Intuitionistic Implication in Abstract Interpretation

Roberto Giacobazzi Francesca Scozzari

Dipartimento di Informatica, Università di Pisa
Corso Italia 40, 56125 Pisa, Italy
E-mail: {giaco, scozzari}@di.unipi.it
Ph.: +39-50-887283 Fax: +39-50-887226

Abstract. In this paper we introduce the notion of Heyting completion in abstract interpretation, and we prove that it supplies a logical basis to specify relational program analyses by means of intuitionistic implication. This provides a uniform algebraic setting where abstract domains can be specified by simple logic formulas, or as solutions of recursive abstract domain equations, involving few basic operations for domain construction. We apply our framework to study directionality in type inference and groundness analysis in logic programming.

Keywords: *Abstract interpretation, directional types, domains, intuitionistic logic, logic programming, static analysis.*

1 Introduction

One of the most attractive features of abstract interpretation is the ability to systematically derive program analyses from formal semantics specifications. Analyses can be composed, refined and decomposed according to the need. A key role in this construction is played by abstract domains: *Modifying domains corresponds to modify analyses.* A number of operations were designed to systematically construct new abstract domains, by refinement ([12,14,15,17,18,21,26]), decomposition ([10]) and compression ([19]), and some of them are included as tools for design aid in modern systems for program analysis, like in GAIA ([12]), PLAI ([9]), and System Z ([27]), providing high-level facilities to tune the analysis in accuracy and costs. Most of these operations arise from the common logical operations on program properties. For instance, the well known *reduced product* ([14]) of two domains is the most abstract domain which is approximated by both the given domains. From the analysis point of view, it corresponds to the simplest analysis which is more precise than the two given analyses, i.e. which derives at least the same properties. The *disjunctive completion* ([14,15]) of a domain is the most abstract domain able to deal with disjunctions of properties in the given domain.

The possibility to tune the analysis in independent attribute and relational mode is probably one of the most attractive feature of this technology. The efficient but less precise *independent attribute* analysis method assigns properties to program objects, independently from the properties of other objects. This method is generalized by the more precise but costly *relational* one, which exploits at each program point the relations between the properties of program objects ([23]). This can be achieved by including relations or functions as domain objects. The very first example of domain constructor for relational analysis by abstract interpretation is Cousot's *reduced (cardinal) power* [14], where monotone functions are used to represent dependency information between properties of program objects. Although the notion of dependency is fundamental in analysis, e.g. for the efficient execution of programs on parallel computers, this construction is rarely applied in the literature, even if, as we prove in this paper, surprisingly, most well known domains for relational program analysis can be specified in this way. This is probably due to the lack of a clean logical interpretation for reduced power. While reduced product and disjunctive completion have both a clean and immediate logical interpretation as conjunction and disjunction of program properties, there is no known logical interpretation for most of the relational combinators, in particular for reduced power.

In this paper we fill this gap by introducing a new operation for abstract domain refinement, which is *Heyting completion*. The idea is to consider the space of all intuitionistic implications built from every pair of elements of the given domain. This domain has an immediate logical interpretation as the collection of intuitionistic formulas built, without nested implication, in the fragment $\wedge, \rightarrow$. We prove that this operation provides a logical foundation for Cousot's reduced power of abstract domains. Moreover, this has the advantage of making applicable relevant results in intuitionistic logic to abstract domain manipulation. We study the algebraic properties of Heyting completion in relation with reduced product and disjunctive completion. The result is an advanced algebra of domain operations, where abstract domains can be specified by simple logic formulas, using few basic operations for domain construction, all characterized by a clean logical interpretation. In this algebraic setting, we study the solutions of implicational domain equations, i.e. equations between abstract domains involving Heyting completion as basic domain constructor. A new family of abstract domains for analysis is introduced as solutions of these equations. These domains are closed under dependency construction, and somehow represent an upper bound on how much this information can be derived systematically from a given domain. This provides also a first account on the possibility to study domains for analysis as solutions of recursive domain equations, as traditionally done for domains in denotational semantics.

We apply Heyting completion as domain refinement to directional type inference in logic programming. Type inference is general enough to include relevant examples of analyses in logic programs. We prove that the same construction by Heyting completion surfaces most of the well known applications of type inference, from groundness ([3]) to more structured directional types ([6,1]). This approach to directionality by systematic domain refinement has the advantage of being independent from specific applications. This supplies a better comprehension on the nature of directionality in these analyses, and suggest how to construct new domains for program analysis.

2 Related works

In the very last section of [14], Cousot and Cousot introduced the notion of reduced power, as a domain of monotone functions between two given abstract domains. This construction was reconsidered in [18], where some properties on the lattice structure of reduced power are studied. The main result in [18] is a necessary and sufficient condition on the abstract domain A, such that its reduced power with A itself is isomorphic to a space of closure operators on A. None of these papers however gave a logical interpretation for the operation of reduced power, in particular in relation with other known operations for domain manipulation. In addition, none of them characterize the notion of domains closed under dependency construction. Nielson's *tensor product* [26] is also known as a basic operator for relational analyses construction. However this operation is not directly comparable, in the sense of abstract interpretation (i.e. it provides neither more abstract nor more concrete domains) with reduced power, as proved in [18]. On the side of program analysis, Codish and Demoen [8] firstly used the directional information obtainable for groundness analysis in *Pos* (also called *Prop*) ([3]), to implement efficient type inference algorithms by abstract interpretation. Their results however are not immediately applicable to other program properties, due to the lack of a suitable domain completion, generalizing their construction.

