rangle$ is any complete lattice.

As we noted above, each injective function is meet-uniform. As far as closures are concerned, obviously, the only injective closure is the identity. Thus, injectivity is not relevant to characterize meet-uniformity of closure operators. Moreover, the following remark shows that, for any closure operator, the dual property of *join-uniformity* is always satisfied.

Remark 4.1. Each $\rho \in uco(C)$ is join-uniform.

Proof. Let $Y \subseteq C$ and $x \in C$, with $|Y| > 1$, such that for any $y \in Y$, $\rho(y) = \rho(x)$. Then, by point (ii) in Section 2.2 and by idempotency, $\rho(\bigvee Y) = \rho(\bigvee \rho(Y)) = \rho(\rho(x)) = \rho(x)$. ▮

Clearly, the bottom element of $uco(C)$, i.e., the identity operator, belongs to $uco^*(C)$. Furthermore, the following result holds.

THEOREM 4.2. *$uco^*(C)$ is a Moore-family of $uco(C)$.*

Proof. Clearly, $\lambda x. \top \in uco^*(C)$. Then, let us consider $\{\rho_i\}_{i \in I} \subseteq uco^*(C)$, with $|I| > 1$. We show that $\bigsqcap_{i \in I} \rho_i \in uco^*(C)$. Consider any $Y \subseteq C$ and $x \in C$ such that for all $y \in Y$, $(\bigsqcap_{i \in I} \rho_i)(y) = (\bigsqcap_{i \in I} \rho_i)(x)$. If we prove that for any $i \in I$, $\rho_i(\bigwedge Y) = \rho_i(x)$, the thesis follows, since $(\bigsqcap_{i \in I} \rho_i)(\bigwedge Y) = \bigwedge_{i \in I} \rho_i(\bigwedge Y) = \bigwedge_{i \in I} \rho_i(x) = (\bigsqcap_{i \in I} \rho_i)(x)$. Thus, consider any $j \in I$ and $y \in Y$. Then, $y \leqslant (\bigsqcap_{i \in I} \rho_i)(y) = (\bigsqcap_{i \in I} \rho_i)(x) = \bigwedge_{i \in I} \rho_i(x)$, and, analogously, $x \leqslant \bigwedge_{i \in I} \rho_i(y)$. Hence, $y \leqslant \rho_j(x)$ and

$x \leqslant \rho_j(y)$, from which $\rho_j(y) = \rho_j(x)$ easily follows. Therefore, for all $i \in I$, $\rho_i(\bigwedge Y) = \rho_i(x)$. ▮

As a consequence, $uco^*(C)$ is a complete lattice with respect to the order inherited from $uco(C)$. The next example shows that, in general, $uco^*(C)$ is not a complete sublattice of $uco(C)$.

Example 4.3. Consider the finite lattice L depicted in Section 3 (the four-point "rhombus" lattice). The closure operators on L are as follows:

$$\rho_1 = \{\top\}, \qquad \rho_2 = \{\top, a\}, \qquad \rho_3 = \{\top, \bot\}, \qquad \rho_4 = \{\top, b\},$$
$$\rho_5 = \{\top, a, \bot\}, \qquad \rho_6 = \{\top, b, \bot\}, \qquad \rho_7 = \{\top, a, b, \bot\},$$

and therefore $uco(L)$ is the lattice depicted in Fig. 5. It is then simple to verify that $uco^*(C)$ is the lattice depicted in Fig. 5. In fact, the only closure which is not meet-uniform is ρ_3: $\rho_3(a) = \rho_3(b) = \top$, whilst $\rho_3(a \wedge b) = \rho_3(\bot) = \bot$. Also, observe that $uco^*(L)$ is not a sublattice of $uco(L)$.

The example above allows one to draw the following two additional consequences.

(i) The property of dual-atomicity of $uco(C)$ does not hold anymore for the complete lattice $uco^*(C)$.

(ii) The composition of two meet-uniform closure operators, whenever this is a closure, in general, is not meet-uniform. In fact, for the meet-uniform closures ρ_5 and ρ_6 above, one has that $\rho_5 \circ \rho_6 = \rho_6 \circ \rho_5 = \rho_3$ is not meet-uniform.

Let S be a set, $f\colon C \to S$, and consider the corresponding equivalence relation $\approx_f$ on C induced by f: $x \approx_f y \Leftrightarrow f(x) = f(y)$. The following definition assigns a canonical representative to each equivalence class defined by $\approx_f$.

Definition 4.4. For any $x \in C$, define $\nabla_f(x) = \bigwedge [x]_{\approx_f} (= \bigwedge \{y \in C \mid f(y) = f(x)\})$.

If f is meet-uniform then it is immediate to observe that this is a good definition, i.e., for any $x \in C$, $\nabla_f(x) \approx_f x$. Further, it is worth noticing that if $f = \rho \in uco(C)$ then $\nabla_\rho(x) = \nabla_\rho(\rho(x))$. We will use these remarks in the following proofs. The following observation provides an alternative characterization of meet-uniformity. Let $\langle A, \leqslant_A \rangle$ be any poset.

Proposition 4.5. *Let $f\colon C \to A$ be monotone. f is meet-uniform iff for any $x \in C$, $\nabla_f(x) \approx_f x$.*

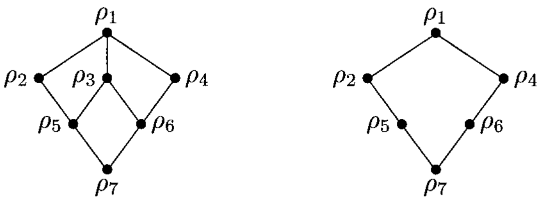


FIG. 5. The lattices $uco(L)$, on the left, and $uco^*(L)$, on the right.

Proof. ($\Rightarrow$) Obvious.

($\Leftarrow$) Consider any $Y \subseteq C$ (with $|Y| > 1$) and $x \in C$ such that $f(y) = f(x)$, for all $y \in Y$. Obviously, $f(\bigwedge_C Y) \leqslant_A f(x)$. On the other hand, by hypothesis, $\nabla_f(x) \approx_f x$, and therefore $f(x) = f(\nabla_f(x)) \leqslant_A f(\bigwedge_C Y)$. Thus, $f(\bigwedge_C Y) = f(x)$. ▌

This observation allows us to give the following slight generalization of the notion of meet-uniformity. Let $f: C \to A$ be monotone and $K \subseteq C$. Then, we define f to be *meet-uniform on K* (or *K-meet-uniform*) when for any $x \in K$, $\nabla_f(x) \approx_f x$. Thus, by Proposition 4.5, f is meet-uniform iff f is meet-uniform on C. In particular, given a complete lattice, we denote by $uco^*_K(C)$ the set of all closures on C which are meet-uniform on K. It is simple to verify that the above Theorem 4.2 admits a straightforward generalization for this notion of K-meet-uniformity, and therefore $uco^*_K(C)$ is a Moore-family of $uco(C)$. We will see the usefulness of this more general concept of meet-uniformity later in Section 7.

For the downward closure of Section 3, a simple characterization of the canonical representative of Definition 4.4 can be given as follows.

Remark 4.6. If Q satisfies the ACC then, for each $I \in \wp(Q)$, $\nabla_\downarrow(I) = max(I)$.

Proof. By Lemma 3.3 (iii), $\nabla_\downarrow(I) = \bigcap \{J \in \wp(Q) \mid \downarrow J = \downarrow I\} = \bigcap \{J \in \wp(Q) \mid max(J) = max(I)\}$. Moreover, since $max(max(I)) = max(I)$, $\nabla_\downarrow(I) \subseteq max(I)$. On the other hand, if $max(J) = max(I)$, then $max(I) \subseteq J$, and therefore $\nabla_\downarrow(I) = max(I)$. ▌

We close this section by observing that meet-uniformity can be induced on closure operators by means of Galois connections. By a slight abuse of terminology, a G.c. (α, C, A, γ) is defined to be meet-uniform if the map α is meet-uniform. The next result shows that meet-uniform G.c.'s induce meet-uniform closures.

PROPOSITION 4.7. A G.c. (α, C, A, γ) is meet-uniform iff $\gamma \circ \alpha \in uco^*(C)$.

Proof. We only prove the "only if" part, since the other direction is similar. Consider any $Y \subseteq C$ and $x \in C$ such that $\forall y \in Y. \gamma(\alpha(y)) = \gamma(\alpha(x))$. Then, for all $y \in Y$, $\alpha(\gamma(\alpha(y))) = \alpha(\gamma(\alpha(x)))$, and, since $\alpha \circ \gamma \circ \alpha = \alpha$, we get $\alpha(y) = \alpha(x)$. Hence, $\alpha(\bigwedge_C Y) = \alpha(x)$, from which, $\gamma(\alpha(\bigwedge_C Y)) = \gamma(\alpha(x))$. ▌

5. LIFTING COMPLETE ORDERS VIA MEET-UNIFORM CLOSURES

The following key definition is obtained as an obvious generalization of the partial order introduced by [Falaschi *et al.* 1993] (see Section 6.2 below). Let $\langle C, \leqslant_C \rangle$ and $\langle A, \leqslant_A \rangle$ be posets, and consider any function $f: C \to A$.

DEFINITION 5.1. The *lifting via f* of the ordering $\leqslant_C$ is the relation $\leqslant^f_C$ on C defined as follows:

$$x \leqslant^f_C y \quad \text{iff} \quad f(x) \leqslant_A f(y) \quad \& \quad (f(y) \leqslant_A f(x) \Rightarrow x \leqslant_C y).$$

In the following, we study the consequences of this notion. First, it is straightforward to note that the above definition actually is correct.

LEMMA 5.2. $\leqslant_C^f$ is a partial order on C.

Let us assume that the function f is monotone. Then, observe that $\leqslant_C \subseteq \leqslant_C^f$, i.e., for all $x, y \in C$, $x \leqslant_C y \Rightarrow x \leqslant_C^f y$. Also, if C is bounded, with top and bottom elements (w.r.t. $\leqslant_C$) $\top$ and $\bot$, then C is bounded for the lifted order too, where $\top$ and $\bot$ are still the top and the bottom w.r.t. $\leqslant_C^f$.

In the following, we will focus on closure operators, because, in our generalized hierarchy of semantics, f will play the role of an abstraction map, which is therefore uniquely determined by a closure operator. For the next result, let us assume that $\langle C, \leqslant \rangle$ is a mere poset.

PROPOSITION 5.3. *If $\rho, \eta \in uco(C_{\leqslant})$ and $\eta \sqsubseteq \rho$, then $\eta \in uco(C_{\leqslant^\rho})$.*

Proof. Idempotency of η is clearly preserved. $\leqslant^\rho$-extensivity of η easily follows from the fact that $\leqslant \subseteq \leqslant^\rho$. We prove monotonicity: $x \leqslant^\rho y \Rightarrow \eta(x) \leqslant^\rho \eta(y)$. First, since $\eta \sqsubseteq \rho$, we have that $\rho \circ \eta = \eta \circ \rho = \rho$. Thus, from the hypothesis $\rho(x) \leqslant \rho(y)$, we get $\eta(\rho(x)) = \rho(\eta(x)) \leqslant \rho(\eta(y)) = \eta(\rho(y))$. On the other hand, assume now that $\rho(\eta(y)) \leqslant \rho(\eta(x))$. Hence, we have that $\rho(y) \leqslant \rho(x)$, which, by hypothesis, implies $x \leqslant y$. Then, by $\leqslant$-monotonicity of η, we get $\eta(x) \leqslant \eta(y)$, which concludes the proof. ∎

As an immediate consequence of the above result, we get that if $\rho \in uco(C_{\leqslant})$ then $\rho \in uco(C_{\leqslant^\rho})$. Let us now give a simple example of lifting a partial order via a closure operator.

