Set Domains for Structural Properties of Terms

Agostino Dovier*
Univ. di Verona, Ist. Policattedra, Fac. Scienze.
Strada Le Grazie 3, 37134 VERONA (I).
dovier@biotech.sci.univr.it

Alberto Policriti
Univ. di Udine, Dip. di Matematica e Informatica.
Via delle Scienze 206, 33100 UDINE (I).
policrit@dimi.uniud.it

Abstract

Usefulness of set domains for abstract interpretation in logic programming has been pointed out by many authors. Such domains are based on 'flat' sets; the possibility of nesting sets, offered by the membership relation, allows to build richer domains.

In this paper we wish to put the basis for a study of this kind of nested-set domains for abstract interpretation. In particular, starting from the domain of first order terms, we present several abstract set domains allowing one to retain as many structural properties of terms as wanted. A (linear) ordering based on the classical property *'to be more abstract'* is given for the abstraction functions presented.

Keywords: Abstract interpretation, abstract domains, set theory, unification.

1 Introduction

Abstract interpretation (see, e.g., [10, 11, 12]) is a theory of semantic approximation which is used for a variety of applications. Among them, data flow analysis, comparison of formal semantics, and the design of proof methods. Abstract interpretation techniques have been carefully studied by logic programmers since they are useful for compiler optimization and for generating optimized code for parallel implementation of logic programming languages (cf., e.g., [20]).

Given a concrete domain of computation, one can map a problem in an abstract domain in which only few particular features are kept into account. Two functions, the abstraction function and the concretization function are needed to relate the two domains. To be useful for abstract interpretation, they must constitute a so-called Galois insertion. Although, modulo isomorphism, the set of Galois insertions starting from a complete lattice can be perfectly characterized (cf., e.g., [10, 11, 12, 15]), given a specific problem, it is important to study some possible (useful) abstract interpretations for it and to develop abstract algorithms.

*The core of this work has been done when this author was affiliated at the *Dip. di Matematica e Informatica* of the *Univ. di Udine*, supported by C.N.R. grant No. 201.15.08.

Abstract interpretation in logic programming has recently profited from expressivity of set domains. For instance, in [7] the authors show the usefulness for sharing analysis of abstract interpretation of Prolog programs into a universe of ACI terms; in [16] the authors prove the usefulness of a powerset operator to produce a more precise abstract interpretation starting from simpler ones; in [13] a set theoretic point of view of abstract interpretation is illustrated as a starting point for abstract interpretation for functional languages.

Moreover, the domain `Prop` of propositional formulas, whose usefulness for groundness analysis is shown in [9], is isomorphic to the domain of boolean functions based on the same set of variables. The (standard) encoding can be done with the following function I:

$$\begin{array}{rcl@{\qquad\qquad}rcl} I(V) & \mapsto & V & I(\varphi \vee \psi) & \mapsto & I(\varphi) \cup I(\psi) \\ I(\neg\varphi) & \mapsto & \overline{I(\varphi)} & I(\varphi \wedge \psi) & \mapsto & I(\varphi) \cap I(\psi) \end{array}$$

Observe that, as opposed to what happens for ACI unification ([2]) or for the more general set unification ([14, 1]), the unification problem of boolean terms obtained in this way admits a unique most general unifier ([3]).

Similarly, the domain `Pos` of those formulas modeled by a positive assignment to all variables (proved to be optimal for groundness analysis in [9]) can be mapped into the subset of boolean formulas built using variables, $\cup$, and the relator '$=$', abstracting $\leftrightarrow$, viewed as a boolean functional symbol.

ACI and boolean domains ([7, 9]) are based on 'flat' sets, namely sets consisting of elements that are not sets. In this paper we wish to put the basis for a study of usefulness of nested-set domains for abstract interpretation. The key feature of the set domains presented is the nesting capability of their elements, typical of *sets* as classically intended. The expressive power of set theory allows to propose natural encodings of terms into sets. This can be done using the classically interpreted membership relation '$\in$' in a rather simpler way (cf., e.g., [17]). On the other hand, the most used set domains for abstract interpretation, constituted of *flat* sets, do not allow such encodings. In this paper we prove how structural properties of terms can be retained forgetting specific parts of the encoding process. Moreover, we show how to to approximate (standard) unification in the presented domains; for doing that, we develop a concept of abstract unification generalizing those of [6, 5, 4]. A (linear) ordering based on the property *'to be more abstract'* is given for the functions presented.

§ 2 gives the main notation and classical results used in the paper; in § 3.0–3.4 five Galois insertions from the universe of (standard) terms to set domains are presented. In § 3.5 it is proved that the proposed Galois insertions constitute a chain. In § 4 some lines concerning the possible future directions of the work are drawn.

2 Preliminaries

Let Σ be a set of functional symbols, $ar : \Sigma \longrightarrow \mathbb{N}$ the arity function, $\mathcal{V}$ a set of variables, $T(\Sigma, \mathcal{V})$ the set of first order terms built using Σ and $\mathcal{V}$, and $T(\Sigma)$ the set of ground terms. If $t \in T(\Sigma, \mathcal{V})$, then $\mathtt{vars}(t)$ returns the set of its variables. The standard concept of *substitution* is assumed.

