Logical optimality of groundness analysis

Francesca Scozzari

Dipartimento di Matematica, Università di Siena,
Via del Capitano 15, 53100 Siena, Italy

Dipartimento di Informatica, Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy
E-mail: scozzari@di.unipi.it
Ph.: +39-50-887248 Fax: +39-50-887226

Abstract. We study the relations among various abstract domains for groundness analyses of logic programming. We re-construct $\mathcal{P}os$ as a logical domain and prove that it is the optimal abstract domain for groundness analysis which can be constructed starting from the property of groundness by applying only logic operations.

Keywords: *Abstract interpretation, static analysis, logic programming, groundness, intuitionistic logic.*

1 Introduction

In the logic programming field, groundness is probably the most important instance of static analysis. Many domains have been proposed in order to study groundness of (pure) logic programs, from the very simple domain $\mathcal{G}$ by Jones and Søndergaard [13], to more complex ones, like $\mathcal{D}ef$ [2] and $\mathcal{P}os$ [13,15,4,2]. The latter is the most widely used, since it is able to characterize both pure groundness, i.e., if a variable is instantiated to a ground term, and groundness relations between different variables, i.e., whether the groundness of a variable depends on the groundness of other variables. In this paper, we show a different way of building the abstract domain $\mathcal{P}os$, which directly comes from the definition of groundness. From our construction, we derive many useful properties, such as a normal form for its elements and a result of optimality. Moreover, we answer to some open questions such as "why $\mathcal{P}os$ is considered optimal" from a computational point of view.

1.1 Motivations

$\mathcal{P}os$ is certainly a well studied domain for groundness analysis of logic programs [13,15,4,2]. In standard literature, the domain $\mathcal{P}os$ is built in 3 steps: First consider the set of (classic) propositional formulas built from a finite set Var and connectives $\wedge, \vee$ and $\Rightarrow$; second select only the positive formulas (which are true when all variables are set to true); third quotient w.r.t. classic logical equivalence. The domain so obtained is then related to the concrete one (sets of substitutions closed by instantiation) through a suitable concretization function, explicitly proving that it induces a Galois insertion. This method of constructing domains suffers from many drawbacks. The most important one is that the domain has to be "invented" in some way (a procedure which is usually not formally related to the property we analyze) and then has to be explicitly proved that the domain so created is an abstraction of the concrete one and is actually useful for the analysis. As logic programs compute substitutions and groundness is a property closed by instantiation, the most natural choice for the concrete domain is sets of substitutions closed by instantiation, denoted by $\wp^{\downarrow}(Sub)$.

Propositional formulas used in the construction of $\mathcal{P}os$ do not reflect the logic in the concrete domain. In order to understand this apparent asymmetry between $\mathcal{P}os$ and its natural concrete domain, we just need to make the following observations.

- From an algebraic point of view, $\mathcal{P}os$ is a boolean algebra (with connectives $\wedge, \vee$ and $\Rightarrow$) while $\wp^{\downarrow}(Sub)$ is not, therefore we can not inherit $\mathcal{P}os$ algebraic properties from the concrete domain.
- When we try to compare the logical operations on $\mathcal{P}os$ to the corresponding ones on $\wp^{\downarrow}(Sub)$, we find that only the meet ($\wedge$) on $\mathcal{P}os$ come from the meet on $\wp^{\downarrow}(Sub)$ (set intersection). In fact, the join ($\vee$) of $\mathcal{P}os$ is not the restriction of the concrete join (set union). For instance, as proved in [8], the concretization of the formula $x \vee (x \Leftrightarrow y)$ is not the union of the concretization of x and $x \Leftrightarrow y$. Moreover, in the concrete domain, we can not have a notion of (classic) implication, since it is not a boolean algebra. So there is no way to inherit the implication $\Rightarrow$ from a corresponding operation of $\wp^{\downarrow}(Sub)$.
- Finally, the last step in the construction of $\mathcal{P}os$ is the quotient w.r.t. classic logic equivalence. For the same motivation in the previous point, it is impossible to define such an equivalence on $\wp^{\downarrow}(Sub)$ (otherwise it would be a boolean algebra). Therefore, also this step in the construction of $\mathcal{P}os$ does not come from the properties of the concrete domain.

1.2 The intuition behind our reconstruction of $\mathcal{P}os$

The main problem of the previous construction is that $\wp^{\downarrow}(Sub)$ is not a boolean algebra. In particular, it does not allow us to define an operation of classic implication. Instead, $\wp^{\downarrow}(Sub)$ is rich enough to allow us to define a notion of intuitionistic implication (or relative pseudo-complement). The intuitionistic implication is a generalization of the classic one which leads to a weaker notion of complement called pseudo-complement. Moreover, intuitionistic implication is independent from the join operation $\vee$, i.e., $a \rightarrow b = \neg a \vee b$ does not hold. Thus $\wp^{\downarrow}(Sub)$ becomes an algebra with three operations $\cap$, $\cup$ and $\rightarrow$ which correspond precisely to the $\wedge$, $\vee$ and $\rightarrow$ of intuitionistic logic.

In this paper we show that $\mathcal{P}os$ can be constructed by using only the definition of groundness. The simplest domain defining the property of groundness is the least abstract domain containing Var, which is $\mathcal{G} = \wp(Var)$ by Jones and Søndergaard [13], where each variable v denotes the set of substitutions which ground v. In the first part of the paper, we show that $\mathcal{P}os$ is exactly the least abstract domain which contains all the (double) intuitionistic implications between elements of $\mathcal{G}$ ($\mathcal{P}os = (\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}) \xrightarrow{\curlywedge} \mathcal{G}$). This result generalizes a similar one on $\mathcal{D}ef$ in [11] which proves that $\mathcal{D}ef$ is the least abstract domain which contains all the intuitionistic implications between elements of $\mathcal{G}$, i.e., $\mathcal{D}ef = \mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$.