3 Preliminaries

3.1 Notation and basic notions

Let A, B and C be sets. The powerset of A is denoted by $\wp(A)$. If $X \subseteq A$, $\overline{X}$ is the set-theoretic complement of X. If A is a poset or a pre-ordered set, we usually denote $\leq_A$ the corresponding partial order. If A is pre-ordered and $I \subseteq A$ then $\downarrow I = \{x \in A \mid \exists y \in I.\ x \leq_A y\}$. For $x \in A$, $\downarrow x$ is a shorthand for $\downarrow \{x\}$. $\wp^{\downarrow}(A)$ denotes the set of *order-ideals* of A, where $I \subseteq A$ is an order-ideal if $I = \downarrow I$. $\wp^{\downarrow}(A)$ is a complete lattice with respect to set-theoretic inclusion, where the join is set union and the meet is set intersection. We write $f : A \longmapsto B$ to mean that f is a total function from A to B. If $C \subseteq A$ then $f(C) = \{f(x) \mid x \in C\}$. By $g \circ f$ we denote the composition $\lambda x.g(f(x))$. Let $\langle A, \leq_A, \vee_A, \wedge_A, \top_A, \bot_A \rangle$ and $\langle B, \leq_B, \vee_B, \wedge_B, \top_B, \bot_B \rangle$ be complete lattices. A function $f : A \longmapsto B$ is *(co-)additive* if for any $C \subseteq A$, $f(\vee_A C) = \vee_B f(C)$ ($f(\wedge_A C) = \wedge_B f(C)$). f is (co-)continuous when the above statement on (co-)additivity holds for chains C only. If A is a complete lattice, the *pseudo-complement* of *a relatively* to b, if it exists, is the unique element $a \to b \in A$ such that for any $x \in A$: $a \wedge_A x \leq_A b$

iff $x \leq_A a \to b$. A is *relatively pseudo-complemented* if $a \to b$ exists for every $a, b \in A$. The *pseudo-complement* of a, if it exists, is $a \to \bot_A$ (see [5]).

3.2 Abstract interpretation, Galois connections and closure operators

The standard Cousot and Cousot theory of abstract interpretation is based on the notion of Galois connection ([13]). If C and A are posets and $\alpha : C \longmapsto A$, $\gamma : A \longmapsto C$ are monotone, such that $\forall c \in C.\ c \leq_C \gamma(\alpha(c))$ and $\forall a \in A.\ \alpha(\gamma(a)) \leq_A a$, then we call the quadruple $\langle C, \gamma, A, \alpha \rangle$ a *Galois connection* (G.c.) between C and A. If in addition $\forall a \in A.\ \alpha(\gamma(a)) = a$, then $\langle C, \gamma, A, \alpha \rangle$ is a *Galois insertion* (G.i.) of A in C. In the setting of abstract interpretation, C and A are called, respectively, *concrete* and *abstract domain*, and they are assumed to be complete lattices. Any G.c. $\langle C, \gamma, A, \alpha \rangle$ can be lifted to a G.i. identifying in an equivalence class those objects in A having the same image (meaning) in C. This process is known as *reduction* of the abstract domain. In the rest of the paper, L is a complete lattice $\langle L, \leq_L, \vee_L, \wedge_L, \top_L, \bot_L \rangle$ playing the role of the concrete domain.

An (*upper*) *closure operator* on L is an operator $\rho : L \longmapsto L$ monotonic, idempotent and extensive (viz. $\forall x \in L.\ x \leq_L \rho(x)$) [24]. Each closure operator ρ is uniquely determined by the set of its fixpoints, which is its image $\rho(L)$. Hence, in the following, when closures are used to denote abstract domains, they will be denoted as sets, with Roman capital letters. $X \subseteq L$ is the set of fixpoints of a closure operator on L iff X is a *Moore-family* of L, i.e. $\top_L \in X$ and X is completely meet-closed (viz. for any non-empty $Y \subseteq X$, $\wedge_L Y \in X$). For any $X \subseteq L$, we denote by $\curlywedge(X)$ the *Moore-closure* of X, i.e. the least subset of L containing X, which is a Moore-family of L. $\rho(L)$ is a complete lattice with respect to $\leq_L$, but, in general, it is not a complete sublattice of L, since the join in $\rho(L)$ might be different from $\vee_L$. $\rho(L)$ is a complete sublattice of L iff ρ is additive. We denote by $\langle uco(L), \sqsubseteq, \sqcap, \sqcup, \lambda x.\top, \lambda x.x \rangle$ the complete lattice of all upper closure operators on L, where the ordering is pointwise i.e., $\rho \sqsubseteq \eta$ iff $\forall x \in L.\ \rho(x) \leq_L \eta(x)$, or equivalently $\eta(L) \subseteq \rho(L)$.

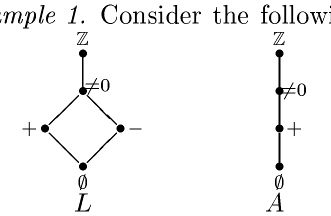
The equivalence between G.i. and closure operators is well known [5]. However, closure operators are often more practical and concise than G.i.'s to reason about abstract domains, being independent from representation choices for domain objects ([14]). Any G.i. $\langle L, \gamma, A, \alpha \rangle$ is uniquely determined (up to isomorphism) by the closure operator $\gamma \circ \alpha$, and, conversely, any closure operator uniquely determines a G.i. (up to isomorphism). The complete lattice of all abstract domains (identified up to isomorphism) on L is therefore isomorphic to $uco(L)$. Thus, in the rest of the paper, we often abuse notation and consider an abstract domain and the corresponding closure as the same object, so whenever we will say that D is an abstraction of C, we will mean that D is isomorphic to $\rho(C)$, for some closure $\rho \in uco(C)$. The order relation on $uco(L)$ corresponds precisely to the order used to compare abstract domains with regard to their precision: If A and B are abstractions of L, then A is *more precise* than B iff $A \sqsubseteq B$ as closures.

4 Refinement by pseudo-complements

An abstract domain for analysis is usually a set of objects, equipped with a partial order. In this structure the information is associated with both the domain objects and the way they are ordered—the top-element representing no information. An *abstract domain refinement* is intended to discover the information apparently hidden in a domain, by exploiting both the domain objects and their relation. Filé et al. studied this notion in [16], as a generalization for most of the well known operations for domain refinement, like *reduced product*, and *disjunctive completion*. A domain refinement is a mapping $\Re : uco(L) \longmapsto uco(L)$, such that for any abstract domain $A \in uco(L)$: $\Re(A)$ contains more information than A (i.e., $\Re$ is *reductive* $\Re(A) \sqsubseteq A$), and $\Re$ is monotonic. Idempotent domain refinements play an important role in this theory, as they upgrade domains all at once. This last condition clearly defines *idempotent refinements as lower closure operators* on $uco(L)$, i.e. monotonic, idempotent and reductive operators. In the following the set of lower closure operations on L is denoted $lco(L)$. Their properties follow by duality from those above for upper closure operators.