Given two partially ordered sets $D^\flat, D^\sharp$ and two functions $\alpha : D^\flat \longrightarrow D^\sharp, \gamma : D^\sharp \longrightarrow D^\flat$, then $\langle D^\flat, \alpha, D^\sharp, \gamma \rangle$ is said to be a *Galois connection* if 1) α and γ are monotonic,

2) $x \leq \gamma(\alpha(x))$ for all $x \in D^\flat$, and 3) $\alpha(\gamma(y)) \leq y$ for all $y \in D^\sharp$. If condition 3 is strengthened by $\alpha(\gamma(y)) = y$, then $\langle D^\flat, \alpha, D^\sharp, \gamma \rangle$ is said to be a *Galois insertion.*

Since, given a set X, $\langle \mathcal{P}(X), \subseteq \rangle$ is a complete lattice, from Propp. 4–8 of [12], it can be immediately derived:

Proposition 2.1 *If $\alpha : \mathcal{P}(\mathcal{A}) \longrightarrow \mathcal{P}(\mathcal{B})$ fulfills $\alpha(\bigcup_{i \in \mathcal{J}} A_i) = \bigcup_{i \in \mathcal{J}} \alpha(A_i)$ for every family $\{A_i \subseteq \mathcal{A} : i \in \mathcal{J}\}$, and $\gamma : \mathcal{P}(\mathcal{B}) \longrightarrow \mathcal{P}(\mathcal{A})$ is defined as $\gamma(B) = \bigcup\{A \subseteq \mathcal{A} : \alpha(A) \subseteq B\}$ for all $B \subseteq \mathcal{B}$, then $\langle \langle \mathcal{P}(\mathcal{A}), \subseteq \rangle, \alpha, \langle \mathcal{P}(\mathcal{B}), \subseteq \rangle, \beta \rangle$ is a Galois connection. Moreover, if α is onto, then it is a Galois insertion.*

This characterization makes trivial the proofs of all Theorems concerning Galois insertions of this paper.

Given a set of terms $\mathcal{A}$, and a first order (possibly equational) theory E, we write $s =_E t$ if $E \vdash \forall\,(s = t)$, for $s, t \in \mathcal{A}$. A substitution θ is a *E-unifier* of a set $S \subseteq \mathcal{A}$ if $s\theta =_E t\theta$ for all $s, t \in S$. A pre-ordering (namely a reflexive and transitive relation) $\leq$ on $\mathcal{A}$ and on the set of substitutions can be defined as follows:

- let $t_1, t_2 \in \mathcal{A}$, then $t_1 \leq t_2$ if and only if there is θ such that $t_1 =_E t_2\theta$ (t_2 is less specific, more general than t_1);
- let θ_1, θ_2 be two substitutions, then $\theta_1 \leq \theta_2$ if and only if there is a substitution γ such that $\theta_1 = \theta_2 \circ \gamma$, namely $t\theta_1 =_E (t\theta_2)\gamma$ for any $t \in \mathcal{A}$.

For instance, if $\mathcal{A} = T(\Sigma, \mathcal{V})$, we have $f(a, a) \leq f(A, a) \leq f(A, A) \leq f(A, B) \leq A$. Moreover, $[X/a, Y/b] \leq [X/A, Y/B]$, $[X/N, Y/N] \leq [X/Y]$, while $[X/a, Y/B]$ and $[X/b, Y/a]$ cannot be compared.

The ordering relation $\leq$ between substitutions induces an equivalence relation $\sim$ on $\mathcal{A}$ as follows: $a \sim b$ if and only if $a \leq b$ and $b \leq a$. In $T(\Sigma, \mathcal{V})$ this means, for instance, that $f(A, B) \sim f(B, A) \sim f(X, Y)$ and $[X/A, Y/B] \sim [X/C, Y/D] \sim [X/Y, Y/X]$.

A E-unifier θ of S is said to be a *more general E-unifier (m.g.u)* if for any E-unifier μ of S, if $\theta \leq \mu$, then $\theta = \mu$. A set Φ is a complete set of E-unifiers of S if for any E-unifier μ of S, there is $\theta \in \Phi$ such that $\mu \leq \theta$. Φ is said *minimal* if for all θ_1, θ_2 in Φ neither $\theta_1 \leq \theta_2$ nor $\theta_2 \leq \theta_1$. If $S = \emptyset$ or $S = \{s\}$, then the empty function ε is the (unique) m.g.u. of S.

We omit the prefix 'E' when the context is clear. We are interested in theories E in which there is a *finite* complete set of m.g.u.'s for any *finite* $S \subseteq \mathcal{A}$. Given a set $S \subseteq \mathcal{A}$, $\mathtt{unify}_{\mathcal{A}}(S)$ returns a (minimal) complete set of m.g.u.'s when it exists, fail elsewhere.