Our formalization of $\mathcal{P}os$ enjoys the following properties.

- $\mathcal{P}os$ is built by using only the definition of groundness (the domain $\mathcal{G}$) and the properties of the concrete domain. We need no more to use boolean algebra, positive formulas, nor to quotient the resulting set. The construction of the abstract domains $\mathcal{G}, \mathcal{D}ef, \mathcal{P}os$ now follows a unique, precise logic: The logic structure of the concrete domain. Moreover, we do not need anymore to prove that these domains are indeed abstractions of the concrete one, since it follows by construction. Also the relations among all the domains are now automatically derivable.
- The operations which characterize $\mathcal{P}os$ are now exactly the same of the concrete one: $\wedge$ and $\rightarrow$ in $\mathcal{P}os$ are the restrictions of $\cap$ and $\rightarrow$ in $\wp^{\downarrow}(Sub)$ and no other operation is considered in $\mathcal{P}os$.
- We immediately obtain a theorem of representation for $\mathcal{P}os$ which states that every element of $\mathcal{P}os$ is an implication between two elements of $\mathcal{D}ef$ or, by using the characterization of $\mathcal{D}ef$, every element of $\mathcal{P}os$ is an implication of implications between elements of $\mathcal{G}$, independently from the cardinality of Var.
- This result allows us to answer to some open questions on $\mathcal{P}os$. The join on $\mathcal{P}os$ differs from the join on $\wp^{\downarrow}(Sub)$ but, in some cases, they coincide.

  Why $\mathcal{P}os$ contains exactly those concrete joins?

  Our representation theorem states that the join on $\mathcal{P}os$ is precise if and only if it can be written as an intuitionistic implication.
- Since $\mathcal{P}os$ is now a sub-algebra of $\wp^{\downarrow}(Sub)$ w.r.t. the $\wedge$ and $\rightarrow$ operations, all the properties of $\mathcal{P}os$ are directly derivable from the properties of the concrete domain. In particular, since $\wp^{\downarrow}(Sub)$ is a model of intuitionistic propositional logic, it follows that also $\mathcal{P}os$ is. Therefore we gain from logic an axiomatization for $\mathcal{P}os$.

In the second part of the paper, we use our formalization of $\mathcal{P}os$ in order to understand why $\mathcal{P}os$ can be considered a "good" domain. To this end, we try to refine $\mathcal{P}os$. We wonder which is the domain which contains the implications between formulas in $\mathcal{P}os$ and find out an important closure property of $\mathcal{P}os$, i.e., that $\mathcal{P}os \xrightarrow{\curlywedge} \mathcal{P}os = \mathcal{P}os$. This implies that $\mathcal{P}os$ can not be further refined w.r.t. intuitionistic implication. Therefore $\mathcal{P}os$ is exactly the least abstract domain closed w.r.t. $\wedge$ and $\rightarrow$.

We then consider the disjunctive completion of $\mathcal{P}os$, $\curlyvee(\mathcal{P}os)$, which includes all unions of formulas in $\mathcal{P}os$. [8] proved that $\mathcal{P}os$ is strictly contained in $\curlyvee(\mathcal{P}os)$. We try to refine this result by allowing implications between disjunctive formulas and prove that it can not be further refined since $\curlyvee(\mathcal{P}os) \xrightarrow{\curlywedge} \curlyvee(\mathcal{P}os) = \curlyvee(\mathcal{P}os)$.

1.3 Related work

Many works have been devoted to the study of groundness analysis. The first ones concentrated much on the definition of groundness and basic properties [13,7], while the last ones studied different characterizations of various abstract domains. [15] proposed propositional formulas to represent groundness relations. Many authors followed this approach [4,2] and contributed to develop and study the domains $\mathcal{D}ef$ and $\mathcal{P}os$, while others focused on the abstract operations or slightly different characterizations [16,14].

All these authors share the same approach: They construct abstract domains independently from the property to be analyzed (i.e., from the semantics or concrete domain) and then prove some properties of the abstraction. Their attention is focused entirely on the representation of formulas in the abstract domains. This forces to work always up to isomorphism.

Our idea is to concentrate exclusively on the property of groundness. We re-construct the domain $\mathcal{P}os$ in a systematic way and show that it is possible to avoid *ad hoc* characterizations in the construction process of new domains. Our work is based on the Heyting completion refinement operator [11]. A first example in this direction has been presented in [11], where Heyting completion is used to construct the abstract domain $\mathcal{D}ef$. An attempt of building the domain $\mathcal{P}os$ starting from the disjunctive completion of the domain $\mathcal{G}$ (by Jones and Søndergaard [13]) is shown in [11]. Also in this case, the construction does not directly come from the property of groundness, but from a domain more complex than $\mathcal{G}$.

2 Preliminaries

Throughout the paper we assume familiarity with lattice theory (e.g. see [3,12]), abstract interpretation [5,6,17] and logic programming [1].