In this section we are interested in determining complementary information in abstract interpretation by domain refinement. This information may be useful to enhance domains, e.g. to represent negative information. Given two domain objects a and b, we are interested in characterizing, by a simple domain transformation, an object c such that, when combined with a, the result is approximated by b.

Example 1. Consider the following lattice L for *sign analysis*, which is an abstract interpretation of $\wp(\mathbb{Z})$, and one of its strict abstractions A, both depicted below. The objects in L and A have the obvious meaning as fixpoints of closures on $\wp(\mathbb{Z})$. $- \in L$ represents the necessary information which, when combined with $+ \in A$, gives $\emptyset$ as result.

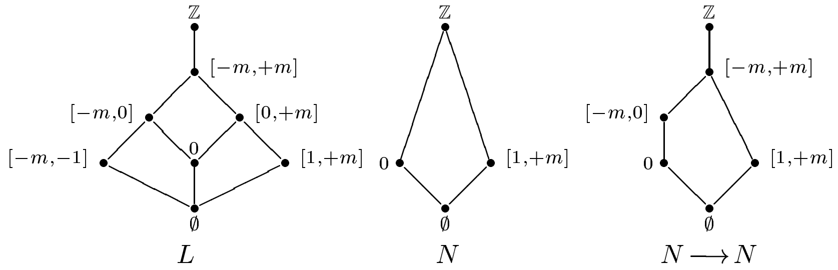


The refinement by complements was introduced in [16]. This refinement is based on the notion of complement, viz. a^* is a complement for a in L if $a \wedge_L a^* = \bot_L$ and $a \vee_L a^* = \top_L$. When the complement exists unique for an element a, it summarizes the common information shared by all the elements c such that $c \wedge_L a = \bot_L$. The previous example shows that the requirement of complementation that $a \vee_L a^* = \top_L$ is too strong to enhance the simple domain of signs A (note that, in Example 1, $-$ is not the complement for $+$ in L). We weaken the definition of complementation in two ways. First, by relaxing that $a \vee_L a^* = \top_L$, the above observation boils down to the well known notion of pseudo-complement, which, when it exists, is unique, indeed $a \to \bot_L = \vee_L\{\, c \mid a \wedge_L c = \bot_L \,\}$. Second, we relativize this notion by considering relative pseudo-complements, which, when they exist, are uniquely given by $a \to b = \vee_L\{\, c \mid a \wedge_L c \leq_L b \,\}$.

Definition 2. Let $A, B \subseteq L$. We define $A \longrightarrow B = \{\, a \to b \in L \mid a \in A, b \in B \,\}$.

It is worth noting that in the above definition we do not require any further hypothesis on the structure of L, apart being a complete lattice. Hence, although $a \to b = \vee_L\{\, c \mid a \wedge_L c \leq_L b \,\}$ always exists unique in complete lattices, it may in general fail to be the pseudo-complement of a relative to b. The following example shows this phenomenon.

Example 3. Consider the following simple lattice L for *overflow analysis* of integer variables, which is an abstract interpretation of the standard lattice for *interval analysis* in [13]. In this example, $m \in \mathbb{N}$ is a constant, playing the role of *maxint*.



Note that L is neither distributive, nor pseudo-complemented. If N is the domain depicted above, which is a strict abstraction of L, although being defined, $0 \to \emptyset = 0 \to [1,+m] = [-m,+m]$ is neither the pseudo-complement of 0, nor the pseudo-complement of 0 relative to $[1,+m]$. In the case of $[1,+m] \to 0 = [1,+m] \to \emptyset$ instead, we obtain precisely the relative pseudo-complement which is indeed $[-m,0]$. Note that when an object $a \to b$ is not the relative pseudo-complement, it still provides an approximation for the information which is complementary to a. In this case, the complementary information for 0, which should be given by the union of the intervals $[-m,-1] \cup [1,+m]$, is approximated by their convex hull $0 \to \emptyset = [-m,+m]$.

We extend the domain transformation $\longrightarrow$ to a binary abstract domain refinement. Given two abstract domains X and Y, this is precisely captured by the least Moore family (viz. abstract domain) containing $X \longrightarrow Y$.

Definition 4. Let $A, B \in uco(L)$. The *Heyting completion of A wrt B*, denoted by $A \stackrel{\curlywedge}{\longrightarrow} B$, is $\curlywedge(A \longrightarrow B)$.

Theorem 5. *Let $A \in uco(L)$. $\lambda X, Y.\ X \stackrel{\curlywedge}{\longrightarrow} Y$ is argumentwise monotonic on $uco(L)$, and $\lambda X.\ A \stackrel{\curlywedge}{\longrightarrow} X$ is reductive on $uco(L)$.*

In the following, the Heyting completion of a domain X is $X \stackrel{\curlywedge}{\longrightarrow} X$. Note that, while the Heyting completion operator $\lambda X.\ X \stackrel{\curlywedge}{\longrightarrow} X : uco(L) \longmapsto uco(L)$ is monotonic, it is not an idempotent refinement. For instance, in Example 3, we have that $[0,+m] \notin N \stackrel{\curlywedge}{\longrightarrow} N$, but $[0,+m] \in (N \stackrel{\curlywedge}{\longrightarrow} N) \stackrel{\curlywedge}{\longrightarrow} (N \stackrel{\curlywedge}{\longrightarrow} N)$.

5 Algebraic properties

In this section we show some basic algebraic properties of Heyting completion with respect to other domain operations. To this aim, we embed the lattice of abstract domains into an algebra of primitive operators for domain refinement. In particular, we consider the meet operation ($\sqcap$) of closure operators, which is (isomorphic to) the well known *reduced product* ([14]) of abstract domains, together with the above Heyting completion, and the operation for *disjunctive completion* ($\curlyvee$) ([14,15,19]).

In the rest of the paper, $A, B, C \in uco(L)$, and denote generic abstract domains.

Proposition 6.