Remark 2.2 *Notice that our definition of E-unifier is not restrictive, provided that for every $n \in \mathbb{N}$, there is an uninterpreted symbol $f \in \Sigma$, $ar(f) = n$. Unifiers (in the usual sense) of a system of the form $\{\ell_1 = r_1, \ldots, \ell_n = r_n\}$ are exactly unifiers of $\{f(\ell_1, \ldots, \ell_n), f(r_1, \ldots, r_n)\}$. Using this definition, unification becomes a unary operation on a domain; thus, in this paper, we assume that for all $n \in \mathbb{N}$, there is $f \in \Sigma$, $ar(f) = n$.*

Given a Galois connection $\langle \mathcal{A}, \alpha, \mathcal{B}, \gamma \rangle$, using the notation of this paper, standard definition of correct (safe) abstract unification ([6, 5, 4]) can be given as follows. For all $B \subseteq \mathcal{B}$, for all $A \subseteq \gamma(\mathcal{B})$,

$$\alpha^*(\mathtt{unify}_{\mathcal{A}}(A)) \leq \mathtt{unify}_{\mathcal{B}}(B)$$

where α^* is the extension of the abstraction function to substitutions. It can be defined as follows:

- let $\theta = [s_1/t_1, \ldots, s_n/t_n]$ be a $E_{\mathcal{A}}$-unifier of $A \subseteq \mathcal{A}$,
- let μ be obtained by removing from $[\alpha(s_1)/\alpha(t_1), \ldots, \alpha(s_n)/\alpha(t_n)]$ all bindings such that $\alpha(s_i) = \alpha(t_i)$;
- if there are two mappings of the form r/s and r/t in μ with $s \neq t$ for some $r, s, t \in \mathcal{B}$, then $\alpha^*(\theta) = \mathsf{fail}$, else $\alpha^*(\theta) = \mu$.

However, such a definition works only if the theories $E_{\mathcal{A}}$ and $E_{\mathcal{B}}$ are *unitary*, namely they ensure the existence of at most one more general unifier for each instance of set unification problem. Since this is not the case for some of the theories analyzed in this paper, we need a new definition, extending the above one.

Intuitively, whenever an abstract set $B \subseteq \mathcal{B}$ (associated to a theory $E_{\mathcal{B}}$) describes a concrete set $A \subseteq \mathcal{A}$ (associated to a theory $E_{\mathcal{A}}$), then the result of $\mathtt{unify}_{\mathcal{B}}(B)$ must describe the result of $\mathtt{unify}_{\mathcal{A}}(A)$. Given a Galois connection $\langle\langle \mathcal{P}(\mathcal{A}), \subseteq\rangle, \alpha, \langle \mathcal{P}(\mathcal{B}), \subseteq\rangle, \gamma\rangle$ such that $\alpha(a) \in \mathcal{B}$ for all $a \in \mathcal{A}$, $\mathtt{unify}_{\mathcal{B}}$ *correctly abstracts* $\mathtt{unify}_{\mathcal{A}}$ if and only if, for any finite $A \subseteq \mathcal{A}$:

- if $\mathtt{unify}_{\mathcal{B}}(\alpha(A))$ returns fail, then $\mathtt{unify}_{\mathcal{A}}(A)$ returns fail, and
- if $\mathtt{unify}_{\mathcal{B}}(\alpha(A))$ returns $\theta_1, \ldots, \theta_m$ and $\mathtt{unify}_{\mathcal{A}}(A)$ returns $\mu_1, \ldots, \mu_n$, then for all μ_i such that $\alpha^*(\mu_i) \neq \mathsf{fail}$ there exists θ_j such that $\alpha^*(\mu_i) \leq \theta_j$.

In this case we also say that $\theta_j \propto \mu_i$.

Remark 2.3 *If $A \in \gamma(B)$, for some $B \subseteq \mathcal{B}$, and the theories $E_{\mathcal{A}}$ and $E_{\mathcal{B}}$ are both unitary, the equivalence of our definition with the classical one is evident; if $E_{\mathcal{A}}$ is unitary and $E_{\mathcal{B}}$ is not, then it is possible that also 'wrong' unifiers are returned by $\mathtt{unify}_{\mathcal{B}}$. However, at least one of them is required to be correct. Aiming at developing a complete abstract interpreter for a logic programming language, this is a new source of non-determinism.*

Throughout this paper the concrete domain $\mathcal{A}$ will be simply $T(\Sigma, \mathcal{V})$. Thus, $\mathsf{unify}_{T(\Sigma,\mathcal{V})}$ (or simply unify) is the standard unification algorithm (cf., e.g., [19]). In particular, in the above definition, $n = 1$ (uniqueness of the concrete m.g.u.). $\mathcal{B}$ will be a set domain, hence m can be greater than 1. To fix the ideas, let us consider the following example:

Example 2.4 *Let $\{f(X, Y), f(X', g(Y'))\} \subseteq T(\Sigma, \mathcal{V})$. Then, $\mathtt{unify}(\{f(X, Y), f(X', g(Y'))\}) = [X/X', Y/g(Y')]$. Assume that the abstraction function α maps a term into the set of its variables: $\alpha(\{f(X, Y), f(X', Y')\}) = \{\{X, Y\}, \{X', Y'\}\}$. The m.g.u.'s (in a set theory) of $\mathtt{unify}(\{\{X, Y\}, \{X', Y'\}\})$ are: 1) $[X/X', Y/Y']$ and 2) $[X/Y', Y/X']$. Observe that $\alpha^*([X/X', Y/g(Y')]) = [X/X', Y/Y']$, thus $[X/X', Y/Y'] \propto [X/X', Y/g(Y')]$.*

3 Set abstractions

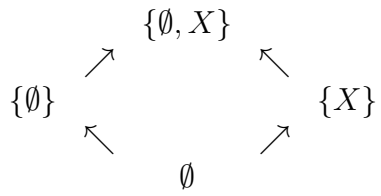
In this section we show five Galois insertions from the domain $\langle \mathcal{P}(T(\Sigma, \mathcal{V})), \subseteq\rangle$, the powerset of first-order terms used in logic programs, and different set domains. We will motivate the rationale behind the choice of such (set) abstract domains.