2.1 Notation and basic notions

Let A, B and C be sets. $A \setminus B$ denotes the set-theoretic difference between A and B, $A \subset B$ denotes proper inclusion and, if $X \subseteq A$, $\overline{X}$ is the set-theoretic complement of X. The powerset of A is denoted by $\wp(A)$. If A is a poset, we usually denote by $\leq_A$ the corresponding partial order. A complete lattice A with partial ordering $\leq_A$, least upper bound $\vee_A$ (join), greatest lower bound $\wedge_A$ (meet), least element $\bot_A$, and greatest element $\top_A$, is denoted by $\langle A, \leq_A, \vee_A, \wedge_A, \top_A, \bot_A \rangle$. Note that, being A complete, the join and meet operations are defined on $\wp(A)$. If A is (partially) ordered and $I \subseteq A$ then $\downarrow I = \{x \in A \mid \exists y \in I.\, x \leq_A y\}$. $\wp^{\downarrow}(A)$ 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. In the following we sometimes use Church's lambda notation for functions, so that a function

f will be denoted by $\lambda x.f(x)$. 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 *additive* if for any $C \subseteq A$, $f(\vee_A C) = \vee_B f(C)$.

2.2 Abstract interpretation and Galois connections

The standard Cousot and Cousot theory of abstract interpretation is based on the notion of Galois connection [5]. If C and A are posets and $\alpha : C \longmapsto A$, $\gamma : A \longmapsto C$ are monotonic functions, 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. by identifying in an equivalence class those objects in A having the same image (meaning) in C.

Let L be 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. $\forall x \in L.\ x \leq_L \rho(x)$) [17]. Each closure operator ρ is uniquely determined by the set of its fixpoints, which is its image $\rho(L)$. $\rho(L)$ 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.

$X \subseteq L$ 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. We denote by $\langle Moore(L), \sqsubseteq, \sqcap, \sqcup, \{\top\}, L \rangle$ the complete lattice of all Moore-families of L. The ordering $\sqsubseteq$ is the inverse of set inclusion ($\supseteq$), $\sqcap$ is the least Moore-family which contains set union, ($\lambda X, Y.\curlywedge(X \cup Y)$), and $\sqcup$ is set intersection ($\cap$).

The equivalence between G.i., closure operators and Moore-families is well known [3]. However, closure operators and Moore-families are often more practical and concise than G.i.'s to reason about abstract domains, being independent from representation choices for domain objects [6]. 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 $Moore(L)$. The order relation on $Moore(L)$ 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 Moore-families.

2.3 Logic programming

Let $\mathcal{V}$ be a (denumerable) set of variables. We fix a first-order language $\mathcal{L}$, with variables ranging in $\mathcal{V}$. $Term$ is the set of terms of $\mathcal{L}$. For any syntactic object s, $vars(s)$ denotes the set of its variables. A term t is ground if $vars(t) = \emptyset$.

The set of idempotent substitutions, i.e., finite mappings from $\mathcal{V}$ to terms in $\mathcal{L}$, is denoted by Sub. Substitutions are lifted to terms in the usual way. If α is a substitution, $dom(\alpha)$ denotes the set $\{ v \in \mathcal{V} \mid \alpha(v) \neq v. \}$, which is always finite by definition. By $\beta \circ \alpha$ we denote the substitution $\lambda v.\alpha(\beta(v))$. Objects in Sub are partially ordered by instantiation: $a \leq b$ iff $\exists \theta \in Sub.\ a = b\theta$.

2.4 Intuitionistic logic and Heyting algebras

Let L be a complete lattice and $a, b \in L$. The *pseudo-complement* (or intuitionistic implication) of a *relatively* to b, if it exists, is the unique element $a \to b \in L$ such that for any $x \in L$: $a \wedge_L x \leq_L b$ iff $x \leq_L a \to b$. Relative pseudo-complements, when they exist, are uniquely given by $a \to b = \bigvee_L \{ c \mid a \wedge_L c \leq_L b \}$. A complete lattice A is a *complete Heyting algebra* (cHa) if it is *relatively pseudo-complemented*, that is $a \to b$ exists for every $a, b \in A$. From a logical point of view, an Heyting algebra L is an algebra equipped with three operations $\wedge, \vee, \to$ and constants $\bot, \top$ which satisfies the following equations for every $a, b, c \in L$:

$$a \to a = \top \qquad\qquad a \wedge \bot = \bot \tag{2.1}$$

$$(a \to b) \wedge b = b \qquad\qquad a \wedge (a \to b) = a \wedge b \tag{2.2}$$

$$a \to (b \wedge c) = (a \to b) \wedge (a \to c) \qquad (a \vee b) \to c = (a \to c) \wedge (b \to c) \tag{2.3}$$

Meet, join and relative pseudo-complement of Heyting algebras precisely correspond to conjunction, disjunction and intuitionistic implication of intuitionistic logic (see Birkhoff [3]).