1. $A \stackrel{\curlywedge}{\longrightarrow} \{\top_L\} = \{\top_L\}$
2. $\{\top_L\} \stackrel{\curlywedge}{\longrightarrow} A = A$
3. $(A \stackrel{\curlywedge}{\longrightarrow} C) \sqcap (B \stackrel{\curlywedge}{\longrightarrow} C) = \curlywedge((A \cup B) \longrightarrow C)$[1]

Complete Heyting algebras play a central role in our construction, as they correspond precisely to the models of propositional intuitionistic logic. L is a *complete Heyting algebra* (cHa), if for each $X \subseteq L$ and $y \in L$, $y \wedge_L (\vee_L X) = \vee_L \{ y \wedge_L x \mid x \in X \}$. If L is cHa, then it is relatively pseudo-complemented, and $\rightarrow$ models intuitionistic implication. Hence, $A \stackrel{\curlywedge}{\longrightarrow} A$ is the most abstract domain containing the implications $a \rightarrow b$ for $a, b \in A$.

Proposition 7. *Let L be a cHa.*

1. $A \stackrel{\curlywedge}{\longrightarrow} (C \sqcap B) = (A \stackrel{\curlywedge}{\longrightarrow} C) \sqcap (A \stackrel{\curlywedge}{\longrightarrow} B)$
2. $(A \sqcap B) \stackrel{\curlywedge}{\longrightarrow} C = A \stackrel{\curlywedge}{\longrightarrow} (B \stackrel{\curlywedge}{\longrightarrow} C)$
3. $A \stackrel{\curlywedge}{\longrightarrow} \curlywedge(X) = \sqcap_{x \in X} A \stackrel{\curlywedge}{\longrightarrow} \{\top_L, x\}$ *if* $X \subseteq L$.

Note that by (3) in Proposition 7, Heyting completion can be constructed from *atomic domains*, i.e., closures of the form $\{ \top_L, x \}$, by considering only the elements x in a generating[2] set X. In this case, if $x \in L$, then $A \stackrel{\curlywedge}{\longrightarrow} \{\top_L, x\}$ is the most abstract domain which contains the pseudo-complements of the elements in A in the interval lattice $[x, \top_L] \subseteq L$[3].

The disjunctive completion was introduced in [14] to express *merge over all paths* data-flow analysis in fixpoint form. The idea is to lift a given abstract domain A, to the least (most abstract) domain which includes A and which is a complete (join-) sublattice of L, viz. no approximation is induced by considering the join of abstract objects. The disjunctive completion of A is defined as the most abstract domain which is an additive closure and includes A (cf. [14,19]):

[1] Note that $A \cup B$ may not be, in general, a Moore-family.
[2] $A \subseteq L$ is generated by $X \subseteq A$ if $A = \curlywedge(X)$.
[3] if $a, b \in L$, the *interval lattice* $[a, b]$ is the complete sub-lattice of L: $\{x \in L \mid a \leq_L x \leq_L b\}$.

$\curlyvee(A) = \sqcup\{\, X \in uco(L) \mid X \sqsubseteq A$ and X is additive $\}$. The following result specifies the disjunctive completion of an abstract domain A when the concrete domain L is either collecting or a cHa with A finite. Note that *collecting domains*, i.e., domains of the form $\wp(S)$ for some set S, are typical concrete domains in collecting semantics for program analysis (e.g. [15,22,25]).

Proposition 8. *If L is collecting or L is a cHa and A is finite, then $\curlyvee(A) = \{\, \vee_L X \mid X \subseteq A \}$.*

Next result follows immediately from the logical properties of intuitionistic implication, and it specifies the relation between disjunctive and Heyting completions.

Proposition 9. *If L is collecting or L is a cHa and A is finite, $\curlyvee(A) \xrightarrow{\curlywedge} B = A \xrightarrow{\curlywedge} B$.*

The inverse operation to disjunctive completion, which is called *disjunctive optimal basis*, was introduced in [19]. Given an abstract domain A, the idea is to find the most abstract domain X (when it exists) such that $\curlyvee(A) = \curlyvee(X)$. This domain, denoted $\Omega(A)$, is such that $A \sqsubseteq \Omega(A)$. Giacobazzi and Ranzato proved in [19] that $\Omega(A)$ exists under some hypotheses, e.g. when L or A are finite, or when L is collecting, and $\Omega(A) = \curlywedge(\{\, x \in A \mid \forall Y \subseteq A.\ x = \vee_L Y \Rightarrow x \in Y \,\})$.

Corollary 10. *If L is collecting or L is a cHa and A is finite, then $A \xrightarrow{\curlywedge} B = \Omega(A) \xrightarrow{\curlywedge} B$.*

This result specifies that $A \xrightarrow{\curlywedge} B$ can always be obtained by considering the more abstract, and hence less expensive, domain $\Omega(A)$. This because, the implicational information generated by the disjunctive information in A, i.e. those objects $x \to b$ where $x \in A$ is such that there exists $C \subseteq A$ and $x = \vee_L C$, can be reconstructed by conjunction of relative pseudo-complements of the form $c \to b$, with $c \in C$. Here, the disjunctive optimal basis plays the role of the least domain generating A by disjunctive completion. By combining the above result with (3) in Proposition 7, we obtain the following result, with $X \subseteq L$ playing the role of generating set. This result may be useful to implement Heyting completion of abstract domains, from the most abstract arguments.

$$A \xrightarrow{\curlywedge} \curlywedge(X) = \sqcap_{x \in X} \Omega(A) \xrightarrow{\curlywedge} \{\top_L, x\}.$$

Note that, the implementation of the Heyting completion $A \xrightarrow{\curlywedge} B$ involving the disjunctive optimal basis of A and an atomic decomposition of B, requires a lower number of reductions to identify pairs $a \to b$ having the same meaning. In particular, whenever $A = \wp(S)$ for some set S and A is a disjunctive domain, then an upper bound to the number of elements in $A \xrightarrow{\curlywedge} A$ is not $|A|^2$, but $2|A|$. This because $|\Omega(A)| = log(|A|)$, $A = \curlywedge(\{\, S \setminus \{\, x \,\} \mid x \in S \,\})$ and $|\{\, S \setminus \{\, x \,\} \mid x \in S \,\}| = log(|A|)$. In this case $|A \xrightarrow{\curlywedge} A| \leq 2|A|$.