EXAMPLE 5.4. Consider the lattice $\langle L, \leqslant \rangle$ of Example 4.3, and the closures $\rho_2 = \{\top, a\}$ and $\rho_3 = \{\top, \bot\}$ defined on L. It is easy to verify that lifting $\leqslant$ via ρ_3 changes nothing in the structure of L, i.e., $\leqslant\, = \,\leqslant^{\rho_3}$. Instead, lifting $\leqslant$ via ρ_2 gives the chain $\{\bot <^{\rho_2} a <^{\rho_2} b <^{\rho_2} \top\}$.

From now on, let $\langle C, \leqslant, \vee, \wedge, \top, \bot \rangle$ be a complete lattice. A key property of lifting a complete order via a meet-uniform closure operator is that this step preserves the complete lattice structure.

THEOREM 5.5. *If $\langle C, \leqslant \rangle$ is a complete lattice and $\rho \in uco^*(C_{\leqslant})$ then $\langle C, \leqslant^\rho \rangle$ is a complete lattice.*

The proof of this theorem consists in giving explicit characterizations of *lub*'s and *glb*'s for the lifted order. Given a meet-uniform closure $\rho \in uco^*(C)$, recall from Definition 4.4 that for each $x \in C$, $\nabla_\rho(x) = \bigwedge \{y \in C \mid \rho(y) = \rho(x)\}$. Then, for any subset $Y \subseteq C$, let us give the following definitions:

$$\dot{\bigvee} Y = \bigvee \left(\left\{ y \in Y \,\middle|\, \nabla_\rho\left(\bigvee Y\right) \leqslant y \right\} \cup \left\{ \nabla_\rho\left(\bigvee Y\right) \right\} \right);$$

$$\underset{\cdot}{\bigwedge} Y = \begin{cases} \bigwedge \{y \in Y \mid \rho(y) = \rho(x)\} & \text{if} \quad \exists x \in Y. \bigwedge \rho(Y) = \rho(x); \\ \bigwedge \rho(Y) & \text{otherwise.} \end{cases}$$

To prove that $\dot{\bigvee}$ and $\dot{\bigwedge}$ actually are the *lub* and *glb* for the lifted order, we need the following preliminary lemma.

LEMMA 5.6. *For all* $Y \subseteq C$, $\rho(\dot{\bigvee} Y) = \rho(\bigvee Y)$.

Proof. The following equalities prove the claim:

$$\rho\left(\dot{\bigvee} Y\right) = \rho\left(\bigvee\left(\left\{y \in Y \,\middle|\, \nabla_\rho\left(\bigvee Y\right) \leqslant y\right\} \cup \left\{\nabla_\rho\left(\bigvee Y\right)\right\}\right)\right)$$

(by point (ii) in Section 2.2)

$$= \rho\left(\bigvee\left(\left\{\rho(y) \in Y \,\middle|\, \nabla_\rho\left(\bigvee y\right) \leqslant y\right\} \cup \left\{\rho\left(\nabla_\rho\left(\bigvee Y\right)\right)\right\}\right)\right)$$

(by meet-uniformity of ρ)

$$= \rho\left(\bigvee\left(\left\{\rho(y) \in Y \,\middle|\, \nabla_\rho\left(\bigvee Y\right) \leqslant y\right\} \cup \left\{\rho\left(\bigvee Y\right)\right\}\right)\right)$$

(by point (ii) in Section 2.2)

$$= \rho\left(\bigvee\left(\left\{y \in Y \,\middle|\, \nabla_\rho\left(\bigvee Y\right) \leqslant y\right\} \cup \left\{\bigvee Y\right\}\right)\right)$$

$$= \rho\left(\bigvee Y\right). \quad \blacksquare$$

THEOREM 5.7. $\dot{\bigvee}$ and $\dot{\bigwedge}$ are the lub and glb in $\langle C, \leqslant^\rho \rangle$.

Proof. Let us first prove that $\dot{\bigvee}$ is the lub. Consider any $Y \subseteq C$.

(a) Let us show that if $y \in Y$ then $y \leqslant^\rho \dot{\bigvee} Y$.

(a$_1$) $\rho(y) \leqslant \rho(\dot{\bigvee} Y)$: Immediate from Lemma 5.6.

(a$_2$) $\rho(\dot{\bigvee} Y) \leqslant \rho(y) \Rightarrow y \leqslant \dot{\bigvee} Y$: Since, from the hypothesis and Lemma 5.6, we get $\rho(y) = \rho(\dot{\bigvee} Y) = \rho(\bigvee Y)$, we also have that $\nabla_\rho(\bigvee Y) = \nabla_\rho(\rho(\bigvee Y)) = \nabla_\rho(\rho(y)) = \nabla_\rho(y)$, and this implies $y \leqslant \dot{\bigvee} Y$.

(b) Assume that there exists $u \in C$ such that $\forall y \in Y. y \leqslant^\rho u$. Then, we prove that $\dot{\bigvee} Y \leqslant^\rho u$.

(b$_1$) $\rho(\dot{\bigvee} Y) \leqslant \rho(u)$: From the hypothesis follows that $\bigvee \rho(Y) \leqslant \rho(u)$. Hence, by Lemma 5.6, $\rho(\dot{\bigvee} Y) = \rho(\bigvee Y) = \rho(\bigvee \rho(Y)) \leqslant \rho(\rho(u)) = \rho(u)$.

(b$_2$) $\rho(u) \leqslant \rho(\dot{\bigvee} Y) \Rightarrow \dot{\bigvee} Y \leqslant u$: By hypothesis and Lemma 5.6, $\rho(u) = \rho(\bigvee Y)$. First, we verify that $\nabla_\rho(\bigvee Y) \leqslant u$: in fact, $\nabla_\rho(\bigvee Y) = \nabla_\rho(\rho(\bigvee Y)) = \nabla_\rho(\rho(u)) = \nabla_\rho(u) \leqslant u$. Second, if $\nabla_\rho(\bigvee Y) \leqslant y$ then $y \leqslant u$: as above, $\nabla_\rho(\bigvee Y) = \nabla_\rho(u)$, and therefore, $\nabla_\rho(u) \leqslant y$. Thus, $\rho(\nabla_\rho(u)) = \rho(u) \leqslant \rho(y)$. Since, by hypothesis, $\rho(u) \leqslant \rho(y) \Rightarrow y \leqslant u$, we get the desired $y \leqslant u$.

Let us now prove that $\dot{\bigwedge}$ is the *glb*. Consider any $Y \subseteq C$. We distinguish between the two mutually exclusive branches of the definition of $\dot{\bigwedge}$.

(a) There exists $x \in Y$ such that $\bigwedge \rho(Y) = \rho(x)$, and hence, $\dot{\bigwedge}\, Y = \bigwedge \{ y \in Y \mid \rho(y) = \rho(x) \}$. Since ρ is meet-uniform, $\rho(\dot{\bigwedge}\, Y) = \rho(x) = \bigwedge \rho(Y)$. We will use this observation during the proof.

(a$_1$) $\forall y \in Y.\ \dot{\bigwedge}\, Y \leqslant^{\rho} y$: First, $\rho(\dot{\bigwedge}\,(Y)) = \rho(x) \leqslant \rho(y)$ holds. Second, $\rho(y) \leqslant \rho(\dot{\bigwedge}\, Y) \Rightarrow \dot{\bigwedge}\, Y \leqslant y$ holds too, since the premise is equivalent to $\rho(y) = \rho(x)$.

(a$_2$) If there exists $l \in C$ such that $\forall y \in Y. l \leqslant^{\rho} y$ then $l \leqslant^{\rho} \dot{\bigwedge}\, Y$: first, $\rho(l) \leqslant \rho(\dot{\bigwedge}\, Y)$, since $\rho(\dot{\bigwedge}\, Y) = \rho(x)$, and, from the hypothesis, $\rho(l) \leqslant \bigwedge \rho(Y) = \rho(x)$. Second, the implication $\rho(\dot{\bigwedge}\, Y) \leqslant \rho(l) \Rightarrow l \leqslant \dot{\bigwedge}\, Y$ holds, since its premise implies $\rho(x) = \rho(l)$, and, if $y \in Y$ is such that $\rho(y) = \rho(x)$ then $\rho(y) = \rho(l)$. By exploiting the hypothesis $l \leqslant^{\rho} y$, this implies $l \leqslant y$, which in turn implies $l \leqslant \dot{\bigwedge}\, Y$.

(b) Assume the other branch, i.e., $\forall y \in Y. \bigwedge \rho(Y) < \rho(y)$. In this case, $\dot{\bigwedge}\, Y = \bigwedge \rho\,(Y)$.

(b$_1$) $\forall y \in Y.\ \dot{\bigwedge}\, Y \leqslant^{\rho} y$: First, $\rho(\dot{\bigwedge}\, Y) = \rho(\bigwedge \rho(Y)) = \bigwedge \rho(Y) \leqslant \rho(y)$. Second, the implication $\rho(y) \leqslant \rho(\dot{\bigwedge}\, Y) \Rightarrow \dot{\bigwedge}\, Y \leqslant y$, i.e., $\rho(y) = \bigwedge \rho(Y) \Rightarrow \dot{\bigwedge}\, Y \leqslant y$, trivially holds, since, by hypothesis, its premise is never satisfied.

(b$_2$) If there exists $l \in C$ such that $\forall y \in Y. l \leqslant^{\rho} y$ then $l \leqslant^{\rho} \dot{\bigwedge}\, Y$: first, $\rho(l) \leqslant \rho(\dot{\bigwedge}\, Y) = \bigwedge \rho(Y)$, since, by hypothesis, $\forall y \in Y. \rho(l) \leqslant \rho(y)$. Second, $\rho(\dot{\bigwedge}\, Y) \leqslant \rho(l) \Rightarrow l \leqslant \dot{\bigwedge}\, Y$ holds, since it is equivalent to $\rho(l) = \bigwedge \rho(Y) \Rightarrow l \leqslant \bigwedge \rho(Y)$, which is true, because $l \leqslant \rho(l)$. ∎

Note that the singleton $\{\nabla_{\rho}(\bigvee Y)\}$ has been added in the above definition of $\dot{\bigvee}$ because if $\{ y \in Y \mid \nabla_{\rho}(\bigvee Y) \leqslant y \}$ results to be empty then $\nabla_{\rho}(\bigvee Y)$ is taken as *lub*, rather than $\bigvee \varnothing = \bot$. Also, as observed above in all generality, note that $\top$ and $\bot$ remain unchanged after the lifting of the ordering: in fact, $\dot{\bigvee}\, C = \dot{\bigwedge}\, \varnothing = \top$ and $\dot{\bigwedge}\, C = \dot{\bigvee}\, \varnothing = \bot$. As a consequence of Theorem 5.7, we get the following fact, which will be useful in the following sections.