An example of cHa is the complete lattice $\langle \wp^{\downarrow}(Sub), \subseteq, \cup, \cap, Sub, \emptyset \rangle$. The logical operations $\vee$ and $\wedge$ correspond to $\cup$ and $\cap$ operations (set intersection and set union). Given $a, b \in \wp^{\downarrow}(Sub)$, the intuitionistic implication $a \to b = \bigcup \{ c \in \wp^{\downarrow}(Sub) \mid a \cap c \subseteq b \}$ is also given by ([3])

$$a \to b = \{ \theta \in Sub \mid \forall \delta \leq \theta \text{ if } \delta \in a \text{ then } \delta \in b \}.$$

Example 1. Let $x, y, w, z \in \mathcal{V}$ be variables, $X = \{ \theta \in Sub \mid vars(\theta(x)) = \emptyset \}$ and $Y = \{ \theta \in Sub \mid vars(\theta(y)) = \emptyset \}$.

$$\begin{aligned} X \to Y &= \{ \theta \in Sub \mid \forall \delta \leq \theta \text{ if } \delta \in X \text{ then } \delta \in Y \} \\ &= \{ \theta \in Sub \mid \forall \delta \leq \theta \text{ if } vars(\delta(x)) = \emptyset \text{ then } vars(\delta(y)) = \emptyset \}. \end{aligned}$$

For instance, given a binary functor f, the substitution $\{ x \leftarrow f(y, w) \}$ belongs to $X \to Y$, while $\{ y \leftarrow f(x, w) \}$ does not. ■

3 Domains refinements

3.1 Reduced product

The *reduced product* [6] of abstract domains corresponds to the meet operation ($\sqcap$) of closure operators. Given two abstract domains X, Y, the reduced product of X and Y, denoted by $X \sqcap Y$, is the least abstract domain which includes both X and Y, i.e., $X \sqcap Y = \curlywedge(X \cup Y)$.

3.2 Disjunctive completion

The *disjunctive completion* [6] of an abstract domain A is 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. [6,10]):

$$\curlyvee(A) = \sqcup\{\, X \in Moore(L) \mid X \sqsubseteq A \text{ and } X \text{ is additive} \,\}.$$

Proposition 2. *If L is a cHa and A is finite, $\curlyvee(A) = \{\, \vee_L X \mid X \subseteq A \,\}$.*

3.3 Heyting completion

Heyting completion [11] is a refinement operator which has been recently introduced in order to logically interpret the Cousot and Cousot's reduced power refinement [6]. In this section we recall the definition and some basic results on Heyting completion refinement [11]. The aim of this refinement is to enhance domains to represent relational or negative information. The idea is to enrich an abstract domain by adding all the relative pseudo-complements (or intuitionistic implications) built from every pair of elements of the given domain. From a logical point of view, the new domain is the collection of intuitionistic formulas built from the $\wedge$ and $\rightarrow$ connectives, without nested implication.

For the sake of simplicity, we recall the definition only in case the concrete domain L is a cHa. Given two abstract domains A and B, the Heyting completion of A wrt B (denoted by $A \stackrel{\curlywedge}{\longrightarrow} B$) is precisely captured by the least Moore family containing all the relative pseudo-complements (without nesting):

$$A \stackrel{\curlywedge}{\longrightarrow} B = \curlywedge(\{\, a \rightarrow b \in L \mid a \in A, b \in B \,\}).$$

The Heyting completion refinement is reductive on the second argument ($B \stackrel{\curlywedge}{\longrightarrow} A \sqsubseteq A$) and argumentwise monotonic (if $A \sqsubseteq B$ and $C \sqsubseteq D$ then $A \stackrel{\curlywedge}{\longrightarrow} C \sqsubseteq B \stackrel{\curlywedge}{\longrightarrow} D$).

We recall some basic algebraic properties of Heyting completion with respect to other domain operations. In particular, we consider reduced product ($\sqcap$) and disjunctive completion ($\curlyvee$) of abstract domains. In the following of the paper, $A, B, C \in Moore(L)$.

Proposition 3. *Let L be a cHa.*

1. $A \xrightarrow{\curlywedge} (B \sqcap C) = (A \xrightarrow{\curlywedge} B) \sqcap (A \xrightarrow{\curlywedge} C)$
2. $(A \sqcap B) \xrightarrow{\curlywedge} C = A \xrightarrow{\curlywedge} (B \xrightarrow{\curlywedge} C)$
3. $\curlywedge(A) \xrightarrow{\curlywedge} B = A \xrightarrow{\curlywedge} B$ *if A is finite.*

4 Groundness: $\mathcal{D}ef$ and $\mathcal{P}os$

Many domains have been proposed in order to study groundness of pure logic programs. If Var is the (finite) set of variables of interest, the simplest one is $\mathcal{G} = \wp(Var)$, due to Jones and Søndergaard [13], where each $V \subseteq Var$ denotes the set of substitutions which ground every variable in V. This domain is certainly the most intuitive one and represents exactly the property we want to analyze, since it is built from the definition of groundness itself. Unfortunately, $\mathcal{G}$ is not very useful for groundness analysis, since it fails in capturing the groundness relations between different variables, i.e., the domain is able to represent the "result" of a groundness analysis, but not to be used for "computing"' the analysis itself.

Other domains, based on (classic) propositional logic, have been proposed in order to enrich $\mathcal{G}$. $\mathcal{P}os$ is the most widely used domain for groundness analysis [13,15,4,2]. $\mathcal{P}os$ 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. $\mathcal{P}os$ is the set of (classic) propositional formulas built from Var, by using the connectives $\wedge, \vee, \Rightarrow$, quotiented w.r.t. (classic) logical equivalence (KL). If we denote by $Form(Var, \circ)$ the set of formulas built from Var by using the connective $\circ$, then $\mathcal{P}os$ can be defined in different ways [4,2]:

$$\mathcal{P}os = Form(Var, \wedge, \vee, \Rightarrow)_{/KL} = Form(Var, \wedge, \Rightarrow)_{/KL}$$

The domain $\mathcal{D}ef$ [2] is built by considering all the formulas whose models are closed under intersection and taking the quotient of this set w.r.t. (classic) logical equivalence. $\mathcal{D}ef$ has been also characterized as the set of formulas which are conjunctions of definite clauses, always quotiented w.r.t. (classic) logical equivalence.