6 Reduced power and Heyting completion

Reduced power was introduced by Cousot and Cousot in [14], under the hypothesis that the concrete domain is some Boolean lattice of assertions. This construction was generalized in [18], where weaker hypotheses (e.g. cHa's) are considered for the existence of reduced power of domains. We follow this presentation, and we call *dependencies* the objects in the reduced power of domains.

We are interested in the case of *autodependencies*, which are monotone functions from an abstract domain A into itself. Monotone functions are considered equivalent, and therefore reduced, if they represent the same dependency between the objects of A. If $A = \rho_A(L)$ for $\rho_A \in uco(L)$, an autodependency is specified by the function $\lambda x.\ \rho_A(d \wedge_L x) : A \longmapsto A$, with $d \in L$. The set of all such dependencies is the reduced power $A^A = \{\, \lambda x \in A.\ \rho_A(d \wedge_L x) \mid d \in L\,\}$. Note that considering autodependencies in A^A is not a serious restriction, because $A^B \subseteq (A \sqcap B)^{(A \sqcap B)}$ ([18]). Next result extends Theorem [14, 10.2.0.1] to cHa's and proves that A^A refines A.

Theorem 11 ([18]). *Let L be a cHa and $A = \rho_A(L)$ for $\rho_A \in uco(L)$. Then $\langle L, \boldsymbol{\alpha}, A^A, \gamma \rangle$ is a G.i. where $\boldsymbol{\alpha} = \lambda d \in L.\ (\lambda x \in A.\ \rho_A(d \wedge_L x))$. Moreover $A^A \sqsubseteq A$.*

The following technical lemma is essential to give a representation result for dependencies in A^A, in terms of relative pseudo-complements.

Lemma 12. *Let L be a cHa and $\rho \in uco(L)$. Then, for any $d \in L$: $\gamma(\lambda x \in \rho(L).\ \rho(d \wedge_L x)) = \bigwedge_{a \in \rho(L)} (a \to \rho(d \wedge_L a))$.*

The following representation result for autodependencies is the main result of this section, and it specifies that, for autodependencies, reduced power is equivalent to Heyting completion.

Theorem 13. *If L is a cHa and $A = \rho_A(L)$ for $\rho_A \in uco(L)$, then $A^A = A \xrightarrow{\curlywedge} A$.*

This result proves a strong link between two apparently unrelated domain refinements, namely between domain refinement by reduced power, and domain completion by relative pseudo-complements.

7 Examples

7.1 Implicational groundness analysis

Pos is the domain most widely used for groundness analysis of logic programs ([3]). *Pos* is able to characterize both pure groundness, i.e. whether a variable is instantiated to ground terms during program execution, and the relations between the groundness of different program variables, providing in this sense a clear example of relational analysis. In this section we show that the domain Pos of positive Boolean functions can be interpreted as an implicational domain. Pos is the result of Heyting completion of a simpler domain for pure groundness. This domain is obtained by lifting Jones and Søndergaard's domain for pure groundness analysis $\mathcal{G}$ ([22]) by disjunctive completion. The disjunctive completion gives here the least amount of disjunctive information which is typical of Pos.

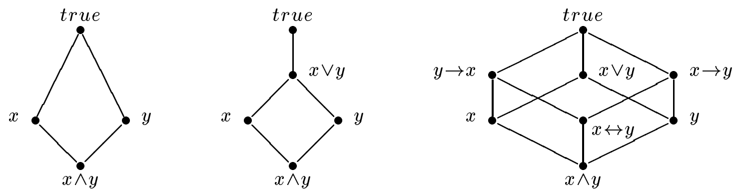
We fix a first-order language $\mathcal{L}$, with variables ranging in $\mathcal{V}$. The set of terms and idempotent substitutions, viz. mappings from $\mathcal{V}$ to terms in $\mathcal{L}$, are denoted respectively $T_{\mathcal{L}}$ and Sub. Objects in $T_{\mathcal{L}}$ and Sub are partially ordered by instantiation: $a \leq b$ iff $\exists \theta \in Sub.\ a = b\theta$. For any syntactic object s, $var(s)$ denotes the set of its variables.

Let $Var \subseteq \mathcal{V}$ be a finite set, representing the set of interesting variables. Pos is the set of (classic) propositional formulas built on Var and the constant $true$, by using the connectives $\wedge, \vee, \rightarrow$. The interpretation of the connectives is the classical one: $\phi \leq \psi$ if and only if ψ is a logical consequence of ϕ. We say that $I \subseteq Var$ is a model for ϕ, denoted $I \models \phi$, if ϕ is true in the interpretation which assigns $true$ to all the variables in I and $false$ to the other ones [3]. For the sake of simplicity we write $\theta\ \mathtt{grounds}\ \phi$ to denote $\{\, x \in Var \mid var(\theta(x)) = \emptyset \,\} \models \phi$, for $\theta \in Sub$.

Because logic programs compute substitutions, the concrete domain we consider is $\wp^{\downarrow}(Sub)$, which is a cHa. Pos is an abstraction of $\wp^{\downarrow}(Sub)$, and the concretization function is $\gamma_{Sub}(\phi) = \{\, \theta \in Sub \mid \forall \sigma \leq \theta.\ \sigma\ \mathtt{grounds}\ \phi \,\}$ ([11]). $\mathcal{G} \subset Pos$ is the standard domain for pure groundness analysis defined in [22], which is a strict abstraction of Pos, including formulas on Var with the only connective $\wedge$. As usual, we identify Pos and $\mathcal{G}$ with the corresponding closures on $\wp^{\downarrow}(Sub)$: $\gamma_{Sub}(Pos)$ and $\gamma_{Sub}(\mathcal{G})$.

Theorem 14. $Pos = \curlyvee(\mathcal{G}) \xrightarrow{\curlywedge} \curlyvee(\mathcal{G})$.

Note that, by Proposition 9, $Pos = \mathcal{G} \xrightarrow{\curlywedge} \curlyvee(\mathcal{G})$. For $Var = \{x, y\}$, the domains $\mathcal{G}$, $\curlyvee(\mathcal{G})$ and Pos are depicted below.