In particular, the abstraction function α_{i+1} described in § $3.i + 1$ is more precise (in the sense of [10, 8]) than that defined in § $3.i$, for $i = 0, \ldots, 3$. As one can expect by the classical results concerning Galois insertions, they are part of a complete lattice. In particular, in § 3.5 we prove that they constitute a chain of domains.

3.0 A minimal abstraction

In this subsection we identify a Galois connection from the lattice $\langle \mathcal{P}(T(\Sigma, \mathcal{V})), \subseteq \rangle$ to a very simple abstract domain. Intuitively, it can be used to abstract the property *'to be ground'* (see [18]). Later, in Remark 3.6, we show that such abstract domain can be obtained by refining an ACI abstraction of terms.

The abstract domain $D^\sharp$ is the powerset of the set $\mathcal{B}$ consisting of the two elements: $\emptyset$ (abstracting a ground term), and X (abstracting a non ground term). Thus, the abstract domain $\langle D^\sharp, \subseteq \rangle$ is the complete lattice:



The abstraction function α_0 can be defined as follows:

$$\begin{cases} \alpha_0(t) = \emptyset & \text{if } t \text{ is ground} \\ \alpha_0(t) = X & \text{if } t \text{ is not ground} \\ \alpha_0(T) = \bigcup_{t \in T} \{\alpha_0(t)\} & \text{if } T \text{ is a set of terms} \end{cases}$$

For instance, $\alpha_0([\,]) = \emptyset$, and $\alpha_0(f(g(X, g(X, Y)), Z, [\,])) = X$. The concretization function γ_0 can be defined as:

$$\begin{cases} \gamma_0(\emptyset) = \emptyset \\ \gamma_0(\{\emptyset\}) = T(\Sigma) \\ \gamma_0(\{X\}) = \{t \in T(\Sigma, \mathcal{V}) : t \text{ is non-ground}\} \\ \gamma_0(\{\emptyset, X\}) = T(\Sigma, \mathcal{V}) \end{cases}$$

Theorem 3.1 $\langle \langle \mathcal{P}(T(\Sigma, \mathcal{V})), \subseteq \rangle, \alpha_0, \langle \mathcal{P}(\{\emptyset, X\}), \subseteq \rangle, \gamma_0 \rangle$ *is a Galois insertion.*

Proof. Immediate, by definition of α_0, γ_0, and Proposition 2.1. 3.1 ■

It remains to prove the correctness of abstract unification. Consider the various elements of $D^\sharp$. The empty substitution ε is the (unique) m.g.u. of $\emptyset$, $\{\emptyset\}$, and $\{X\}$. $[X/\emptyset]$ is the unique unifier of $\{\emptyset, X\}$. Let $\texttt{unify}_\sharp$ be the function returning such m.g.u.'s. Then,

Theorem 3.2 $\texttt{unify}_\sharp$ *correctly abstracts* $\texttt{unify}$.

Proof. $\texttt{unify}_\sharp$ applied to set of terms returned by α_1 never returns fail, hence the first condition trivially holds. Let T be a unifiable set of terms. If T is empty or T consists of ground terms only, then $\texttt{unify}(T) = \varepsilon = \texttt{unify}_\sharp(\alpha_0(T))$. If T contains a non ground term, then $\texttt{unify}(T)$ is of the form $[X_1/t_1, \ldots, X_n/t_n]$, for some $X_1, \ldots, X_n$ and $t_1, \ldots, t_n$. Thus, $\alpha_0^*([X_1/t_1, \ldots, X_n/t_n])$ can be either $[X/\emptyset]$ or ε. If in T there is also a ground term, then $\alpha_0(T) = \{\emptyset, X\}$ and all $t_1, \ldots, t_n$ are ground. Hence, $\alpha_0^*([X_1/t_1, \ldots, X_n/t_n]) = [X/\emptyset] = \texttt{unify}_\sharp(\{\emptyset, X\})$. If all terms in t are non ground, then $\alpha_0(T) = \{X\}$ and $\texttt{unify}_\sharp(\{X\}) = \varepsilon$ is more general than $\alpha_0^*([X_1/t_1, \ldots, X_n/t_n])$. 3.2 ■

3.1 Collecting variables

The abstract domain proposed in this section is devoted to abstract a term by forgetting all but its set of variables.

Consider the following definition of the abstraction function:

$$\begin{cases} \alpha_1(X) &= X & \text{if } X \text{ is a variable} \\ \alpha_1(c) &= \emptyset & \text{if } c \text{ is a constant} \\ \alpha_1(f(t_0,\ldots,t_n)) &= \alpha_1(t_0) \cup \cdots \cup \alpha_1(t_n) & \\ \alpha_1(T) &= \bigcup_{t\in T}\{\alpha_1(t)\} & \text{if } T \text{ is a set of terms} \end{cases}$$

where $\cup$ is an $ACI1$ functional symbol, namely it fulfills the axioms:

$$\begin{array}{llllll} (A) & (x \cup y) \cup z &=& x \cup (y \cup z) & (I) \quad x \cup x &= x \\ (C) & x \cup y &=& y \cup x & (1) \quad x \cup \emptyset &= x\,. \end{array}$$

while $\emptyset$ (the empty-set) is the identity. For example: $\alpha_1([]) = \emptyset$, $\alpha_1(f(g(X, g(X, Y)), Z, [])) = (X \cup (X \cup Y)) \cup Z \cup \emptyset = X \cup Y \cup Z$.