COROLLARY 5.8. *For any* $Y \subseteq C$, $\dot{\bigwedge}\, \rho(Y) = \bigwedge \rho(Y)$.

It is important to remark that meet-uniformity is the crucial property of the closure ρ that allows us to prove Theorem 5.5. In fact, the following example shows that if the closure is not meet-uniform, then, in general, the lifted order does not give rise to a complete lattice.

EXAMPLE 5.9. Consider the finite lattice $\langle C, \leqslant \rangle$, depicted in Fig. 6, and consider the closure $\rho \in uco(C)$ defined by $\rho(C) = \{\top, c, d, \bot\}$. It is immediate to verify that ρ is not meet-uniform: in fact, $\rho(a) = \rho(b) = \top$, but $\rho(a \wedge b) = \rho(\bot) = \bot$. In this case, the lack of meet-uniformity for ρ implies that the lifted poset $\langle C, \leqslant^{\rho} \rangle$, depicted in Fig. 6, is not a lattice anymore, because, by definition of $\leqslant^{\rho}$, $d \leqslant^{\rho} a$ and $c \leqslant^{\rho} b$.

There is a further remarkable consequence of lifting a complete order $\leqslant$ via a meet-uniform closure operator ρ. In fact, by this process, we upgrade the properties enjoyed by ρ whenever considered w.r.t. the lifted complete order $\leqslant^{\rho}$: ρ becomes a co-additive closure operator.

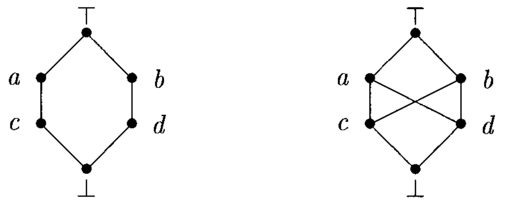


FIG. 6. The lattice $\langle C, \leqslant \rangle$ on the left, and the poset $\langle C, \leqslant^{\rho} \rangle$ on the right.

THEOREM 5.10. *If $\rho \in uco^*(C_{\leqslant})$ then $\rho \in uco^{ca}(C_{\leqslant^{\rho}})$.*

Proof. By Proposition 5.3, ρ still is a closure operator on $\langle C, \leqslant^{\rho} \rangle$. Let us show that it is co-additive. By Corollary 5.8, it is enough to show that for any $Y \subseteq C$, $\rho(\dot{\bigwedge} Y) = \bigwedge \rho(Y)$. We distinguish the two branches of the definition of $\dot{\bigwedge}$.

(i) If $\dot{\bigwedge} Y = \bigwedge \{y \in Y \mid \rho(y) = \rho(x)\}$ for a suitable $x \in Y$ such that $\rho(x) = \bigwedge \rho(Y)$, then $\rho(\dot{\bigwedge} Y) = \rho(x) = \bigwedge \rho(Y)$, because ρ is meet-uniform (on $C_{\leqslant}$).

(ii) Otherwise, $\dot{\bigwedge} Y = \bigwedge \rho(Y)$, and hence, $\rho(\dot{\bigwedge} Y) = \rho(\bigwedge \rho(Y)) = \bigwedge \rho(Y)$. ▮

6. GENERALIZED ORDER-THEORETIC SEMANTICS

In this section, we show how to exploit the general order-theoretic results of the previous sections in order to define an order-theoretic semantics generalizing the model-theoretic logic program semantics proposed by Falaschi *et al.* [1993].

6.1. Interpretations as Models

The s-semantics for logic programs has been defined by Falaschi *et al.* [1989] with the aim of providing a declarative semantic counterpart to the operational observable property given by computed answer substitutions, since this was not possible by means of the standard ground success-set semantics of van Emden and Kowalski [1976]. Let us introduce the following notation: Π denotes the set of predicate symbols of the underlying first-order language $\mathscr{L}$, and $\bar{X}$ denotes a sequence of distinct variables; moreover, if P is a program and $G = b_1, ..., b_n$ $(n \geqslant 0)$ is a goal, then we write $G \xrightarrow{\theta}_P \square$ iff there exists a SLD-refutation of G in P with computed answer substitution θ. Then, the s-semantics $\mathscr{S}(P) \subseteq Atom$ of a program P is defined as

$$\mathscr{S}(P) = \left\{ p(\bar{X})\,\theta \;\middle|\; p \in \prod, \theta \in Sub, p(\bar{X}) \xrightarrow{\theta}_P \square \right\}.$$

It turns out that this semantics is fully abstract with respect to the notion of computed answer substitution (cf. [Falaschi *et al.* 1989]). Let P_1 and P_2 be programs, and define P_1 and P_2 to be equivalent for the computed answer substitutions observable when, for any goal G and substitutions θ and σ, $p(\bar{X}) \xrightarrow{\theta}_{P_1} \square$ iff $p(\bar{X}) \xrightarrow{\sigma}_{P_2} \square$ and $G\theta \sim G\sigma$. Then, the full abstraction theorem states that $\mathscr{S}(P_1) = \mathscr{S}(P_2)$ if and only if P_1 and P_2 are equivalent in that sense. For all the details, properties and consequences of the s-semantics the reader is referred to [Falaschi *et al.* 1989], since here we recalled its definition for illustrative purposes only.

The s-semantics of a program is therefore defined as a subset of the nonground Herbrand base, namely it is an interpretation. Falaschi *et al.* [1993] gave a model-theoretic interpretation for sets of nonground atoms, with the obvious goal of making the s-semantics a model as well as any standard ground Herbrand model. Let us recall this notion. Given a standard ground Herbrand interpretation $I \in \wp(Atom_{\varnothing})$ and a definite clause c, the notion of truth of c in I is the standard logical one, i.e., $I \models c$. On the other hand, given any nonground interpretation $I \in \wp(Atom)$, from a model-theoretic viewpoint, the idea is that I is a denotation for the set of its ground instances $gr(I)$. Accordingly, Falaschi *et al.* [1993] proposed the following approach: an interpretation $I \in \wp(Atom)$ is a *model* of a program P if all the clauses of P are true (in the standard logical sense) in $gr(I)$. By this definition, any standard Herbrand model is also a model in the above sense, and therefore this implies that any program has a model. Moreover, it turns out that, for any program P, the s-semantics $\mathscr{S}(P)$ is a model of P, and for any ground Herbrand interpretation $I \in \wp(Atom_{\varnothing})$, this new notion of being a model is equivalent to the standard old one.

It is well-known [van Emden and Kowalski 1976] that, for any program P, its set $\mathscr{M}_P$ of standard Herbrand models is closed by set intersection, and therefore the least Herbrand model exists. Also, the ground Herbrand base $Atom_{\varnothing}$ is a model, i.e., $Atom_{\varnothing} \in \mathscr{M}_P$. This implies that $\mathscr{M}_P$ is a Moore-family of $\wp(Atom_{\varnothing})$, i.e., $\mathscr{M}_P$ can be thought of as a closure on $\wp(Atom_{\varnothing})$ such that for any $I \in \wp(Atom_{\varnothing})$, I is a Herbrand model of P iff I is a fixpoint of $\mathscr{M}_P$. This observation was first reported by Lassez and Maher [1984]. As a side observation, we note that, in general, this closure operator is not meet-uniform. In fact, consider $P = \{p \leftarrow q, q \leftarrow p\}$, and $I = \{p\}$, $J = \{q\} \in \wp(Atom_{\varnothing})$; it is clear, that $\mathscr{M}_P(I) = \mathscr{M}_P(J) = \{p, q\}$, whilst $I \cap J = \varnothing \in \mathscr{M}_P$.

Order-theoretic generalization. In our order-theoretic approach, the above notion of model is formalized naturally as follows. Recall that in the generalized hierarchy of Section 3, the set of ground Herbrand interpretations corresponds to the complete lattice $\langle g(\rho(C)), \leqslant \rangle$, where any generalized interpretation $x \in C$ is "grounded" to $g(\rho(x))$. Thus, given any program P,[2] its set of ground Herbrand models is generalized by a closure operator $m_P \in uco(g(\rho(C)))$. Therefore, an interpretation $x \in C$ is a (*generalized*) *model* of P whenever $g(\rho(x))$ is a fixpoint of m_P, i.e., when $g(\rho(x))$ is a generalized Herbrand model of P. The set $\mathscr{G}_P$ of (generalized) models of P is then defined as: $\mathscr{G}_P = \{x \in C \mid m_P(g(\rho(x))) = g(\rho(x))\}$.

6.2. *Generalizing the Model-Theoretic Semantics*

In contrast to standard Herbrand models, Falaschi *et al.* [1993] observe, by a simple counterexample, that the model intersection property does not hold any

[2] It is worth noting that in our order-theoretic reconstruction, we do not need at all to refer to 'real' logic programs. Thus, we can think of the set of all programs as a mere set of objects acting as indices.

longer for the notion of nonground model recalled in Section 6.1. This implies that the least model of a program with respect to set inclusion, in general, does not exist. Of course, in our order-theoretic construction, this counterexample shows that, in general, the set of generalized models $\mathscr{G}_P$ is not closed by *glb* (of C). Falaschi *et al.* [1993] point out that this phenomenon "can easily be explained by noting that set inclusion does not adequately reflect the property of nonground atoms of being representatives of all their ground instances." Then, they propose a new partial order on $\wp(Atom)$, as given in the following definition, which allows one to restore the desired model intersection property.

Definition 6.1. ([Falaschi *et al.* 1993, Definition 4.1]). Let $I, J \in \wp(Atom)$. Then,

(i) $I \vdash J \;\Leftrightarrow\; \forall A \in I.\exists B \in J.B \preccurlyeq A$;

(ii) $I \trianglelefteq J \;\Leftrightarrow\; (I \vdash J)\ \&\ (J \vdash I \Rightarrow I \subseteq J)$.

As Falaschi *et al.* note, clearly, $I \vdash J$ iff $\ulcorner I \urcorner \subseteq \ulcorner J \urcorner$, and therefore

$$I \trianglelefteq J \quad \Leftrightarrow \quad (\ulcorner I \urcorner \subseteq \ulcorner J \urcorner) \quad \& \quad (\ulcorner J \urcorner \subseteq \ulcorner I \urcorner \Rightarrow I \subseteq J). \tag{$*$}$$

By this formulation, this definition gets a clearer meaning. In fact, it says that in order to compare two interpretations I and J we have to look at their respective abstractions in $\wp^{\downarrow}(Atom)$: I is smaller than J if $\ulcorner I \urcorner \subset \ulcorner J \urcorner$, and, whenever $\ulcorner I \urcorner = \ulcorner J \urcorner$, the original subset-ordering, $I \subseteq J$, is considered.