$\mathcal{G}$ $\quad$ $\curlyvee(\mathcal{G})$ $\quad$ $\curlyvee(\mathcal{G}) \xrightarrow{\curlywedge} \curlyvee(\mathcal{G})$

7.2 Directionality in type inference

In this section we consider the problem of inferring types in logic programs by abstract interpretation. Given a domain for (basic) type analysis, we show how the Heyting completion of such a domain captures the notion of directionality among types and, more in general, how this construction surfaces directionality in any property closed under instantiation. In this case, Heyting completion supplies a domain for type inference, including the same dependency information between types as traditionally obtained in standard results on directional type checking by verification methods [1].

The operational semantics of a logic program P, viz. a finite set of Horn clauses in $\mathcal{L}$, is defined by SLD-resolution. If G is a goal, viz. a conjunction of atoms in $\mathcal{L}$, then we write $G \xrightarrow{\vartheta}{}^{*}_{P} \square$ iff $\vartheta \in Sub$ is a computed answer substitution for G in P ([2]). A type is a set of terms closed under substitution [1]. Hence, the concrete domain in which we interpret types is $\wp^{\downarrow}(T_{\mathcal{L}})$. An (abstract) domain of types $\mathcal{T}$ is therefore any abstraction of $\wp^{\downarrow}(T_{\mathcal{L}})$.

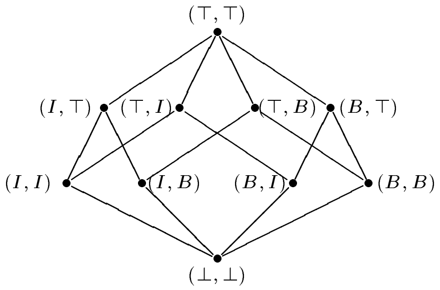
A type for a predicate p of arity n is an n-tuple of types. In the following, for the sake of simplicity, we assume that all predicates in the program have the same arity n, and abuse $T_{\mathcal{L}}$ to denote n-tuples of terms. A *directional type* for a predicate p has the form $\langle I, O\rangle$ where I and O are types for p. The type I is the "input" type and type O is the "output" type of p. A predicate p has type $\langle I, O\rangle$ in a program P (denoted $p \in \langle I, O\rangle$) if for any goal $p(t)$, $t \in I \;\Rightarrow\; \theta(t) \in O$ for any answer substitution θ (cf. [1]). Next result relates the Heyting completion refinement to the notion of directional type. The idea is to prove that an implicational type $\tau_1 \to \tau_2$ models exactly the above notion of directional type.

Theorem 15. *Let P be a program, $\mathcal{T}$ be a domain of types, $\tau_1, \tau_2 \in \mathcal{T}$ and p be a predicate. Then* $(\forall\theta.\ p(\bar{X}) \xrightarrow{\theta}{}^{*}_{P} \square \Rightarrow \bar{X}\theta \in \tau_1 \to \tau_2) \;\Leftrightarrow\; (\forall \bar{t} \in \tau_1 . \forall\sigma.\ p(\bar{t}) \xrightarrow{\sigma}{}^{*}_{P} \square \Rightarrow \bar{t}\sigma \in \tau_2)$.

Corollary 16. *In the hypothesis of Theorem 15, p has type $\tau_1 \to \tau_2$ in $\mathcal{T} \xrightarrow{\curlywedge} \mathcal{T}$ iff $p \in \langle \tau_1, \tau_2 \rangle$.*

Clearly, the domain $\mathcal{T} \xrightarrow{\curlywedge} \mathcal{T}$ is still closed under instantiation. Directional types in $\mathcal{T} \xrightarrow{\curlywedge} \mathcal{T}$, that satisfy the theorem above, can be obtained by abstract interpretation, with abstract domain $\mathcal{T} \xrightarrow{\curlywedge} \mathcal{T}$, by means of any bottom-up/top-down analyzer which is able to approximate correct or even computed answer substitutions (e.g. [4,7]).

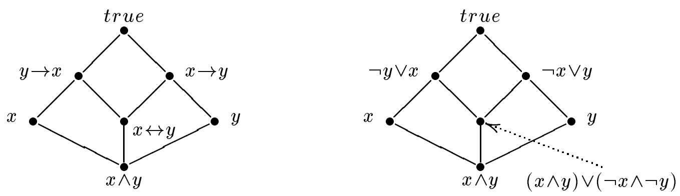
Example 17. Let us consider the program $P = \{\, p(X, X).,\, r(2, Y) : -p(2, Y).\,\}$ and the following simple domain of basic types $\mathcal{T}$, where symbols I (integer) and B (Boolean) have the obvious meaning as fixpoints of a closure operator on $\wp^{\downarrow}(T_{\mathcal{L}})$. A standard type inference which can be obtained from the above domain



of types $\mathcal{T}$, returns the following type patterns: $\{\, p(\top,\top), r(I,\top)\,\}$. Although correct, this result is not very precise. By abstract interpretation in the refined domain $\mathcal{T} \xrightarrow{\curlywedge} \mathcal{T}$, we obtain the more precise type patterns: $\{\, p(X, X)., r(I, I)\,\}$, where $p(X, X)$ is a shorthand for $\bigwedge_{(\tau_1,\tau_2)\in\mathcal{T}} p(\tau_1, \tau_2) \to p(\tau_1 \cap \tau_2, \tau_1 \cap \tau_2)$, i.e. $p(I,\top) \to p(I, I) \wedge p(\top, I) \to p(I, I) \wedge p(B,\top) \to p(B, B) \wedge p(\top, B) \to p(B, B)$.

7.3 From types back to groundness

A similar result of the above one was proved in [18] in the particular case of ground dependency analysis, with no disjunctive information. The authors proved that the abstract domain $Def \subset Pos$ ([3]), which contains formulas whose models are closed under intersection, is, in view of Theorem 13, precisely $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$. Hence in this case, pure groundness $\mathcal{G}$ or disjunctive groundness $\curlyvee(\mathcal{G})$ can be both viewed as some basic type definition, and ground dependency, both in Def and Pos, can be interpreted as directional type inference. The key point here is that the concrete domain is some lattice of order-ideals, i.e. sets of objects closed under instantiation, which are non-Boolean cHa. A different concrete domain would change completely the interpretation of directionality. Consider the Boolean concrete domain $\wp(Sub)$. If $Var = \{x, y\}$ we have the following domain $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$, which differs from the analogous completion Def on the non-Boolean lattice $\wp^{\downarrow}(Sub)$.