This abstraction is, essentially, the same used in [7], apart from the part dealing with non-linear terms. In [7] it is shown how it can be used for sharing analysis; moreover, it has also been shown that $ACI1$ must be suitably enhanced to gain an effective usefulness. In [2] a deep and clear analysis of ACI (and $ACI1$) unification is performed.

Example 3.3 *In general, an instance of the $ACI1$ unification problem requires more than one unifier to cover all possible solutions (see [2]). For instance, $\{X_1 \cup X_2, a \cup b\}$ where X_1, X_2 are variables and a, b are constant terms, requires the independent unifiers:*

1. $[X_1/\emptyset, X_2/a \cup b]$,
2. $[X_1/a, X_2/b]$,
3. $[X_1/a, X_2/a \cup b]$,
4. $[X_1/b, X_2/a]$,
5. $[X_1/b, X_2/a \cup b]$,
6. $[X_1/a \cup b, X_2/\emptyset]$,
7. $[X_1/a \cup b, X_2/a]$,
8. $[X_1/a \cup b, X_2/b]$,
9. $[X_1/a \cup b, X_2/a \cup b]$.

Nevertheless, if the problem concerns only variables, the constant $\emptyset$, and the functional symbol $\cup$, and the terms involved are all non-ground, then there is a (unique) most general unifier. For instance, the unique solution for $\{X_1 \cup X_2, Y_1 \cup Y_2\}$ is

$$[X_1/N_1 \cup N_2, X_2/N_3 \cup N_4, Y_1/N_1 \cup N_3, Y_2/N2 \cup N_4]$$

where $N_1, \ldots, N_4$ are new variables. It can be computed building the auxiliary matrix:

	Y_1	Y_2
X_1	N_1	N_2
X_2	N_3	N_4

$ACI1$ axioms guarantee that any term distinct from $\emptyset$ can be equivalently rewritten as a term of the form $X_1 \cup \cdots \cup X_m$ with X_i's pairwise distinct variables (the ordering of the X_i's in the term is immaterial). The abstract domain is therefore the powerset of the set of terms built from $\emptyset$, $\cup$, and $\mathcal{V}$, modulo the congruence relation induced by $ACI1$. We call such set $ACI1$-terms.

We define the concretization function γ_1:

$$\begin{cases} \gamma_1(\emptyset) &= T(\Sigma) \\ \gamma_1(X_1 \cup \cdots \cup X_n) &= \{t \in T(\Sigma,\mathcal{V}) : \mathtt{vars}(t) = \{X_1,\ldots,X_n\}\} \\ \gamma_1(S) &= \bigcup_{s\in S} \gamma_1(s) \qquad \text{if } s \text{ is a set of } ACI1\text{-terms} \end{cases}$$

Theorem 3.4 $\langle\langle\mathcal{P}(T(\Sigma,\mathcal{V})),\subseteq\rangle,\alpha_1,\langle\mathcal{P}(ACI1\text{-terms}),\subseteq\rangle,\gamma_1\rangle$ *is a Galois insertion.*

Proof. Immediate, by definition of α_1,γ_1, and Proposition 2.1. 3.4 ■

It remains to prove the corresponding property for unifiers. Let $\texttt{unify}_{ACI1}$ be a function returning a (minimal) complete set of $ACI1$-m.g.u.'s. Then

Theorem 3.5 $\texttt{unify}_{ACI1}$ *correctly abstracts* `unify`.

Proof. $\texttt{unify}_{ACI1}$ never returns fail, hence the first condition trivially holds. It is sufficient to prove that for any pair of terms t_1,t_2, if θ is a unifier of $\{t_1,t_2\}$, then $\alpha_1^*(\theta)$ is a unifier of $\{\alpha_1(t_1),\alpha_1(t_2)\}$. As a matter of fact, the property for n terms follows from this,[1] since unifiers of $\{t_1,\ldots,t_n\}$ are exactly unifiers of $\{f(t_1,t_2,\ldots,t_{n-1}),f(t_2,t_3,\ldots,t_n)\}$.

We prove this fact by induction on the structure of the terms t_1,t_2.
If $t_1=t_2=c$ for some constant c or if one term from t_1 and t_2 is a variable X and the other is a term not containing X, then the result is trivial.
Assume now that $t_1=f(s_1,\ldots,s_n)$ and $t_2=f(s'_1,\ldots,s'_n)$. θ is a unifier of $\{t_1,t_2\}$ if and only if θ is a unifier of all $\{s_1,s'_1\},\ldots,\{s_n,s'_n\}$. By inductive hypothesis, $\alpha_1^*(\theta)$ is a unifier of all $\{\alpha_1(s_1),\alpha_1(s'_1)\},\ldots,\{\alpha_1(s_n),\alpha_1(s'_n)\}$. Since, by definition, $\alpha_1(t_1)=\alpha_1(s_1)\cup\cdots\cup\alpha_1(s_n)$ and $\alpha_1(t_2)=\alpha_1(s'_1)\cup\cdots\cup\alpha_1(s'_n)$, then $\alpha_1^*(\theta)$ is also a unifier of $\{\alpha_1(t_1),\alpha_1(t_2)\}$ (observe that $ACI1$ axioms are not needed to prove that).