$$\mathcal{D}ef = \{ \bigwedge_{i \in I} (\bigwedge_{j \in J} x_i^j \Rightarrow y^j) \mid \forall i \in I, j \in J \; x_i^j, y^j \in Var \}_{/KL}$$

For the sake of notation, note that also the domain $\mathcal{G} = \wp(Var)$ can be defined as a domain of formulas by using the connective $\wedge$ to represent subsets:

$$\mathcal{G} = Form(Var, \wedge)_{/KL}$$

We can relate the three domains to the concrete one by using the same concretization function. The concrete domain considered in all cases is $\wp^{\downarrow}(Sub)$ (which is a cHa). 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 [2]. For the sake of simplicity we write $\theta \,\texttt{grounds}\, \phi$ to denote $\{ x \in Var \mid vars(\theta(x)) = \emptyset \} \models \phi$, for $\theta \in Sub$. The concretization function is [4]:

$$\gamma_{Sub}(\phi) = \{ \theta \in Sub \mid \forall \sigma \leq \theta \ \sigma \,\texttt{grounds}\, \phi \}$$

As usual, we shall abuse of notation and call $\mathcal{G}$, $\mathcal{D}ef$ and $\mathcal{P}os$ the corresponding subsets of $\wp^{\downarrow}(Sub)$: $\gamma_{Sub}(\mathcal{G})$, $\gamma_{Sub}(\mathcal{D}ef)$ and $\gamma_{Sub}(\mathcal{P}os)$.

5 Implicational groundness analysis

We start listing some well known facts about $\mathcal{G}$, $\mathcal{D}ef$ and $\mathcal{P}os$. A first result on the reconstruction of abstract domains as implicational ones was proved in [11]. The authors proved that $\mathcal{D}ef$ is precisely the Heyting completion of $\mathcal{G}$.

$$\mathcal{D}ef = \mathcal{G} \xrightarrow{\curlywedge} \mathcal{G} \tag{5.1}$$

In this section we prove that also $\mathcal{P}os$ is an implicational domain which depends only on the simple domain $\mathcal{G}$. We shall prove that $\mathcal{P}os = (\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}) \xrightarrow{\curlywedge} \mathcal{G}$. Therefore by 5.1, $\mathcal{P}os = \mathcal{D}ef \xrightarrow{\curlywedge} \mathcal{G}$. In order to prove it, we need to better understand the relations among the logical operations in $\mathcal{P}os$ and in $\wp^{\downarrow}(Sub)$. Since $\mathcal{P}os$ is a Moore family on $\wp^{\downarrow}(Sub)$, the meet operation $(\wedge)$ on $\mathcal{P}os$ must be the restriction of the meet operation on $\wp^{\downarrow}(Sub)$, which is set intersection. It follows that, for every formula $\phi_i \in \mathcal{P}os$,

$$\bigwedge_{i \in I} \gamma_{Sub}(\phi_i) = \gamma_{Sub}(\bigwedge_{i \in I} \phi_i) \tag{5.2}$$

It is well known that the join operation $(\vee)$ of $\mathcal{P}os$ is not the restriction of the concrete one on $\wp^{\downarrow}(Sub)$, which is set union [8]. But the isomorphism holds if we consider variables only.

Lemma 4.

$$\bigvee_{j \in J} \gamma_{Sub}(y_j) = \gamma_{Sub}(\bigvee_{j \in J} y_j)$$

A similar result holds also for the implication. In general, the classic implication $\Rightarrow$ does not coincide with the intuitionistic one. If we consider implications between a conjunction of variables and a disjunction of variables only, then they are isomorphic.

Lemma 5.

$$\gamma_{Sub}(\bigwedge_{i \in I} x_i \Rightarrow \bigvee_{j \in J} y_j) = \gamma_{Sub}(\bigwedge_{i \in I} x_i) \rightarrow \gamma_{Sub}(\bigvee_{j \in J} y_j)$$

The idea is to find a normal form for elements in $\mathcal{P}os$ which allows disjunction only between variables and (classic) implication only between a conjunction and a disjunction of variables. It is well known that every (classic) formula can be put in conjunctive normal form, i.e., it can be written as conjunction of disjunctions. Moreover, every disjunction is a clause, therefore we can write it as a unique implication, by putting all the negated variables on the left and the positive ones on the right.

Lemma 6. *In $\mathcal{P}os$, every formula ϕ is equivalent to a formula of the form: $\bigwedge_{k\in K}(\bigwedge_{i\in I} x_{i,k} \Rightarrow \bigvee_{j\in J} y_{j,k})$.*

The next step is to transform every formula in $\mathcal{P}os$ into a formula of $\wp^{\downarrow}(Sub)$, which is not a boolean algebra, as next example shows.

Example 7. In a boolean algebra $(a \wedge b) \Rightarrow c = (a \Rightarrow c) \vee (b \Rightarrow c)$ holds. Let $X = \{\, \theta \in Sub \mid vars(\theta(x)) = \emptyset \,\}$, $Y = \{\, \theta \in Sub \mid vars(\theta(y)) = \emptyset \,\}$ and $Z = \{\, \theta \in Sub \mid vars(\theta(z)) = \emptyset \,\}$, where $x, y, z \in Var$. Consider in $\wp^{\downarrow}(Sub)$ the formula $(X \wedge Y) \to Z = \{\, \theta \in Sub \mid \forall \delta \leq \theta \text{ if } \delta \in X \wedge Y \text{ then } \delta \in Z \,\}$. We show that $(X \to Z) \vee (Y \to Z)$ is strictly contained in $(X \wedge Y) \to Z$. In fact, the substitution $\alpha = \{\, z \leftarrow f(x, y) \,\}$ belongs to $(X \wedge Y) \to Z$ but not to $(X \to Z)$ nor to $(Y \to Z)$.