Falaschi *et al.* [1993] present a number of important consequences of Definition 6.1. They show that $\langle \wp(Atom), \trianglelefteq \rangle$ is a complete lattice, and provide an explicit characterization for the corresponding *lub*. Further, they prove that for any program P, the *glb* w.r.t. $\trianglelefteq$ of its set of nonground models is still a model, thus obtaining the $\trianglelefteq$-least nonground model for P, which is also shown to coincide with the standard least Herbrand model. It is important to remark that in order to prove all these results, specific logic programming concepts and tools are heavily used, like clauses, substitutions, instances, and Herbrand models. In contrast, our generalized approach involves pure order-theoretic notions only and no specific logic programming concept.

Order-theoretic generalization. By the characterization $(*)$ above, it is immediate to observe that the ordering given by Definition 6.1 is an instance of the definition of lifted partial order induced by the meet-uniform closure $\ulcorner \cdot \urcorner$ on the domain of nonground interpretations $\wp(Atom)$. By the results of Section 4, in our generalized hierarchy, the hypothesis that the first step of abstraction of C is given by a meet-uniform closure ρ, just allows us to lift the complete order $\leqslant$ of C via ρ, then obtaining the lifted complete lattice $\langle C, \leqslant^{\rho} \rangle$. Hence, this is precisely the order-theoretic generalization of the corresponding result of Falaschi *et al.* [1993, Theorem 4.9]. Also, we noted that, in general, the *glb* (in C) of a set of models of P, i.e., of a subset of $\mathscr{G}_P = \{x \in C \mid m_P(g(\rho(x))) = g(\rho(x))\}$, is not a model of P. Instead, the next theorem shows that the meet-uniformity of the closure ρ is, once again, the crucial property that permits to recover the generalized model-intersection property w.r.t. the lifted complete order $\leqslant^{\rho}$.

THEOREM 6.2. *If $\rho \in uco^{*}(C_{\leqslant})$ then $\mathcal{G}_P$ is a Moore-family of $\langle C, \leqslant^{\rho} \rangle$.*

Proof. Let $\{x_i\}_{i \in I} \subseteq \mathcal{G}_P$. The following equalities hold:

$$\begin{aligned}
m_P\left(g\left(\rho\left(\dot{\bigwedge_{i\in I}} x_i\right)\right)\right) &= (\text{by Theorem 5.10})\\
m_P\left(g\left(\dot{\bigwedge_{i\in I}} \rho(x_i)\right)\right) &= (\text{by Corollary 5.8})\\
m_P\left(g\left(\bigwedge_{i\in I} \rho(x_i)\right)\right) &= (\text{by } \leqslant\text{-co-additivity of } g)\\
m_P\left(\bigwedge_{i\in I} g(\rho(x_i))\right) &= (\text{since } \forall i \in I. x_i \in \mathcal{G}_P)\\
m_P\left(\bigwedge_{i\in I} m_P(g(\rho(x_i)))\right) &= (\text{since } m_P \in uco(g(\rho(C))_{\leqslant}) \text{ and by (i) in Section 2.2})\\
\bigwedge_{i\in I} m_P(g(\rho(x_i))) &= (\text{since } \forall i \in I. x_i \in \mathcal{G}_P)\\
\bigwedge_{i\in I} g(\rho(x_i)) &= (\text{by } \leqslant\text{-co-additivity of } g)\\
g\left(\bigwedge_{i\in I} \rho(x_i)\right) &= (\text{by Corollary 5.8})\\
g\left(\dot{\bigwedge_{i\in I}} \rho(x_i)\right) &= (\text{by Theorem 5.10})\\
g\left(\rho\left(\dot{\bigwedge_{i\in I}} x_i\right)\right).&
\end{aligned}$$

This shows that $\dot{\bigwedge}_{i\in I} x_i \in \mathcal{G}_P$, and therefore closes the proof. ▮

As a consequence, for any program P, $\mathcal{G}_P$ is a complete lattice w.r.t. $\leqslant^{\rho}$—this generalizes [Falaschi *et al.* 1993, Corollary 4.11]—and the $\leqslant^{\rho}$-least generalized model is therefore $\dot{\bigwedge} \mathcal{G}_P$. The overall scenario is summarized by Fig. 7.

By assuming further hypotheses, we are also able to give the generalization in our framework of the fact proved by Falaschi *et al.* [1993, Theorem 4.15] that the $\trianglelefteq$-least model coincides with the $\subseteq$-least Herbrand model. Recall that in our generalized approach, $g(\rho(C))$ stands for the subset of C of ground Herbrand interpretations, while, for any program P, $m_P \in uco(\langle g(\rho(C)), \leqslant \rangle)$ is the closure representing the Herbrand models of P. Thus, the set $\mathcal{H}_P$ of generalized Herbrand models of a program P is just given by $\mathcal{H}_P = \{z \in g(\rho(C)) \mid m_P(z) = z\}$. Hence, $\bigwedge \mathcal{H}_P$ is the order-theoretic counterpart of the $\subseteq$-least Herbrand model, and $\dot{\bigwedge} \mathcal{G}_P = \bigwedge \mathcal{H}_P$ states the equality of the two generalized least models. Analogously to the logic programming side, it is straightforward to note that, for any program P,

$$\begin{tikzcd}
\langle C,\leq,\vee,\wedge,\top_C,\bot_C\rangle \arrow[d, "\rho\in uco^*(\langle C,\leq\rangle)"] \arrow[rr, dotted, "\text{lifting } \leq \text{ via } \rho"] & & \langle C,\leq^\rho,\dot\vee,\dot\wedge,\top_C,\bot_C\rangle \\
\langle \rho(C),\leq,\vee_{\rho(C)},\wedge,\top_C,\rho(\bot_C)\rangle \arrow[d, "g:\rho(C)\to\rho(C) \text{ co-additive}"] & & \\
\langle g(\rho(C)),\leq,\vee_{g(\rho(C))},\wedge,g(\top_C),g(\rho(\bot_C))\rangle \arrow[d, Rightarrow, "m_P\in uco(\langle g(\rho(C)),\leq\rangle)"] & & \\
\mathscr{G}_P=\{x\in C \mid m_P(g(\rho(x)))=g(\rho(x))\} \arrow[r, squiggly] \arrow[uuurr, dotted, "\mathscr{G}_P\in uco(\langle C,\leq^\rho\rangle)"'] & \text{in general, } \mathscr{G}_P\notin uco(\langle C,\leq\rangle) &
\end{tikzcd}$$

FIG. 7. The overall generalized order-theoretic approach

by grounding its set of generalized models, one gets its set of generalized Herbrand models, i.e., $\mathscr{H}_P=\{g(\rho(x))\in g(\rho(C)) \mid x\in\mathscr{G}_P\}$. The further hypotheses required by the next theorem correspond, on the logic programming side, to the observations that the grounding operator $gr\colon \wp^{\downarrow}(Atom)\to\wp^{\downarrow}(Atom)$ is indeed a (co-additive) lower closure operator, and that for any $I\in\wp(Atom)$ and $J\in\wp^{\downarrow}(Atom)$, whenever J indeed is ground (i.e., $gr(J)=J$) and $I\subseteq J$, then I is ground as well.

Theorem 6.3. *If $g\in lco(\rho(C)_{\leqslant})$, and, for any $x\in C$ and $y\in\rho(C)$, $x\leqslant g(y)=y$ implies $x\in\rho(C)$ and $g(x)=x$, then, for any program P, $\dot\bigwedge\mathscr{G}_P=\bigwedge\mathscr{H}_P$.*

Proof. First, note that $\mathscr{H}_P=g(\rho(\mathscr{G}_P))$. Also, the following equalities hold:

$$\bigwedge\mathscr{H}_P=(\text{since } \mathscr{H}_P=g(\rho(\mathscr{G}_P)))$$

$$\bigwedge g(\rho(\mathscr{G}_P))=\left(\text{by } \bigwedge\text{-co-additivity of } g\right)$$

$$g\left(\bigwedge\rho(\mathscr{G}_P)\right)=(\text{by Corollary 5.8})$$

$$g\left(\dot\bigwedge\rho(\mathscr{G}_P)\right)=(\text{by Theorem 5.10})$$

$$g\left(\rho\left(\dot\bigwedge\mathscr{G}_P\right)\right)=(\text{by Theorem 6.2}) \tag{1}$$

$$m_P\left(g\left(\rho\left(\dot\bigwedge\mathscr{G}_P\right)\right)\right). \tag{2}$$

Moreover, by the equality (1), $g(\rho(\bigwedge\mathscr{H}_P))=g(\rho(g(\rho(\dot\bigwedge\mathscr{G}_P))))$, and, from $g(\rho(C))\subseteq\rho(C)$ and by idempotency of g and ρ, we get

$$g\left(\rho\left(\bigwedge\mathscr{H}_P\right)\right)=g\left(\rho\left(\dot\bigwedge\mathscr{G}_P\right)\right)=(\text{by (1)})=\bigwedge\mathscr{H}_P. \tag{3}$$

($\dot{\bigwedge}\, \mathscr{G}_P \leqslant^{\rho} \bigwedge \mathscr{H}_P$) The following equalities show that $\bigwedge \mathscr{H}_P \in \mathscr{G}_P$:

$$\begin{aligned}
m_P\left(g\left(\rho\left(\bigwedge \mathscr{H}_P\right)\right)\right) &= (\text{by } (1)) \\
m_P\left(g\left(\rho\left(g\left(\rho\left(\dot{\bigwedge}\, \mathscr{G}_P\right)\right)\right)\right)\right) &= (\text{by } g(\rho(C)) \subseteq \rho(C) \\
&\qquad \text{and indempotency of } g \text{ and } \rho) \\
m_P\left(g\left(\rho\left(\dot{\bigwedge}\, \mathscr{G}_P\right)\right)\right) &= (\text{by } (2)) \\
\bigwedge \mathscr{H}_P &= (\text{by } (3)) \\
g\left(\rho\left(\bigwedge \mathscr{H}_P\right)\right).
\end{aligned}$$

($\bigwedge \mathscr{H}_P \leqslant^{\rho} \dot{\bigwedge}\, \mathscr{G}_P$) We show that if $x \in \mathscr{G}_P$ then $\bigwedge \mathscr{H}_P \leqslant^{\rho} x$.

(i) From $x \in \mathscr{G}_P$, we get $\dot{\bigwedge}\, \mathscr{G}_P \leqslant^{\rho} x$, from which, $\rho(\dot{\bigwedge}\, \mathscr{G}_P) \leqslant \rho(x)$, and, in turn, $g(\rho(\dot{\bigwedge}\, \mathscr{G}_P)) \leqslant g(\rho(x))$. By the equality (1) above, $\bigwedge \mathscr{H}_P = g(\rho(\dot{\bigwedge}\, \mathscr{G}_P))$, and since $\rho(\bigwedge \mathscr{H}_P) = \bigwedge \mathscr{H}_P$, we get, by reductivity of g, $\rho(\bigwedge \mathscr{H}_P) \leqslant g(\rho(x)) \leqslant \rho(x)$, as desired.