Def $\qquad\qquad$ $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$

While $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$ is no longer able to express ground dependencies, it is now able to express negative groundness information. This because intuitionistic implication is replaced by classical one. This is a consequence of the peculiar nature of the concrete (collecting) domain, which is a Boolean lattice. In this case, the relative pseudo-complements introduced by the Heyting completion provide negation of properties. This because, in Boolean lattices, $a \to b = \neg a \vee b$, for instance $\neg x \vee y = x \to x \wedge y = \overline{\gamma_{Sub}(x)} \cup \gamma_{Sub}(x \wedge y)$, while this is not the case in complete Heyting algebras. For instance, the empty substitution belongs to $\neg x \vee y$, but it provides no groundness information. Clearly, because the set-theoretic complement of a set closed under instantiation is not, in general, closed under instantiation, then $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$ is not here an abstraction of $\wp^{\downarrow}(Sub)$. In this sense, the analysis obtained from $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$ cannot be regarded as type inference.

8 Implicational domain equations

In this section we consider the idempotent extension of Heyting completion. Given an abstract domain A, we are interested in the least abstract domain X which is more concrete than A and it is closed under $\xrightarrow{\curlywedge}$, i.e. the least (most abstract) solution (if any) of the implicational domain equation

$$X = A \sqcap (X \xrightarrow{\curlywedge} X) \tag{1}$$

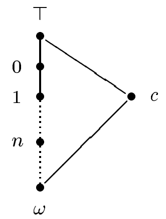
We consider the algebra $\langle uco(L), \sqcap, \curlyvee, \stackrel{\curlywedge}{\longrightarrow}\rangle$, involving conjunction of domains by reduced product, disjunctive completion and Heyting completion, which are all monotonic operators on domains. Note that the least fixpoint of $\lambda X.\ A \sqcap (X \stackrel{\curlywedge}{\longrightarrow} X)$ is L, which is the identical closure $\lambda x.x$. We are interested in the greatest fixpoint of $\lambda X.\ A \sqcap (X \stackrel{\curlywedge}{\longrightarrow} X)$ in $uco(L)$, which exists and it is the domain corresponding to the least closure ordinal of the above operator, in the following family of domains: $B_1 = A$, $B_{\alpha+1} = A \sqcap (B_\alpha \stackrel{\curlywedge}{\longrightarrow} B_\alpha) = B_\alpha \stackrel{\curlywedge}{\longrightarrow} B_\alpha$ and $B_\alpha = \sqcap_{\gamma<\alpha} B_\gamma$ if α is a limit ordinal.

Theorem 18. *Let L be a cHa and $X \sqsubseteq A$.*

1. *X is a solution of (1) iff it is a subalgebra of L with respect to the relative pseudo-complement.*
2. *X is a complete Heyting subalgebra of L iff it is a solution of (1) and it is disjunctive.*

It is worth noting that the operator associated with the equation (1), $\lambda X.\ A \sqcap (X \stackrel{\curlywedge}{\longrightarrow} X)$ is not, in general, co-continuous in $uco(L)$, as shown by the following example. Hence ω may not in general be its least closure ordinal.

Example 19. Consider the following complete lattice $L = \mathbb{N} \cup \{\omega, \top, c\}$ depicted below, and the family of its abstractions $A_i \in uco(L)$, such that for any $i \in \mathbb{N}$, $A_i = \{\, n \in \mathbb{N} \mid n \leq i \,\} \cup \{\top\}$.



$\{A_i\}_{i\in\mathbb{N}}$ is a chain, and for any $i \in \mathbb{N}$, $A_i = A_i \stackrel{\curlywedge}{\longrightarrow} A_i$, but $\sqcap_{i\in\mathbb{N}} A_i = \mathbb{N} \cup \{\omega\}$ is not dependency closed, since $c = 0 \to \omega \in (\sqcap_{i\in\mathbb{N}} A_i) \stackrel{\curlywedge}{\longrightarrow} (\sqcap_{i\in\mathbb{N}} A_i)$. Note that L above is not a cHa. However, unfortunately, it is not co-continuous even if the concrete domain is a Boolean algebra. It is sufficient to consider $L = \langle \wp(\omega), \omega, \emptyset, \cap, \cup, \subseteq \rangle$, and the chain of domains $A_i = \curlywedge(\cup_{1\leq j\leq i}\{\, \emptyset, \omega, \omega \setminus \{\, i \,\}, \{\, i \,\} \,\}) \in uco(\wp(\omega))$, for $i \geq 1$. Note that $\sqcap_{i\geq 1} A_i = \curlywedge(\cup_{i\geq 1} A_i)$ and $\{\, 0 \,\}, \emptyset \in \sqcap_{i\geq 1} A_i$ but, being $\cap$-irreducible, $\{\, 0 \,\} \to \emptyset = \omega \setminus \{\, 0 \,\} \notin \sqcap_{i\geq 1}(A_i \stackrel{\curlywedge}{\longrightarrow} A_i)$.

Clearly, co-continuity is ensured when L is a finite domain, because $uco(L)$ is finite too, or when $uco(L)$ satisfies the descending chain condition, i.e. it does not contains infinite descending chains. Because $uco(L)$ is a domain of functions, both of these conditions are far too restrictive for a concrete domain for semantics and analysis. However, co-continuity is maintained when the descending chain condition holds on the concrete domain L, as proved by the next result. Note that the operator we want to prove co-continuous is defined on $uco(L)$ and our condition, in general, does not imply that $uco(L)$ enjoys the descending chain property.

Theorem 20. *If L enjoys the descending chain property, the operator $\lambda X.\, A \sqcap (X \xrightarrow{\curlywedge} X)$ is co-continuous in $uco(L)$.*

Next result characterizes a generic solution, and the most abstract solution to the equation (1), under the hypothesis that L is a collecting domain.