We remind that $\wp^{\downarrow}(Sub)$ is a cHa [3], i.e., it is an algebra equipped with three operations: Meet, join and intuitionistic implication. Meet and join on $\wp^{\downarrow}(Sub)$ are set intersection and set union, while intuitionistic implication is given by $a \to b = \{\, \theta \in Sub \mid \forall \delta \leq \theta \text{ if } \delta \in a \text{ then } \delta \in b \,\}$ [3]. This allows us to deal with formulas in $\wp^{\downarrow}(Sub)$ by simply using its operations as connectives.

The previous representation lemma, together with the isomorphism relations shown in Lemmata 4 and 5, allows us to directly transform every formula of $\mathcal{P}os$ into a formula of $\wp^{\downarrow}(Sub)$.

Example 8. Let $x \vee (x \Leftrightarrow y)$ be a formula in $\mathcal{P}os$. We look for its concretization $\gamma_{Sub}(x \vee (x \Leftrightarrow y))$. First we transform $x \vee (x \Leftrightarrow y)$ in normal form, which always exists by Lemma 6.

$$
\begin{aligned}
&x \vee (x \Leftrightarrow y) = x \vee ((x \Rightarrow y) \wedge (y \Rightarrow x)) = \\
&= (x \vee (x \Rightarrow y)) \wedge (x \vee (y \Rightarrow x)) = (x \vee \neg x \vee y) \wedge (x \vee \neg y \vee x) = \\
&= x \vee \neg y \vee x = y \Rightarrow x
\end{aligned}
$$

Next we transform $y \Rightarrow x$ by using Lemma 5 and obtain $\gamma_{Sub}(y \Rightarrow x) = \gamma_{Sub}(y) \to \gamma_{Sub}(x)$, which is a formula in $\wp^{\downarrow}(Sub)$.

As shown by the above example, we have a constructive method to transform formulas in $\mathcal{P}os$ into formulas in $\wp^{\downarrow}(Sub)$. Thus we obtain an image of $\mathcal{P}os$ on the concrete domain. Let $\phi = \bigwedge_{k\in K}(\bigwedge_{i\in I} x_{i,k} \Rightarrow \bigvee_{j\in J} y_{j,k})$ be a generic element of $\mathcal{P}os$ (Lemma 6). The concretization of ϕ, $\gamma_{Sub}(\phi)$ is precisely $\bigwedge_{k\in K}(\bigwedge_{i\in I} \gamma_{Sub}(x_{i,k}) \to \bigvee_{j\in J} \gamma_{Sub}(y_{j,k}))$. Therefore, we can compute the concrete image of $\mathcal{P}os$, as shown by the following theorem.

Theorem 9.

$$\gamma_{Sub}(\mathcal{P}os) = \{ \bigwedge_{k\in K} (\bigwedge_{i\in I} x_{i,k} \rightarrow \bigvee_{j\in J} y_{j,k}) \mid x_{i,k}, y_{j,k} \in \gamma_{Sub}(Var) \}.$$ [1]

Our aim is to describe the domain $\mathcal{P}os$ by using the Heyting completion refinement only. Therefore, we look for a normal form for concrete objects which uses meets and intuitionistic implications only, that is a normal form which does not use the disjunctive operation. Note that, in the formulas on the concrete domain, disjunctions are computed between variables only. The last step of the construction is then to find a representation in $\wp^{\downarrow}(Sub)$ for unions of variables in terms of intuitionistic implications. The next lemma shows that a (concrete) disjunction is always equivalent to a (double) implication.

Lemma 10. *Let* $X = \bigwedge_{i\in I} x_i$ *and* $Y = \bigvee_{j\in J} y_j$*, where* $x_i, y_j \in \wp^{\downarrow}(Sub)$*. Then* $(X \rightarrow Y) \rightarrow Y = X \vee Y$ *holds.*

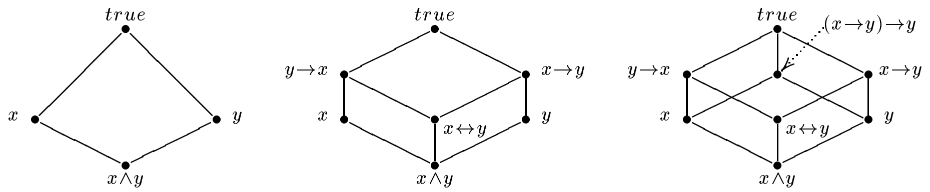
Therefore we obtain a transformation of formulas in $\mathcal{P}os$ into formulas in $\wp^{\downarrow}(Sub)$ which use $\wedge$ and $\rightarrow$ only. In particular, note that the $\rightarrow$ is used with at most two levels of nesting. This suggests to construct $\mathcal{P}os$ as a double implication.

Theorem 11. $\mathcal{P}os = (\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}) \xrightarrow{\curlywedge} \mathcal{G}$

By 5.1 and properties of Heyting completion (Proposition 3.2 and monotonicity), we immediately obtain:

Corollary 12. $\mathcal{P}os = \mathcal{D}ef \xrightarrow{\curlywedge} \mathcal{G} = \mathcal{D}ef \xrightarrow{\curlywedge} \mathcal{D}ef$

For $Var = \{x, y\}$, the domains $\mathcal{G}$, $\mathcal{D}ef$ and $\mathcal{P}os$ are depicted below.