(ii) We show that $\rho(x) = \rho(\bigwedge \mathscr{H}_P) \Rightarrow \bigwedge \mathscr{H}_P \leqslant x$. From $\rho(x) = \rho(\bigwedge \mathscr{H}_P) = \bigwedge \mathscr{H}_P$, we get $g(\rho(x)) = g(\bigwedge \mathscr{H}_P) =$ (since g is a lower closure) $= \bigwedge \mathscr{H}_P = \rho(x)$. Thus, $x \leqslant \rho(x) = g(\rho(x))$, and by the hypotheses of the theorem, we obtain $g(x) = x$. Hence, $\rho(x) = \rho(g(x)) =$ (since $g(\rho(C)) \subseteq \rho(C)$) $= g(x)$, from which, $x = \rho(x) = \bigwedge \mathscr{H}_P$. ▮

It is worth noting that it is possible to slightly generalize our hierarchy, and in particular Theorem 6.2. In fact, after the first step of abstraction given by a meet-uniform closure $\rho \in uco^*(C_{\leqslant})$, we can consider an arbitrary finite number of further abstractions given by co-additive functions. This is possible since the composition of co-additive functions is clearly still co-additive. Also, note that by considering the mapping g as the identity, we can deal only with the unique abstraction given by ρ.

7. UNIFORM CLOSURES AND ABSTRACT DOMAIN REFINEMENTS

In this section, we show how the novel order-theoretic notion of uniformity finds relevant applications also in abstract interpretation theory, specifically in the area of refinement operators of abstract domains (as far as the precision is concerned).

7.1. Abstract Domain Refinements

In abstract interpretation, a domain refinement is intended as an operator which takes as input a given abstract domain (that is, a closure) and returns as output an enhanced domain, i.e., a more precise domain, by systematically adding new information. A formal treatment of abstract domain refinements has been recently put forward by Filé *et al.* [1996] and successively sharpened by Giacobazzi and Ranzato [1997], generalizing most of the well-known domain refinements, like reduced product and disjunctive completion [Cousot and Cousot 1979]. In the following, we assume that $\langle C, \leqslant \rangle$ is the complete lattice that plays the role of the concrete domain of reference. As recalled in Section 2.3, the lattice of abstract interpretations of a given concrete domain C is isomorphic to the complete lattice $uco(C)$ of all upper closure operators on C. As recalled in the Introduction, an abstract domain *refinement* is defined as an operator $\mathfrak{R}: uco(C) \to uco(C)$ on the lattice $uco(C)$ of abstractions of C, which is monotone and reductive (i.e., $\mathfrak{R}(A) \sqsubseteq A$). Idempotency is an additional reasonable requirement, ensuring that the refinement of an abstract domain is performed all at once. Thus, a *refinement is a lower closure operator* on the complete lattice $uco(C)$, i.e., any mapping in $lco(uco(C))$, and therefore, some properties of domain refinements can be derived by duality from those of abstract domains, which are upper closure operators. Among them, (i) the image of $\mathfrak{R}$ coincides with the set of *refined abstract domains*: $\mathfrak{R}(uco(C)) = \{A \in uco(C) \mid \mathfrak{R}(A) = A\}$, and (ii) the set $\langle lco(uco(C)), \sqsubseteq \rangle$ of all the refinements is a complete lattice (by a slight abuse of notation, we always use the symbol $\sqsubseteq$ for any order between closures), where $\mathfrak{R}_1 \sqsubseteq \mathfrak{R}_2$ iff for any $A \in uco(C)$, $\mathfrak{R}_1(A) \sqsubseteq_{uco(C)} \mathfrak{R}_2(A)$ iff the set of domains refined by $\mathfrak{R}_1$ is contained in the set of those refined by $\mathfrak{R}_2$. Thus, analogously to the case of abstract domains, the complete ordering $\sqsubseteq$ between refinements can be interpreted as a relation of precision, where $\mathfrak{R}_1$ is more precise than $\mathfrak{R}_2$ iff $\mathfrak{R}_1 \sqsubseteq \mathfrak{R}_2$. For more details and properties on abstract domain refinements, we refer to [Giacobazzi and Ranzato 1997].

EXAMPLE 7.1. Reduced product[3] is the simplest and probably most familiar example of abstract domain refinement. It has been successfully applied in many works on program analysis, for instance in [Codish *et al.* 1995; Granger 1988; Muthukumar and Hermenegildo 1991; Sundararajan and Conery 1992]. As recalled in Section 2.3, the reduced product corresponds to the *glb* in the complete lattice of (upper) closure operators. The terminology reduced product of abstract domains stems from the fact that it can be represented by means of the standard cartesian product, where equivalent tuples of objects are identified, i.e., reduced (more details are given, e.g., in [Cousot and Cousot 1992a]). For example, consider the abstract domains A^- and A^+ in Fig. 8, which are abstractions of the complete lattice $\langle \wp(\mathbb{Z}), \subseteq \rangle$ (with the most obvious meaning of their elements), and are typically used for sign analysis of integer variables. It is easily seen that, up to isomorphic representation of domain's objects, the reduced product $A^- \sqcap A^+$ is the abstract domain *Sign* of Fig. 1. Observe that *Sign* has two new elements with respect to A^- and A^+, i.e., 0 and $\varnothing$, which are obtained by combining, by set intersection in $\wp(\mathbb{Z})$, the corresponding sets of integers, respectively, -0 with $0+$, and $-$ with $+$. Here, reduction is necessary only for identifying distinct pairs of elements denoting the empty set of integers: the pairs $\langle -, + \rangle$, $\langle -0, + \rangle$, and $\langle -, 0+ \rangle$ all denote $\varnothing$. By fixing one of the arguments of reduced product, one

[3] For simplicity of notation, we consider the reduced product as a binary operator.

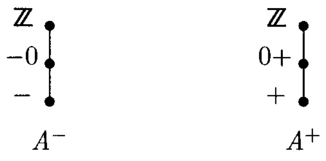


FIG. 8. The abstract domains A^- and A^+.

obviously gets a refinement. Let C be the concrete domain and $A \in uco(C)$ be any fixed abstract domain. Then, the reduced product refinement with respect to A is the lower closure operator $\mathfrak{R}_{\sqcap A} = \lambda X.(A \sqcap X) \in lco(uco(C))$.

7.2. Inverting Refinements

Filé *et al.* [1996] motivated and introduced the notion of inverse of an abstract domain refinement. For a given refinement $\mathfrak{R} \in lco(uco(C))$ and an abstract domain $A \in uco(C)$, the *optimal basis* of A for $\mathfrak{R}$, when it exists, is a domain $D \in uco(C)$ such that $\mathfrak{R}(D) = \mathfrak{R}(A)$, and for any $B \in uco(C)$, $\mathfrak{R}(B) = \mathfrak{R}(A)$ implies $B \sqsubseteq D$. Clearly, if an optimal basis D exists, then this domain is unique. Thus, we will refer to the existence of *the* optimal basis. In other terms, whenever the optimal basis of A for $\mathfrak{R}$ exists, this is the most abstract domain having the same refinement (for $\mathfrak{R}$) as A. Given a class $\mathbb{K} \subseteq uco(C)$ of abstract domains, if any $A \in \mathbb{K}$ admits the optimal basis, the mapping $\mathfrak{R}^-: \mathbb{K} \to uco(C)$ providing the optimal basis is called the *inverse* of $\mathfrak{R}$ on $\mathbb{K}$. Thus, the inverse $\mathfrak{R}^-$ does exist on $\mathbb{K}$ iff there exists the optimal basis for $\mathfrak{R}$ of any domain in $\mathbb{K}$. The intuition is that the refined domain $\mathfrak{R}(A)$ can be systematically reconstructed by applying the refinement $\mathfrak{R}$ to the more abstract, and therefore simpler, domain $\mathfrak{R}^-(A)$. Therefore, roughly speaking, $\mathfrak{R}^-$ is the operator providing the simplest abstract domains of input for the refinement $\mathfrak{R}$.

The notion of abstract domain refinement can be slightly generalized, by allowing the possibility of having as domain of definition of a refinement any subset $\mathbb{Q}$ of $uco(C)$ (see [Giacobazzi and Ranzato 1997] for all the details). In this case, the inverse of a refinement $\mathfrak{R}: \mathbb{Q} \to uco(C)$ could exist for some subset $\mathbb{K}$ of $\mathbb{Q}$ and it should take values over $\mathbb{Q}$, i.e., $\mathfrak{R}^-: \mathbb{K} \to \mathbb{Q}$. An example of this kind of partial refinement is provided by the *negative completion* refinement (cf. Filé *et al.* 1996; Giacobazzi and Ranzato 1997]). Let the concrete domain C be a complete Boolean algebra. The negative completion $\mathfrak{R}_{\neg}$ is then defined on the sublattice $uco^a(C)$ of $uco(C)$ of additive closures (also called disjunctive abstract domains), and it upgrades a given abstract domain by adding denotations for the lattice-theoretic complements of its elements. It is easy to verify that if $A \in uco^a(C)$ then $\neg A = \{\neg a \in C \mid a \in A\} \in uco^a(C)$. Thus, $\mathfrak{R}_{\neg}$ lifts a given disjunctive abstract domain A to the most abstract domain containing both A and $\neg A$, i.e., $\mathfrak{R}_{\neg}: uco^a(C) \to uco(C)$ maps A to the reduced product $\mathfrak{R}_{\neg}(A) = A \sqcap \neg A$. Although, given $A \in uco^a(C)$, $\mathfrak{R}_{\neg}(A)$, in general, does not belong to $uco^a(C)$ (cf. [Giacobazzi and Ranzato 1997]), it turns out that $\mathfrak{R}_{\neg}$ is monotone, reductive and

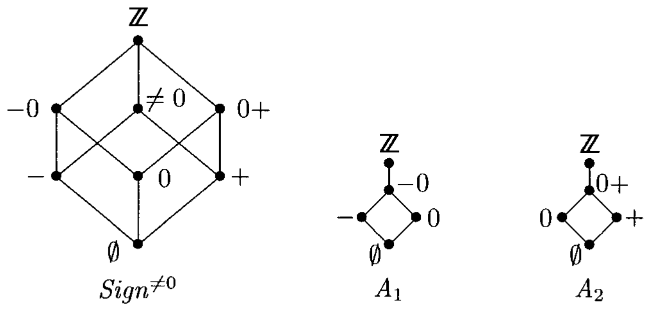


FIG. 9. The abstract domains $Sign^{\neq 0}$, A_1 and A_2.

idempotent, and therefore is a refinement in the generalized framework of [Giacobazzi and Ranzato 1997]. In general, optimal bases do not always exist, and the negative completion provides a meaningful example of a refinement which is not invertible on any really significant class of abstract domains, as first noted by Filé *et al.* [1996]. Let us consider the disjunctive abstract domains $Sign^{\neq 0}$, A_1 and A_2 depicted in Fig. 9, for sign analysis of integer variables, with their obvious meanings as additive closures on $\langle \wp(\mathbb{Z}), \subseteq \rangle$ (which obviously is a complete Boolean algebra). It turns out that the the optimal basis of $Sign^{\neq 0}$ for $\mathfrak{R}_\neg$ does not exist. In fact, $\mathfrak{R}_\neg(A_1) = \mathfrak{R}_\neg(A_2) = \mathfrak{R}(Sign^{\neq 0}) = Sign^{\neq 0}$, while for the least common abstraction $A_1 \sqcup A_2 = \{\mathbb{Z}, 0, \emptyset\}$, $\mathfrak{R}_\neg(A_1 \sqcup A_2) = \{\mathbb{Z}, 0, \neq 0, \emptyset\}$, and hence this implies that $Sign^{\neq 0}$ does not admit the optimal basis for $\mathfrak{R}_\neg$. Since in this example the concrete domain is a complete Boolean algebra, and $Sign^{\neq 0}$ enjoys all most important lattice-theoretic properties, this means that $\mathfrak{R}_\neg$ is not invertible on any significant class of abstract domains.