To complete the proof, it only remains to say that, since $\texttt{unify}_{ACI1}$ returns a complete set of $ACI1$ unifiers of $\{\alpha_1(t_1),\alpha_1(t_2)\}$, in particular it returns one more general than $\alpha_1^*(\theta)$. 3.5 ■

Remark 3.6 *The set of $ACI1$-terms can be ordered by the $\leq$ relation (the relation: 'to be an instance of') defined in § 2. It is interesting to point out that all $ACI1$-terms distinct from $\emptyset$ are pairwise equivalent w.r.t. $\leq$. As a matter of fact, let*

$$\begin{array}{rcl} t_1 & = & V_1\cup\cdots\cup V_h\cup X_1\cup\cdots\cup X_m \quad \textit{and}\\ t_2 & = & V_1\cup\cdots\cup V_h\cup Y_1\cup\cdots\cup Y_n \end{array}$$

be two $ACI1$-terms. Then, with

$$\begin{array}{rcl} \theta_1 & = & [X_1/Y_1\cup\cdots\cup Y_n,\ldots,X_m/Y_1\cup\cdots\cup Y_n] \quad \textit{and}\\ \theta_2 & = & [Y_1/X_1\cup\cdots\cup X_m,\ldots,Y_n/X_1\cup\cdots\cup X_m] \end{array}$$

we have $t_1=t_2\theta_2$ and $t_2=t_1\theta_1$, thus $t_1\leq t_2$ and $t_2\leq t_1$.

This means that the abstract domain, modulo the congruence relation induced by $\leq$ and $ACI1$, can be viewed as composed by two classes only: $[\emptyset]$ and $[X]$, with $[\emptyset]\leq[X]$. The abstract domain analyzed in § 3.0 is exactly its powerset.

Remark 3.7 *Notice that an abstraction function reflecting the semantics of the function* `vars` *(that returns the* set *of variables occurring in a term) cannot be used naively to implement abstract unification. For instance, $f(X)$ and $f(a)$ are unifiable with (unique) m.g.u. $[X/a]$, while $\{X\}$ and $\emptyset$ are not unifiable in set theory, since the set $\{X\}$ is not empty.*

[1] We recall that correctness of abstract unification is required for *finite* sets of terms.

3.2 Keeping track of nesting

In this and in the remaining subsection, the expressivity offered by the nesting capability of sets is used to keep track of some properties of terms such as: depth of an occurrence of a subterm, ordering of arguments, and functor names. The abstract domain presented in this section is aimed to abstract from a term any occurrence of a variable occurring in it, together with the nesting of the variable in the term; moreover, duplicates at the same nesting level are removed. The abstraction α_2 'forgets' the names of the function symbols and the ordering of arguments, but not the rank and the overall structure of a term:

$$\begin{cases} \alpha_2(X) &= X & \text{if } X \text{ is a variable} \\ \alpha_2(f(t_1,\ldots,t_n)) &= \{\alpha_2(t_1),\ldots,\alpha_2(t_n)\} & \\ \alpha_2(T) &= \bigcup_{t\in T}\{\alpha_2(t)\} & \text{if } T \text{ is a set of terms} \end{cases}$$

As usual, the empty union (emerging when f is a constant symbol) is $\emptyset$. For example: $\alpha_2([]) = \emptyset$, $\alpha_2(f(g(X, g(X, Y)), Z, [])) = \{\{\{X, \{X, Y\}\}\}, Z, \emptyset\}$.

The abstract domain can be viewed as the powerset of the set closed_set-terms defined as follows. $T(\{\emptyset, \cup, \{\cdot\}\}, \mathcal{V})/ACI1$ is the set of terms built from $\emptyset$, $\cup$, the singleton functional symbol $\{\cdot\}$, and a denumerable set of variables $\mathcal{V}$, modulo the congruence relation induced by $ACI1$ axioms. closed_set-terms is the subset of $T(\{\emptyset, \cup, \{\cdot\}\}, \mathcal{V})/ACI1$ of all the terms without subterms of the form $t \cup X$ or $X \cup t$, X variable, unless t is X or $\emptyset$. 'closed' indicates that all set terms have a bounded cardinality.[2] The set of example above is represented by: $\{\{\{X\} \cup \{\{X\} \cup \{Y\}\}\}\} \cup \{Z\} \cup \{\emptyset\}$.