$\mathcal{G}$ $\qquad$ $\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$ $\qquad$ $(\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}) \xrightarrow{\curlywedge} \mathcal{G}$

The first important consequence of Theorem 11 is that the domain $\mathcal{P}os$ is constructed by using only the definition of groundness ($\mathcal{G}$) and the logical properties of the concrete domain. We do not need to "invent" the domain, to prove that it is actually an abstraction of $\wp^{\downarrow}(Sub)$, nor to prove that it refines $\mathcal{G}$, since all these properties hold by construction. In our framework, $\mathcal{P}os$ arises as the natural refinement of $\mathcal{G}$ and $\mathcal{D}ef$.

Another consequence is the normal form for elements in $\gamma_{Sub}(\mathcal{P}os)$. This result allows us to get rid of $\mathcal{P}os$ and deal directly with $\gamma_{Sub}(\mathcal{P}os)$. In fact, when we use a formula of $\mathcal{P}os$, actually we use an equivalence class w.r.t. (classic) logic equivalence (KL). In our formalization, we do not need to use equivalence

[1] Note that $x_{i,k}, y_{j,k}$ denote sets of substitutions.

classes any longer. Moreover, the "form" of concrete formulas is indeed very natural and preserves the intuitive meaning of abstract formulas. Finally, note that our normal form states that every formula in $\gamma_{Sub}(\mathcal{P}os)$ can be written by using only two levels of nesting for the implication and no disjunctions. This yields a precise upper bound to the length of the formula. It is worth noticing that both the normal form and the upper bound are independent from the cardinality of Var.

6 Optimality of $\mathcal{P}$os

An abstract domain is a collection of points selected from the concrete domain. Some points are used to represent the result of the analysis, while the other ones are used only during the computation. For instance, in $\mathcal{P}os$ the points which represent the final result of groundness analysis are those in $\mathcal{G}$ only, since the other ones do not provide groundness information. From equations 2.1, 2.2 and 2.3, it is obvious that only axioms 2.2 produce a result which belongs to $\mathcal{G}$ (i.e., formulas with conjunctions only). This suggests that an abstract domain, to be optimal, should contain all and only those formulas which have an implicational form, since implications only can be reduced to formulas in $\mathcal{G}$. Therefore, to obtain an optimal domain, we have to include all (and only) the implications in the abstract domain. This concept is precisely captured by the notion of implicational domain equation.

An abstract domain X is *optimal w.r.t. a given domain* A (and the operation $\wedge$) if it is the least (most abstract) solution of the implicational domain equation

$$X = A \sqcap (X \xrightarrow{\curlywedge} X). \tag{6.1}$$

The solution, which always exists, is the most abstract domain X which is more concrete than A and is closed under $\xrightarrow{\curlywedge}$. Moreover, it turns out to be the most refined abstract domain we can obtain by using Heyting completion refinement.

6.1 Solution to the equation

In the previous section, starting from the result $\mathcal{D}ef = \mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}$, we have shown that $\mathcal{P}os = (\mathcal{G} \xrightarrow{\curlywedge} \mathcal{G}) \xrightarrow{\curlywedge} \mathcal{G}$. A question which naturally arises is: what is the domain $\mathcal{P}os \xrightarrow{\curlywedge} \mathcal{P}os$? By Proposition 3.2 we know that $\mathcal{P}os \xrightarrow{\curlywedge} \mathcal{P}os = \mathcal{P}os \xrightarrow{\curlywedge} \mathcal{G}$. More generally, we wonder which is the most abstract domain which is more precise than $\mathcal{G}$ and closed w.r.t. Heyting completion, i.e., which is the solution to the equation: $X = \mathcal{G} \sqcap (X \xrightarrow{\curlywedge} X)$.

The next theorem answers both questions, since $\mathcal{P}os$ is already closed w.r.t. Heyting completion. Therefore, it is precisely the least (most abstract) solution to the equation.

Theorem 13. $\mathcal{P}os = \mathcal{P}os \xrightarrow{\curlywedge} \mathcal{P}os$

The theorem states that we can not further refine $\mathcal{P}os$ with Heyting completion refinement. This result comes from properties of substitutions, since, in the concrete domain $\wp^{\downarrow}(Sub)$, join and implication operations are not completely independent from each other (see Lemma 10).

Moreover this theorem yields a representation result for elements in $\mathcal{P}os$. It precisely states that an element of $\wp^{\downarrow}(Sub)$ belongs to $\mathcal{P}os$ if and only if it can be written by using meets and implications only. This is much stronger than the previous one since it completely characterizes the image of $\mathcal{P}os$. Differently from previous characterizations of $\mathcal{P}os$ [2,4], all these results hold on the concrete domain, that is we can directly deal with formulas without using any isomorphisms. Therefore, the concretization function γ_{Sub} becomes, in our construction, the identity function.

This result allows us to answer the question "why $\mathcal{P}os$ is considered optimal without being disjunctive", also from an intuitive point of view. From the previous characterization, we know that a disjunctive formula belongs to $\mathcal{P}os$ if and only if it can be put in implicational form. From a logic point of view, the elements useful for the analysis are implications only (as they can be reduced by *modus ponens*). Since a good domain should contain all and only those joins which are indeed useful, we would include those joins which can be reduced only, which are exactly the joins that $\mathcal{P}os$ contains. This explains the result in [9] that $\mathcal{P}os$ is complete (or precise) w.r.t. $\curlyvee(\mathcal{P}os)$, that is $\mathcal{P}os$ is the "optimal" domain for groundness analysis.