7.3. Invertibility as Join-Uniformity

As the attentive reader might have already guessed, it turns out that join-uniformity captures exactly the concept of invertible domain refinement. Recalling by duality from Section 4 the notion of K-join-uniformity, one can state the following immediate result, whose prime importance stems from the striking connection between two notions that came out from very different questions.

THEOREM 7.2. *Given $\mathbb{K} \subseteq uco(C)$, a refinement $\mathfrak{R}: uco(C) \to uco(C)$ is invertible on $\mathbb{K}$ iff $\mathfrak{R}$ is join-uniform on $\mathbb{K}$.*

Moreover, given a refinement $\mathfrak{R}: uco(C) \to uco(C)$ which is invertible on $\mathbb{K}$, the canonical representative of Definition 4.4 of an abstract domain $A \in \mathbb{K}$ for $\mathfrak{R}$ is $\nabla_{\mathfrak{R}}(A) = \bigsqcup \{D \in uco(C) \mid \mathfrak{R}(D) = \mathfrak{R}(A)\}$, and therefore it is exactly the optimal basis of A, i.e., $\nabla_{\mathfrak{R}}(A) = \mathfrak{R}^-(A)$. In other words, the canonical representative operator $\nabla_{\mathfrak{R}}$ is the inverse of $\mathfrak{R}$. Hence, following the notation of Section 4, we have that $lco^*_{\mathbb{K}}(uco(C))$ denotes the set of abstract domain refinements invertible on $\mathbb{K} \subseteq uco(C)$. By duality from the observations and results of Section 4, we can then derive the following properties of invertible refinements:

(i) Invertible refinements on some $\mathbb{K}$ give rise to a complete lattice $\langle lco^*_{\mathbb{K}}(uco(C)), \sqsubseteq \rangle$, which, in particular, is a complete join subsemilattice, i.e., a dual-Moore-family, of $lco(uco(C))$, with top and bottom elements $\lambda X.X$ (the identity refinement) and $\lambda X.C$ (the full refinement), respectively;

(ii) The lattice of invertible refinements is not dual-atomic;

(iii) Invertibility is not preserved by composition.

By property (i), the space of refinements that are invertible on some fixed $\mathbb{K}$ is itself the set of fixpoints of a lower closure operator $\psi^* \in lco(lco(uco(C)))$, defined as follows: for any $\mathfrak{R} \in lco(uco(C))$, $\psi^*(\mathfrak{R}) = \bigsqcup \{\mathfrak{I} \in lco(uco(C)) \mid \mathfrak{I} \sqsubseteq \mathfrak{R}$, $\mathfrak{I}$ is invertible on $\mathbb{K}\}$. This operator transforms any, possibly noninvertible on $\mathbb{K}$, refinement $\mathfrak{R} \in lco(uco(C))$, into the weakest refinement (with respect to the order $\sqsubseteq$ between refinements explained in Section 7.1) which is both invertible on $\mathbb{K}$ and more precise than $\mathfrak{R}$.

Let us now see an important example of an invertible abstract domain refinement.

EXAMPLE 7.3. *Disjunctive completion* was originally introduced by Cousot and Cousot [1979] to prove that merge-over-all-paths data-flow analyses can be always expressed in least fixpoint form. It has been then considered in Nielson's [1984] approach to abstract interpretation using domain theory, and applied in program analysis, e.g., in Cousot and Cousot's [1994] comportment analysis, in Deutsch's [1992] alias analysis, in Jensen's [1997] disjunctive strictness logic, and in analysis of logic program groundness-dependencies [Filé and Ranzato 1998].

The disjunctive completion enhances an abstract domain as little as possible so that it becomes disjunctive, i.e., an additive closure. It should be evident that disjunctive abstract domains are practically very useful, since their *lub*'s are as precise as possible. Following the general approach of Giacobazzi and Ranzato [1998], given a concrete domain C, the disjunctive completion $\mathfrak{R}_{\vee} : uco(C) \to uco(C)$ is defined as $\mathfrak{R}_{\vee}(A) = \bigsqcup \{D \in uco^a(C) \mid D \sqsubseteq A\}$, for any $A \in uco(C)$. It is immediate to observe that $\mathfrak{R}_{\vee}$ actually is a lower closure, and hence correctly defines an abstract domain refinement. Whenever the concrete domain is completely distributive, the disjunctive completion of an abstract domain can be characterized by various equivalent powerset constructions [Cousot and Cousot 1994; Filé and Ranzato 1998]. As a simple example, if *Sign* is the abstraction of $\langle \wp(\mathbb{Z}), \subseteq \rangle$ of Fig. 1, its disjunctive completion $\mathfrak{R}_{\cup}(Sign)$ is the domain $Sign^{\neq 0}$ considered above and depicted in Fig. 9, which contains the concrete *lub* (i.e., set union) of any subset of *Sign*, and in particular it contains a new object $\neq 0 = - \bigcup 0$, denoting nonzero integers. The inverse of disjunctive completion should therefore be an operation which returns (when possible) the most abstract domain whose disjunctive completion is a given disjunctive abstract domain. Giacobazzi and Ranzato [1998] introduced the notion of least disjunctive basis for an abstract domain, that turns out to be an instance of the concept of optimal basis, and hence allows us to define the inverse of the disjunctive completion refinement. Giacobazzi and Ranzato [1998, Theorem 4.10 and 4.11] give two results of existence of the least disjunctive basis, which can be read in the terminology of this paper as follows:

(i) If C is a dual-algebraic completely distributive lattice then $\mathfrak{R}_{\vee}$ is invertible on all $uco(C)$;

(ii) If C is distributive then $\mathfrak{R}_\vee$ is invertible on $\{A \in uco(C) \mid |A| < \aleph_0\}$.

It is not hard to verify that the optimal basis $\mathfrak{R}^-_\cup(Sign)$ of *Sign* is the abstract domain $A^{\pm 0}$ of Fig. 1.

From what has been discussed above, it would be natural to think that the inverse of an abstract domain refinement can be formalized as an operator of *simplification*, somehow dual to a refinement. Intuitively, an abstract domain simplification should be an operator that for a given domain of input, returns a more abstract domain, that is, it should be an extensive operator on the lattice of abstract interpretations. Moreover, as well as for refinements, a simplification operator should monotonically transform abstract domains, and perform simplifications all at once. In this sense, while refinements are lower closures, their inverses should generally be upper closures. Instead, we will see that when a given refinement $\mathfrak{R} \in lco(uco(C))$, possibly under some hypotheses on C, is invertible on all $uco(C)$, the inverse operator $\mathfrak{R}^-$ is not necessarily an upper closure. More in general, when $\mathfrak{R}$ is invertible on $\mathbb{K} \subseteq uco(C)$, the inverse $\mathfrak{R}^-: \mathbb{K} \to uco(C)$ is not necessarily monotone. In fact, we can only prove the following general result for join-uniform functions, where f^ι is just the dual canonical representative operator of Definition 4.4.

PROPOSITION 7.4. *Let L be a complete lattice and $f: L \to L$ be monotone and join-uniform on $K \subseteq L$. Then, $f^\iota: K \to L$ defined as $f^\iota(x) = \bigvee \{y \in L \mid f(y) = f(x)\}$ is extensive and idempotent* (*i.e., if $f^\iota(x) \in K$ then $f^\iota(f^\iota(x)) = f^\iota(x)$*).

Proof. Extensivity is obvious by definition. Moreover, by join-uniformity, we have that for any $x \in K$ such that $f^\iota(x) \in K$, $f^\iota(f^\iota(x)) = \bigvee \{y \in L \mid f(y) = f(f^\iota(x))\} = \bigvee \{y \in L \mid f(y) = f(x)\} = f^\iota(x)$, proving idempotency. ▌

In general, monotonicity of an inverse operator may fail. This is the case of the disjunctive completion, as shown by the following example taken from [Giacobazzi and Ranzato 1998, Example 5.3].

EXAMPLE 7.5. Consider the abstract domains *Sign*, A^+ and $A^\pm$, all already introduced above. Observe that $\mathfrak{R}^-_\cup(A^+) = A^+$, while $\mathfrak{R}^-_\cup(Sign) = A^{\pm 0}$. This shows that the inverse operator of the disjunctive completion refinement is neither monotone nor antimonotone, since A^+ and $A^{\pm 0}$ are incomparable abstractions of $\wp(\mathbb{Z})$.

By Proposition 7.4, given a lower closure $\eta \in lco^*(L)$ which is join-uniform (on all L), and its associated generalized inverse $\eta^\iota: L \to L$, we have that, for any $x \in L$, $\eta(\eta^\iota(x)) = \eta(x) \leqslant x$ and $\eta^\iota(\eta(x)) = \eta^\iota(x) \geqslant x$, but the pair η and η^ι does not constitute in general a Galois connection, due to the lack of monotonicity of η^ι. Clearly, if η^ι is monotone, then it is the right-adjoint of η. In this sense, our notion of inverting a refinement does not correspond to the inversion as right-adjoint of the refinement. On the other hand, it is well known how pervasive the notion of adjunction is in theoretical computer science, and very often the existence of an adjunction is considered as a weak form of inversion (see, e.g., the paradigmatic case of the weak inverse in Hoare's logic, cf. [Hoare *et al.* 1987]). Thus, in the following, we will focus on the inversion of a refinement in the sense of adjunctions.

Proposition 7.6. *Let L be a complete lattice and $f: L \to L$ be a monotone join-uniform operator (where $f^\imath$ is its generalized inverse). The following statements are equivalent*:

(i) *f is additive*;

(ii) *$f^\imath$ is monotone*;

(iii) *$f^\imath = f^r$.*

Proof. We prove the following implications.

(i) $\Rightarrow$ (ii) Let $x, y \in L$ such that $x \leqslant y$. Then, by monotonicity of f, $f(x) \leqslant f(y)$. Moreover, if for some $z \in L$, $f(z) = f(x)$ then, by additivity, $f(z \vee y) = f(z) \vee f(y) = f(y)$. Hence, we have that $f^\imath(x) = \bigvee \{z \in L \mid f(z) = f(x)\} \leqslant f^\imath(y) = \bigvee \{u \in L \mid f(u) = f(y)\}$.

(ii) $\Rightarrow$ (iii) Both f and $f^\imath$ are monotone, and, for any $x \in L$, $f(f^\imath(x)) \leqslant x$ and $f^\imath(f(x)) \geqslant x$. Then, f and $f^\imath$ constitute a Galois connection on L, and therefore, $f^\imath$ is the right-adjoint of f.

(iii) $\Rightarrow$ (i) Because in any adjunction the right-adjoint of a function exists if and only if this function is additive. ∎

The following result shows that, whenever a lower closure admits the right-adjoint, this is always an upper closure, and, therefore, if an abstract domain refinement admits the right-adjoint, this is a simplification operator. Conversely, by duality, the left-adjoint of a simplification operator, when it exists, is always a refinement.