The concretization function can be recursively defined as follows (w.l.o.g., we can assume that $s_1, \ldots, s_n$ are not equivalent in the theory).

$$\begin{cases} \gamma_2(\emptyset) &= \{c \in \Sigma : ar(c) = 0\} \\ \gamma_2(X) &= \{X\} \\ \gamma_2(\{s_1,\ldots,s_n\}) &= \{f(t_1,\ldots,t_m) : f \in \Sigma, ar(f) = m \geq n, \\ & \quad (\forall i \in \{1,\ldots,n\})\,(\exists j \in \{1,\ldots,m\})\, t_i \in \gamma_2(s_j), \\ & \quad (\forall j \in \{1,\ldots,m\})\,(\exists i \in \{1,\ldots,n\})\, t_i \in \gamma_2(s_j)\,\} \\ \gamma_2(S) &= \bigcup_{s\in S}\{\gamma_2(s)\} \qquad \text{if } S \text{ is a set of closed set terms} \end{cases}$$

Theorem 3.8 *$\langle\langle \mathcal{P}(T(\Sigma,\mathcal{V})), \subseteq\rangle, \alpha_2, \langle \mathcal{P}(\text{closed\_set-terms}), \subseteq\rangle, \gamma_2\rangle$ is a Galois insertion.*

Proof. Immediate, by definition of α_2, γ_2, and Proposition 2.1. 3.8 ■

Unification between objects of closed_set-terms is not $ACI1$ unification, but (nested) set-unification. To see this, consider the following example:

Example 3.9 *One can think that a nested set unification problem as $\{\{X_1, X_2\}, \{a, b\}\}$ can be translated in a $ACI1$ unification problem simply replacing variables (constants) by singletons. In case of the above nested set unification problem we would obtain $\{X_1 \cup X_2, a \cup b\}$, already seen in Example 3.9. The reader can easily check that the nested set unification problem admits only two solutions, while the second, as we already observed, admits 9 independent solutions.*

[2] Alternatively, the abstract domain is the set of 'closed' terms that can be built using the constant symbol $\emptyset$, and the binary set constructor symbol $\{\cdot \mid \cdot\}$, together with the usual denumerable set of variables $\mathcal{V}$. Using the same shorthand as for lists in PROLOG (namely, $\{r, s \mid t\}$ stands for $\{r \mid \{s \mid t\}\}$ and $\{r\}$ stands for $\{r \mid \emptyset\}$). Closed terms are those ended by $\emptyset$. The following are the two axioms regulating the behavior of $\{\cdot \mid \cdot\}$: $\forall xyz\,(\{x, y \mid z\} = \{y, x \mid z\})$, $\forall xy\,(\{x, x \mid y\} = \{x \mid y\})$.

For a nested set unification algorithm, see [14, 1]. The same kind of unification, that we indicate as $\texttt{unify}_{Set}$, will be the abstract unification adopted in the remaining part of this paper. It can be proved that if $[X_1/t_1, \ldots, X_n/t_n]$ is a m.g.u. returned by $\texttt{unify}_{Set}(S)$, for $S \subseteq \mathsf{closed\_set\text{-}terms}$, then $t_1, \ldots, t_n$ are all closed set terms.

Theorem 3.10 $\texttt{unify}_{Set}$ *correctly abstracts* unify.

Proof. As done for Theorem 3.5, we prove that for any pair of terms t_1, t_2, if θ is a unifier of $\{t_1, t_2\}$, then $\alpha_2^*(\theta)$ is a set-unifier of $\{\alpha_2(t_1), \alpha_2(t_2)\}$. We prove this fact by induction on the structure of the terms t_1, t_2.
If $t_1 = t_2 = c$ for some constant c or if one term from t_1 and t_2 is a variable X and the other is a term not containing X, then the result is trivial.
Assume now $t_1 = f(s_1, \ldots, s_n)$ and $t_2 = f(s'_1, \ldots, s'_n)$. θ is a unifier of $\{t_1, t_2\}$ if and only if θ is a unifier of all $\{s_1, s'_1\}, \ldots, \{s_n, s'_n\}$. By inductive hypothesis, $\alpha_2^*(\theta)$ is a unifier of all $\{\alpha_2(s_1), \alpha_2(s'_1)\}, \ldots, \{\alpha_2(s_n), \alpha_2(s'_n)\}$. Since, by definition, $\alpha_2(t_1) = \{\alpha_2(s_1), \ldots, \alpha_2(s_n)\}$ and $\alpha_2(t_2) = \{\alpha_2(s'_1), \ldots, \alpha_2(s'_n)\}$, then $\alpha_2^*(\theta)$ is also a unifier of $\{\alpha_2(t_1), \alpha_2(t_2)\}$.

As a consequence, if $\mathsf{unify}_{Set}(\alpha_2(T))$ returns fail, then also $\mathsf{unify}(T)$ returns fail. The result follows by completeness of unify_{Set}. 3.10 ■

3.3 Abstracting the functor name

The Galois connection presented below keeps track of the structure of a term but abstracts the functor name:

$$\left\{\begin{array}{rcll} \alpha_3(X) & = & X & \text{if } X \text{ is a variable} \\ \alpha_3(f(t_1, \ldots, t_n)) & = & \{\langle \underline{1}, \alpha_3(t_1)\rangle, \ldots, \langle \underline{n}, \alpha_3(t_n)\rangle\} & \\ \alpha_3(T) & = & \bigcup_{t \in T}\{\alpha_3(t)\} & \text{if } T \text{ is a set of terms} \end{array}\right.$$

where $\langle x, y\rangle$ stands for $\{\{x\}, \{x, y\}\}$ (cf., e.g., [17]). Observe how the nesting of sets is used to keep track of the ordering of arguments.

For example: $\alpha_3([]) = \emptyset$; moreover, $\alpha_3(f(g(X, g(X, Y)), Z, [])) = \{\langle \underline{1}, \{\langle \underline{1}, X\rangle, \{\langle \underline{1}, X\rangle, \langle \underline{2}, Y\rangle\}\}\rangle, \langle \underline{2}, Z\rangle, \langle \underline{3}, \emptyset\rangle\}$.