6.2 On disjunctive completion

We proved that we can not further refine $\mathcal{P}os$ by Heyting completion and [9] showed that it is pointless to use $\curlyvee(\mathcal{P}os)$ instead of $\mathcal{P}os$.

The idea is to refine $\mathcal{P}os$ by disjunctive completion and then to refine it by Heyting completion, i.e., to find the solution of the equation $X = \curlyvee(\mathcal{P}os) \sqcap (X \xrightarrow{\curlywedge} X)$. The next theorem claims that also $\curlyvee(\mathcal{P}os)$ is closed w.r.t. Heyting completion.

Theorem 14. $\curlyvee(\mathcal{P}os) = \curlyvee(\mathcal{P}os) \xrightarrow{\curlywedge} \curlyvee(\mathcal{P}os)$

Therefore, $\curlyvee(\mathcal{P}os)$ is the "biggest" domain we can obtain by disjunctive and Heyting completion refinements and, in view of the result of completeness of $\mathcal{P}os$ w.r.t. $\curlyvee(\mathcal{P}os)$, we have proved that $\mathcal{P}os$ is definitely the best domain for groundness analysis.

7 Conclusions

In this paper we have reconstructed $\mathcal{P}os$ by using the properties of the concrete domain only. With our formalization, we automatically obtain the properties of $\mathcal{P}os$ by construction. We show that we can get rid of the construction of $\mathcal{P}os$ with positive formulas and equivalence classes and use directly the operations which naturally arise from the concrete domain. Moreover, we show a result of optimality for $\mathcal{P}os$ by proving that it contains exactly all and only the elements really useful to the analysis. This result is very surprising if we consider that the domain $\mathcal{P}os$ was built without following any formal notion of optimality.

The main feature of our construction is that it can easily be applied to other kind of analyses of logic programs, since it depends only on the properties of the concrete domain. We trust that (many) other analyses can be naturally formalized in this way, in particular analyses of properties closed under instantiation, such as type analysis, where the semantics (concrete domain) is always $\wp^{\downarrow}(Sub)$.

References

1. K. R. Apt. Introduction to logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 495–574. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.
2. T. Armstrong, K. Marriott, P. Schachte, and H. Søndergaard. Boolean functions for dependency analysis: algebraic properties and efficient representation. In B. Le Charlier, editor, *Proc. of the 1st Int'l Static Analysis Symposium (SAS '94)*, volume 864 of *LNCS*, pages 266–280. Springer-Verlag, 1994.
3. G. Birkhoff. Lattice theory. In *AMS Colloquium Publication, third ed.* AMS Press, 1967.
4. A. Cortesi, G. Filè, and W. Winsborough. *Prop* revisited: Propositional formula as abstract domain for groundness analysis. In *Proc. Sixth IEEE Symp. on Logic In Computer Science*, pages 322–327. IEEE Computer Society Press, 1991.
5. P. Cousot and R. Cousot. 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)*, pages 238–252. ACM Press, 1977.
6. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symposium on Principles of Programming Languages (POPL '79)*, pages 269–282. ACM Press, 1979.
7. S.K. Debray. Static inference of modes and data dependencies in logic programs. *ACM TOPLAS*, 11:418–450, 1989.
8. G. Filé and F. Ranzato. Improving abstract interpretations by systematic lifting to the powerset. In M. Bruynooghe, editor, *Proc. of the 1994 Int'l Logic Programming Symposium (ILPS '94)*, pages 655–669. The MIT Press, 1994.
9. G. Filé and F. Ranzato. The Powerset Operator on Abstract Interpretations. To appear in Theoretical Computer Science, 1997.
10. R. Giacobazzi and F. Ranzato. Compositional optimization of disjunctive abstract interpretations. In H.R. Nielson, editor, *Proc. of the 1996 European Symposium on Programming*, volume 1058 of *LNCS*, pages 141–155. Springer-Verlag, Berlin, 1996.
11. R. Giacobazzi and F. Scozzari. Intuitionistic implication in abstract interpretation. In *Proceedings of Ninth International Symposium on Programming Languages, Implementations, Logics and Programs PLILP'97*, Lecture Notes in Computer Science. Springer-Verlag, 1997. To appear.
12. G. Gierz, K.H. Hofmann, K. Keimel, J.D. Lawson, M. Mislove, , and D.S. Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, 1980.

13. N. D. Jones and H. Søndergaard. A Semantics-based Framework for the Abstract Interpretation of Prolog. In S. Abramsky and C. Hankin, editors, *Abstract Interpretation of Declarative Languages*, pages 123–142. Ellis Horwood Ltd, 1987.
14. N.D. Jones, K. Marriot, and H. Søndergaard. Denotational abstract interpretation of logic programs. *ACM TOPLAS*, 16:607–648, 1994.
15. K. Marriott and H. Søndergaard. Abstract interpretation of logic programs: the denotational approach. In A. Bossi, editor, *Proc. GULP '90*, pages 399–425. Padova, 1990.
16. K. Marriott and H. Søndergaard. Precise and efficient groundness analysis for logic programs. *ACM Letters on Programming Languages and Systems*, 2(1-4):181–196, 1993.
17. J. Morgado. Some results on the closure operators of partially ordered sets. *PortugaliæMathematica*, 19(2):101–139, 1960.