Proposition 7.7. *Let L be a complete lattice.*

(i) *If $\rho \in lco(L)$ admits the right-adjoint ρ^r, then $\rho^r \in uco(L)$.*

(ii) *If $\rho \in uco(L)$ admits the left-adjoint ρ^l, then $\rho^l \in lco(L)$.*

Proof. We only prove (i), because (ii) follows by duality. As recalled in Section 2.3, in any adjunction on a complete lattice, the left-adjoint is additive and the right-adjoint is co-additive. Thus, let $\rho \in lco(L)$ be an additive lower closure that forms an adjunction on L with ρ^r. We prove that ρ^r is an upper closure. Monotonicity follows by adjunction. Let $x \in L$. Extensivity follows because from $\rho(x) \leqslant x$ we get $\rho^r(x) \geqslant x$. Let us now turn to idempotency. By (i) in Section 2.3, we get $\rho^r(\rho^r(x)) \geqslant \rho^r(x)$. Thus, let us prove that $\rho^r(\rho^r(x)) \leqslant \rho^r(x)$. By definition, $\rho^r(\rho^r(x)) = \bigvee_L \{y \in L \mid \rho(y) \leqslant \rho^r(x)\}$. Let $z \in \{y \in L \mid \rho(y) \leqslant \rho^r(x)\}$. Then, by monotonicity, idempotency, and additivity of ρ, we obtain

$$\begin{aligned} \rho(z) &\leqslant \rho\left(\bigvee \{y \in L \mid \rho(y) \leqslant x\}\right) \\ &= \bigvee \{\rho(y) \in L \mid \rho(y) \leqslant x\} \\ &\leqslant x. \end{aligned}$$

Hence, $\rho(z) \leqslant x$. Thus, we get $z \leqslant \rho^r(\rho(z)) \leqslant \rho^r(x)$, and from $z \leqslant \rho^r(x)$ we therefore obtain the idempotency. ▮

Thus, by the above result, given an abstract domain refinement $\mathfrak{R} \in lco(uco(C))$, requiring that $\mathfrak{R}$ admits an inverse on all $uco(C)$, which in addition is a simplification is equivalent to requiring that $\mathfrak{R}$ is additive. However, additivity for $\mathfrak{R}$ is a quite stronger requirement than join-uniformity, that corresponds to the invertibility of $\mathfrak{R}$ on all the space $uco(C)$. For instance, reduced product and disjunctive completion are relevant examples of refinements, which are not additive but still join-uniform (i.e., invertible) under certain nonrestrictive hypotheses on the concrete domain. The reduced product refinement is a paradigmatic case. It is easy to observe that the reduced product is additive if and only if $uco(C)$ is completely meet-distributive, i.e., a complete Heyting algebra. This latter condition is equivalent to the weaker (finite) distributivity of $uco(C)$ (cf. [Morgado 1962]), and this holds if and only if the concrete domain C is a complete chain (cf. [Dwinger 1954]). Obviously, being a complete chain for the concrete domain is not a reasonable hypothesis in semantics and program analysis. On the other hand, the general nonadditivity of the disjunctive completion refinement follows from Example 7.5 and Proposition 7.6.

7.4. *Reordering Abstract Domains*

We have seen that an invertible abstract domain refinement and its inverse does not constitute, in general, an adjunction. This asymmetry can be overcome by considering the lifted complete order induced on the lattice of abstractions by an invertible (i.e., join-uniform) refinement. In fact, dually to what we have seen in logic program semantics, join-uniformity becomes additivity with respect to the lifted complete order. In this way, for a refinement $\mathfrak{R}$ which, possibly under some conditions on the concrete domain C, is invertible on all $uco(C)$, by Proposition 7.6, we have that $\mathfrak{R}^r = \mathfrak{R}^-$, and, in particular, the inverse $\mathfrak{R}^-$ becomes a simplification operator on the lifted order. This is stated by the following consequence of Theorem 5.10, Proposition 7.6, and 7.7.

COROLLARY 7.8. *Let C be a complete lattice, and let $\mathfrak{R} \in lco^*(uco(C))$ be a refinement which is invertible on $uco(C)$. Then, $(\mathfrak{R}, uco(C)_{\sqsubseteq^{\mathfrak{R}}}, uco(C)_{\sqsubseteq^{\mathfrak{R}}}, \mathfrak{R}^-)$ is an adjunction.*

Although being not additive, both the reduced product refinement of Example 7.1 and the disjunctive completion refinement of Example 7.3 are join-uniform under certain weak hypotheses. The following examples show the meaning of the lifted order on abstract domains in these two cases. First, let us recall that if L is a meet semilattice with bottom, then the *pseudocomplement* of $x \in L$, if it exists, is the (unique) element $x^* \in L$ such that $x \wedge x^* = \bot$ and $\forall y \in L.(x \wedge y = \bot) \Rightarrow (y \leqslant x^*)$. In a complete lattice L, if the pseudocomplement x^* exists then $x^* = \bigvee \{y \in L \mid x \wedge y = \bot\}$. If every $x \in L$ has the pseudocomplement, L is called *pseudocomplemented* (for more details see, e.g., [Birkhoff 1967]).

EXAMPLE 7.9. Giacobazzi *et al.* [1996] proved that if C is a meet-continuous lattice,[4] then for each continuous closure $X \in uco(C)$, $\uparrow X = \{Y \in uco(C) \mid X \sqsubseteq Y\} \subseteq uco(C)$ is a pseudocomplemented lattice. In particular, $uco(C)$ itself, which coincides with $\uparrow C$, is pseudocomplemented. By the equivalence between closure operators and abstract domains, this result provided the basis for defining the operation of domain *complementation* in abstract interpretation [Cortesi *et al.* 1997]. Complementation is an operation which starting from any two abstract domains $A, B \in uco(C)$ such that $A \sqsubseteq B$ (i.e., $B \in \uparrow A$), gives as result the most abstract domain $A \sim B$ whose reduced product with B is A, i.e., $(A \sim B) \sqcap B = A$. Hence, the pseudocomplement of B in $\uparrow A$, when it exists, is denoted by $A \sim B$ and called complement of B in A (see [Cortesi *et al.* 1997] for more details).

Although a complement $A \sim B$ in $uco(C)$ is defined for any meet-continuous lattice C, continuous closure A and arbitrary B, in order to get that complementation is the inverse of the reduced product refinement on all the space $uco(C)$, we assume the more restrictive hypothesis that the concrete domain C satisfies the ACC. This condition in fact ensures that $A \sim B$ exists for any pair $A, B \in uco(C)$ such that $A \sqsubseteq B$, because C and any closure on C are, respectively, meet-continuous and continuous. Notice that C may well be thought of as a meaningful abstraction of the actual concrete domain, and in this case, the ACC is not a severe requirement for an abstract domain used in program analysis. It is easy to observe that complementation is the inverse of the reduced product refinement. In fact, for any $A \in uco(C)$, the refinement $\mathfrak{R}_{\sqcap A} = \lambda X.(A \sqcap X) \in lco(uco(C))$ is join-uniform on all $uco(C)$, because, for any $X \in uco(C)$, $A \sqcap X$ satisfies the ACC, and therefore $\uparrow(A \sqcap X)$ is pseudocomplemented. Moreover, it turns out that the inverse of $\mathfrak{R}_{\sqcap A}$ is $\mathfrak{R}^{-}_{\sqcap A} = \lambda X.(A \sqcap X) \sim A$, since, for any $X \in uco(C)$, $\bigsqcup \{Y \in uco(C) \mid A \sqcap Y = A \sqcap X\}$ is just the complement of A in $A \sqcap X$. Thus, by Corollary 7.8, reduced product and complementation form an adjunction relatively to the lifted complete order on abstract domains, and $\mathfrak{R}^{-}_{\sqcap A}$ is a simplification operator (namely, an upper closure) in the reordered complete lattice of abstract domains $\langle uco(C), \sqsubseteq^{(\mathfrak{R}_{\sqcap A})} \rangle$.

The lifted order between abstract domains induced by some $\mathfrak{R}_{\sqcap A}$ allows one to give a correct interpretation to its inverse $\mathfrak{R}^{-}_{\sqcap A}$. Consider for instance the abstract domains A^{+} of Fig. 8, A_2 of Fig. 9, already considered above, and the abstract domain D of Fig. 10. It turns out that $\mathfrak{R}^{-}_{\sqcap A^{+}}$ is not monotone with respect to the standard ordering $\sqsubseteq$ which relates abstract domains in terms of their relative precision. In fact, we have that $D \sqsubseteq A_2 \sqsubseteq A^{+}$, while $\mathfrak{R}^{-}_{\sqcap A^{+}}(D) = D \sim A^{+} = \{\mathbb{Z}, -0\}$ is not comparable with $\mathfrak{R}^{-}_{\sqcap A^{+}}(A_2) = A_2 \sim A^{+} = \{\mathbb{Z}, 0\}$, with respect to $\sqsubseteq$. Instead, $\mathfrak{R}^{-}_{\sqcap A^{+}}$ becomes monotone with respect to the lifted order $\sqsubseteq^{(\mathfrak{R}_{\sqcap A^{+}})}$, and therefore the optimal bases $\mathfrak{R}^{-}_{\sqcap A^{+}}(D)$ and $\mathfrak{R}^{-}_{\sqcap A^{+}}(A_2)$ are comparable: $\mathfrak{R}^{-}_{\sqcap A^{+}}(D) \sqsubseteq^{(\mathfrak{R}_{\sqcap A^{+}})} \mathfrak{R}^{-}_{\sqcap A^{+}}(A_2)$. This relationship reflects the relative power of the domain $\mathfrak{R}^{-}_{\sqcap A^{+}}(D) = \{\mathbb{Z}, -0\}$ with respect to $\mathfrak{R}^{-}_{\sqcap A^{+}}(A_2) = \{\mathbb{Z}, 0\}$: when composed with A^{+}, the first resulting in a more precise domain than the latter.

[4] A complete lattice C is *meet-continuous* if for any chain $Y \subseteq C$ and $x \in C$, $x \wedge (\bigvee Y) = \bigvee_{y \in Y} (x \wedge y)$ (cf. [Gierz *et al.* 1980]).

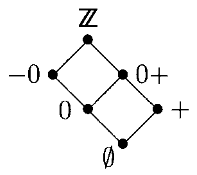


FIG. 10. The abstract domain D

As observed in the example above, the lifted complete order on abstract domains reflects precisely the relative precision of abstract domains with respect to a given invertible refinement operator: A is more precise than B in the lifted order for an invertible refinement $\mathfrak{R}$, if $\mathfrak{R}(A)$ is more precise than $\mathfrak{R}(B)$ in the standard sense, and, whenever they are the same (i.e., $\mathfrak{R}(A)=\mathfrak{R}(B)$), then A is more precise than B in the standard sense.