Theorem 21. *Let $L = \wp(S)$ for some set S. $A = A \xrightarrow{\curlywedge} A$ iff A is disjunctive and complemented, viz. $A = \curlyvee(A)$ and $A = A \sqcap (A \xrightarrow{\curlywedge} \{\top_L, \bot_A\})$. Moreover $\curlyvee(A \sqcap (A \xrightarrow{\curlywedge} \{\top_L, \bot_A\}))$ is the most abstract solution domain for* (1).

9 Conclusions and future work

A logical interpretation for the basic operations for domain refinement gives rise to an algebra, where domains, and therefore analyses, can be manipulated in a logical framework. Hence, analyses can be specified in equational form, as domains in denotational semantics. Domains are no longer specified in terms of their elements, but in terms of the property they have to meet. Heyting completion is, in this construction, the basic step to build relational analyses. The logic interpretation of domain refinements has another interesting application. Hankin and Le Metayer ([20]) proved that some properties of data-structures during program execution can be inferred by a simple type system, in a kind of *proof theoretic* approach to program analysis. We believe that there is a link between systematic refinements of domains and corresponding refinements of proof systems for data-flow analysis, i.e., there is a proof-theoretic counterpart of our Heyting completion.

References

1. A. Aiken and T.K. Lakshman. Directional type checking of logic programs. In *Proc. 1^{st} Static Analysis Symp. SAS'94*, LNCS 864, pp. 43–60. Springer, 1994.
2. K. R. Apt. Introduction to logic programming. In *Handbook of Theoretical Computer Science*, Vol. B, pp. 495–574. Elsevier, 1990.
3. T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Boolean functions for dependency analysis: algebraic properties and efficient representation. In *Proc. 1^{st} Static Analysis Symp. SAS'94*, LNCS 864, pp. 266–280. Springer, 1994.
4. R. Barbuti, R. Giacobazzi, and G. Levi. A general framework for semantics-based bottom-up abstract interpretation of logic programs. *ACM TOPLAS*, 15(1):133–181, 1993.
5. G. Birkhoff. Lattice theory. In *AMS Colloquium Publication, third ed.* AMS Press, 1967.
6. F. Bronsard, T.K. Lakshman, and U.S. Reddy. A framework of directionality for proving termination of logic programs. In *Proc. 1992 Joint Int'l Conf. and Symp. Logic Prog.*, pp. 321–335. MIT Press, 1992.
7. M. Bruynooghe. A practical framework for the abstract interpretation of logic programs. *J. of Logic Programm.* 10:91–124, 1991.
8. M. Codish and B. Demoen. Deriving type dependencies for logic programs using multiple incarnations of *Prop*. In *Proc. 1^{st} Static Analysis Symp. SAS'94*, LNCS 864. Springer, 1994.

9. M. Codish, A. Mulkers, M. Bruynooghe, M. García de la Banda, and M.Hermenegildo. Improving abstract interpretations by combining domains. *ACM TOPLAS*, 17(1):28–44, 1995.
10. A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato. Complementation in abstract interpretation. *ACM TOPLAS* 19(1):7–47, 1997.
11. A. Cortesi, G. Filè, and W. Winsborough. *Prop* revisited: Propositional formulas as abstract domain for groundness analysis. In *Proc.* 6^{th} *IEEE LICS*, pp. 322–327. IEEE Comp. Soc. Press, 1991.
12. A. Cortesi, B. Le Charlier, and P. Van Hentenryck. Combinations of abstract domains for logic programming. In 21^{st} *ACM POPL*, pp. 227–239. ACM Press, 1994.
13. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In 4^{th} *ACM POPL*, pp. 238–252. ACM Press, 1977.
14. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In 6^{th} *ACM POPL*, pp. 269–282. ACM Press, 1979.
15. P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and per analysis of functional languages). In *Proc. 1994 Int'l Conf. on Comp. Lang. ICCL'94*, pp. 95–112. IEEE Comp. Soc. Press, 1994.
16. G. Filé, R. Giacobazzi, and F. Ranzato. A unifying view on abstract domain design. *ACM Comp. Surveys*, 28(2):333–336, 1996.
17. G. Filé and F. Ranzato. Improving abstract interpretations by systematic lifting to the powerset. In *Proc. 1994 Int'l Logic Prog. Symp.*, pp. 655–669. MIT Press, 1994.
18. R. Giacobazzi and F. Ranzato. Functional dependencies and Moore-set completions of abstract interpretations and semantics. In *Proc. 1995 Int'l Logic Prog. Symp.*, pp. 321–335. MIT Press, 1995.
19. R. Giacobazzi and F. Ranzato. Compositional optimization of disjunctive abstract interpretations. In *Proc. ESOP'96*, LNCS 1058, pp. 141–155. Springer, 1996.
20. C. Hankin and D. Le Metayer. Deriving algorithms from type inference systems: Applications to strictness analysis. In 21^{st} *ACM POPL*, pp. 202–212. ACM Press, 1994.
21. T.P. Jensen. Disjunctive strictness analysis. In 7^{th} *IEEE LICS*, pp. 174–185. IEEE Comp. Soc. Press, 1992.
22. N. D. Jones and H. Søndergaard. A Semantics-based Framework for the Abstract Interpretation of Prolog. In S. Abramsky and C. Hankin, eds., *Abstract Interpretation of Declarative Languages*, pp. 123–142. Ellis Horwood Ltd, 1987.
23. N.D. Jones and S.S. Muchnick. Complexity of flow analysis, inductive assertion synthesis and a language due to Dijkstra. In S. Muchnick and N.D. Jones, eds., *Program Flow analysis: Theory and Applications*, pp. 380–393. Prentice-Hall, 1981.
24. J. Morgado. Some results on the closure operators of partially ordered sets. *Port. Math.*, 19(2):101–139, 1960.
25. A. Mycroft and F. Nielson. Strong abstract interpretation using power domains. In *Proc.* 10^{th} *ICALP*, LNCS 154, pp. 536–547. Springer, 1983.
26. F. Nielson. Tensor products generalize the relational data flow analysis method. In *Proc.* 4^{th} *Hung. Comp. Science Conf.*, pp. 211–225, 1985.
27. K. Yi and W.L. Harrison. Automatic generation and management of interprocedural program analyses. In 20^{th} *ACM POPL*, pp. 246–259. ACM Press, 1993.