Example 7.10. Analogously to the above example of reduced product and complementation, the disjunctive completion refinement and its inverse, the least disjunctive basis (cf. Example 7.3), form an adjunction with respect to the lifted order. Thus, because when C is a dual-algebraic completely distributive lattice, $\mathfrak{R}_\vee$ is invertible on all $uco(C)$ (cf. Example 7.3), we get the adjunction $(\mathfrak{R}_\vee, uco(C)_{\sqsubseteq^{\mathfrak{R}_\vee}}, uco(C)_{\sqsubseteq^{\mathfrak{R}_\vee}}, \mathfrak{R}_\vee^-)$. Here again, the inverse $\mathfrak{R}_\vee^-$ can be interpreted as a simplification operator with respect to $\sqsubseteq^{\mathfrak{R}_\vee}$. Now, for the abstract domains *Sign* and A^+ considered in Example 7.5, it turns out that $\mathfrak{R}_\cup^-(Sign) \sqsubseteq^{\mathfrak{R}_\cup} \mathfrak{R}_\cup^-(A^+)$: in fact, when $\mathfrak{R}_\cup^-(Sign)$ is refined by the disjunctive completion, it provides a more concrete domain than $\mathfrak{R}_\cup^-(A^+)$ does.

8. CONCLUSION

In this work, we have introduced the order-theoretic notion of uniform closure operator on complete lattices and shown that this is the right key property for generalizing the standard hierarchy of declarative semantics for logic programming and to develop a general theory for domain refinement and simplification in abstract interpretation. These results have shown an unexpected relationship between two different fields of application of abstract interpretation. On the one hand, meet-uniformity and abstract interpretation provide the right framework for reordering semantic interpretations of logic programs, keeping into account the relative precision of the models of a program specified at different levels of abstraction in the hierarchy of semantics. On the other hand, the dual property of join-uniformity applied to abstract domain refinements yields the precise characterization of the concept of invertibility for a refinement, and moreover it has been proved that refinements and their inverses constitute an adjunction relatively to a reordered space of abstract domains. Both reordered spaces are based on the same lifted partial order induced by uniformity of the involved closures.

The notion of uniformity might be fruitfully exploited in other areas of application of abstract interpretation, such as in hierarchies of inductive definitions for specifying (nonnecessarily logic) program semantics and type systems. Inductive definitions are fundamental in mathematical logic and theoretical computer science to provide rule-based presentations of inductively defined sets, or equivalently in the definition of sets generated by closure conditions. In semantics, this is the case for the set of execution traces of a transition system, or more in general, in rule-based specification methods for semantics, e.g., in Plotkin's [1981] structural operational semantics SOS. As shown by Cousot and Cousot [1992b] and Cousot [1997b], hierarchies of semantics and type systems can be specified as inductive definitions and related with each other by abstract interpretation. We believe that the order-theoretic reconstruction of the model-theoretic semantics of logic programming presented in the paper can be further generalized by considering, instead of logic programs, generic inductive definitions. The relationship between logic programs and inductive definitions is well known: while logic programs are finite sets of untyped Horn-clauses, inductive definitions may be given as possibly infinite sets of possibly typed clauses or rules. An account of this analogy can be found in [Bol and Groote 1996]. As observed in Section 6.1, we put no constraint on the structure of programs. In particular, they can be possibly infinite sets of possibly typed clauses, i.e., arbitrary sets of rules. The essential point in our construction is that the space of all models of a program P is a closure m_P, and this is just the case of models of inductive definitions (cf. [Aczel 1977]). Thus, in view of our approach, the model-theoretic reconstruction of the semantics of logic programs of Falaschi *et al.* [1993] might be fully generalized to hierarchies of inductive definitions, and therefore applied to more general rule-based presentations of semantics or type systems related by abstract interpretation.

ACKNOWLEGMENTS

This work was begun in early 1995 while the authors were visiting the LIX, Laboratoire d'Informatique, École Polytechnique, in Paris. They thank the LIX, and in particular Radhia Cousot, for the kind hospitality. The work of Roberto Giacobazzi has been partly supported by the EEC Human Capital and Mobility individual Grant "Semantic Definitions, Abstract Interpretation and Constraint Reasoning", No. ERBCHBICT930822.

Received February 10, 1997; final manuscript received January 22, 1998

REFERENCES

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

Adaricheva, K., and Gorbunov, V. (1990), Equational closure operator and forbidden semidistributive lattices, *Siberian Math. J.* **30**(6), 831–849.

Amato, G., and Levi, G. (1997), Properties of the lattice of observables in logic programming, *in* "Proceedings of the Italian-Portuguese-Spanish Joint Conference on Declarative Programming, (APPIA-GULP-PRODE '97)," pp. 175–187.

Apt, K. R. (1990), Introduction to logic programming, *in* "Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics" (J. van Leeuwen, Ed.), pp. 495–574, Elsevier, Amsterdam, The MIT Press, Cambridge, Mass.

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.

Birkhoff, G. (1967), "Lattice Theory," AMS Colloquium Publications, Vol. XXV, 3rd ed., AMS, Providence, R.I.

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

Bossi, A., and Cocco, N. (1993), Basic transformation operations for logic programs which preserve computed answer substitutions, *J. Logic Program.* **16**, 47–87.

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

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

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

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

Comini, M., and Levi, G. (1994), An algebraic theory of observables, *in* "Proceedings of the 1994 International Logic Programming Symposium (ILPS '94)" (M. Bruynooghe, Ed.), pp. 172–186, MIT Press, Cambridge, MA.

Comini, M., Levi, G., and Meo, M. (1995), Compositionality of *SLD*-derivations and their abstractions, *in* "Proceedings of the 1995 International Symposium on Logic Programming (ILPS '95)" (J. Lloyd, Ed.), MIT Press, Cambridge, MA.

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

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

Cousot, P. (1996), Abstract interpretation, *ACM Comp. Surv.* **28**(2), 324–328.

Cousot, P. (1997a), Constructive design of a hierarchy of semantics of a transition system by abstract interpretation, *in* "Proceedings of the 13th International Symposium on Mathematical Foundations of Programming Semantics (MFPS '97)" (S. Brookes, and M. Mislove, Eds.), Electronic Notes in Theoretical Computer Science, Vol. 6, Elsevier, Amsterdam.

Cousot, P. (1997b), Types as abstract interpretations, *in* "Conference Record of the 24th ACM Symposium on Principles of Programming Languages (POPL '97)," pp. 316–331, ACM Press, New York.

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)," pp. 238–252, ACM Press, New York.

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

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

Cousot, P., and Cousot, R. (1992b), Inductive definitions, semantics and abstract interpretation, *in* "Conference Record of the 19th ACM Symposium on Principles of Programming Languages (POPL '92)," pp. 83–94, ACM Press, New York.

Cousot, P., and Cousot, R. (1994), Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and per analysis of functional languages), *in* "Proceedings of the IEEE International Conference on Computer Languages (ICCL '94)," pp. 95–112, IEEE Computer Society Press, Los Alamitos, CA.

Cousot, P., and Cousot, R. (1995), Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form, *in* "Proceedings of the 7th International Conference on Computer Aided Verification (CAV '95)" (P. Wolper, Ed.), Lecture Notes in Computer Science, Vol. 939, pp. 293–308, Springer-Verlag, Berlin.

Deutsch, A. (1992), "Operational models of programming languages and representations of relations on regular languages with application to the static determination of dynamic aliasing properties of data," Ph.D. thesis, University of Paris VI, Paris, France.

Dwinger, P. (1954), On the closure operators of a complete lattice, *Indagat. Math.* **16**, 560–563.

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

Fages, F., and Gori, R. (1996), A hierarchy of semantics for normal constraint logic programs, *in* "Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP '96)" (M. Hanus and M. Rodríguez-Artalejo, Eds.), Lecture Notes in Computer Science, Vol. 1139, pp. 77–91, Springer-Verlag, Berlin.

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

Falaschi, M., Levi, G., Martelli, M., and Palamidessi, C. (1993), A model-theoretic reconstruction of the operational semantics of logic programs, *Inf. Comput.* **103**(1), 86–113.

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

Filé, G., and Ranzato, F. (1998), The powerset operator on abstract interpretations, *Theor. Comput. Sci*, to appear.

Giacobazzi, R. (1996), "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics, *in* "Proceedings of the 13th International Symposium on Theoretical Aspects of Computer Science (STACS '96)" (C. Puech, Ed.), Lecture Notes in Computer Science, Vol. 1046, pp. 503–514, Springer-Verlag, Berlin.

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

Giacobazzi, R., and Ranzato, F. (1995), Functional dependencies and Moore-set completions of abstract interpretations and semantics, *in* "Proceedings of the 1995 International Symposium on Logic Programming (ILPS '95)" (J. Lloyd, Ed.), pp. 321–335, MIT Press, Cambridge, MA.

Giacobazzi, R., and Ranzato, F. (1996), Complementing logic program semantics, *in* "Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP '96)" (M. Hanus and M. Rodríguez-Artalejo, Eds.), Lecture Notes in Computer Science, Vol. 1139, pp. 238–253, Springer-Verlag, Berlin.

Giacobazzi, R., and Ranzato, F. (1997), Refining and compressing abstract domains, *in* "Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP '97)" (P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, Eds.), Lecture Notes in Computer Science, Vol. 1256, pp. 771–781, Springer-Verlag, Berlin.

Giacobazzi, R., and Ranzato, F. (1998), Optimal domains for disjunctive abstract interpretation, *Sci. Comput. Program* **32**(1–3), 177–210.

Gierz, G., Hofmann, K. H., Keimel, K., Lawson, J. D., Mislove, M., and Scott, D. S. (1980), "A Compendium of Continuous Lattices," Springer-Verlag, Berlin.

Granger, P. (1988), Combinations of semantic analyses, *in* "Proceedings of the 2nd French-Soviet Workshop on Methods of Compilation and Program Construction, Informatika '88" (P. Deransart, Ed.), pp. 71–88, INRIA, Rocquencourt, France.

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

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

Lassez, J., and Maher, M. (1984), Closures and fairness in the semantics of programming logic, *Theor. Comput. Sci.* **29**, 167–184.

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

Morgado, J. (1962), On complete congruences of complete lattices, *Portugal. Math.* **21**(1), 11–25.

Muthukumar, K., and Hermenegildo, M. (1991), Combined determination of sharing and freeness of program variables through abstract interpretation, *in* "Proceedings of the 8th International Conference on Logic Programming (ICLP '91)" (K. Furukawa, Ed.), pp. 49–63, MIT Press, Cambridge, MA.

Nielson, F. (1984), Abstract Interpretation using Domain Theory, Ph.D. thesis CST-31/84, University of Edinburgh, Edinburgh, Scotland.

Plotkin, G. (1981), A structural approach to operational semantics, Tech. Rep. DAIMI FN-19, Computer Science Dept., Aarhus University, Aarhus, Denmark.

Sundararajan, R., and Conery, J. (1992), An abstract interpretation scheme for groundness, freeness, and sharing analysis of logic programs, *in* "Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS '92)" (R. Shyamasundar, Ed.), Lecture Notes in Computer Science, Vol. 652, pp. 203–216, Springer-Verlag, Berlin.

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

Yi, K., and Harrison, W. L. (1993), Automatic generation and management of interprocedural program analyses, *in* "Conference Record of the 20th ACM Symposium on Principles of Programming Languages (POPL '93)," pp. 246–259, ACM Press, New York.