The abstract domain is the powerset of the set, denoted ordered_set-terms, that can be recursively described using BNF:

$$D ::= V \mid \{\langle \underline{1}, D\rangle, \ldots, \langle \underline{n}, D\rangle\} \qquad n \in \mathbb{N}$$

Observe that $\mathsf{ordered\_set\text{-}terms} \subseteq T(\{\emptyset, \cup, \{\cdot\}\}, \mathcal{V})/ACI1$. Thus, abstract unification is again $\texttt{unify}_{Set}$. Let us define the concretization function γ_3:

$$\left\{\begin{array}{rcll} \gamma_3(\emptyset) & = & \{c \in T(\Sigma) : c \text{ is a constant}\} & \\ \gamma_3(X) & = & \{X\} & \\ \gamma_3(\{\langle \underline{1}, t_1\rangle, \ldots, \langle \underline{n}, t_n\rangle\}) & = & \{f(s_1, \ldots, s_n) : f \in \Sigma, ar(f) = n, s_i \in \gamma_3(t_i)\} & \\ \gamma_3(S) & = & \bigcup_{s \in S}\{\gamma_3(s)\} & \text{if } S \text{ is a set of terms} \end{array}\right.$$

Theorem 3.11 $\langle\langle \mathcal{P}(T(\Sigma, \mathcal{V})), \subseteq\rangle, \alpha_3, \langle \mathcal{P}(\mathsf{ordered\_set\text{-}terms}), \subseteq\rangle, \gamma_3\rangle$ *is a Galois insertion.*

Proof. Immediate, by definition of α_3, γ_3, and Proposition 2.1. 3.11 ■

Also for this case it can be proved that, if $S \subseteq \mathsf{ordered\_set\text{-}terms}$, then $\texttt{unify}_{Set}(S)$ returns unifiers dealing with ordered_set-terms only.

Theorem 3.12 $\texttt{unify}_{Set}$ *correctly abstracts* unify.

Proof. The proof is essentially the same as that of Theorem 3.10. 3.12 ■

3.4 A one to one set abstraction

In this subsection we will present a one-to-one set abstract interpretation. A similar rewriting technique (there used to simulate 'hybrid' sets) can be found in [21]. Being isomorphical to $\langle \mathcal{P}(T(\Sigma,\mathcal{V}),\subseteq\rangle$, this abstract interpretation is practically unuseful. We present it for the sake of completeness and of showing how nesting of sets can be used, as needed, to abstract general properties.

$$\left\{\begin{array}{rcll} \alpha_4(X) & = & X & \text{if } X \text{ is a variable} \\ \alpha_4(f(t_1,\ldots,t_n)) & = & \{\underline{c_f},\langle\underline{1},\alpha_4(t_1)\rangle,\ldots,\langle\underline{n},\alpha_4(t_n)\rangle\} & \\ \alpha_4(T) & = & \bigcup_{t\in T}\{\alpha_4(t)\} & \text{if } T \text{ is a set of terms} \end{array}\right.$$

where a numeral *à la* von Neumann $\underline{c_f}$ is associated to each $f \in \Sigma$. For example, assuming $c_{[]} = \underline{0}$, $c_f = \underline{1}$, $c_g = \underline{2}$, then $\alpha_4([]) = \{\underline{0}\}$, and

$$\begin{array}{rcl} \alpha_4(f(g(X,g(X,Y)),Z,[])) & = & \{\underline{1}, \\ & & \langle\underline{1},\{\underline{2},\langle\underline{1},X\rangle,\langle\underline{2},\{\underline{2},\langle\underline{1},X\rangle,\langle\underline{2},Y\rangle\}\rangle\}\rangle, \\ & & \langle\underline{2},Z\rangle,\langle\underline{3},\{\underline{0}\}\rangle\} \end{array}$$

The abstract domain is the powerset of the subset of the set of closed set terms defined in § 3.2 containing the image of any term. We call such a set funct_set-terms. It can be defined by the BNF:

$$D ::= V \mid \{\underline{c_f},\langle\underline{1},D\rangle,\ldots,\langle\underline{n},D\rangle\} \qquad f\in\Sigma, ar(f)=n$$

It is immediate to see that the abstract domain is a isomorphic copy of the concrete domain; hence, for all $T \subseteq T(\Sigma,\mathcal{V})$, $\mathtt{unify}_{Set}$ admits a unique more (the most) general unifier. The definition of γ_4 is trivial (it is sufficient to invert that of α_4). ¿From these observations, usual results follow trivially:

Theorem 3.13 $\langle\langle\mathcal{P}(T(\Sigma,\mathcal{V})),\subseteq\rangle,\alpha_4,\langle\mathcal{P}(\mathsf{funct\_set\text{-}terms}),\subseteq\rangle,\gamma_4\rangle$ *is a Galois insertion.*

Theorem 3.14 $\mathtt{unify}_{Set}$ *correctly abstracts* unify.

Remark 3.15 *Functions similar to α_2,α_3, and α_4 can be used to encode rational terms into the universe of hypersets (i.e., non well founded sets). This fact suggests a way to perform abstract interpretation of the universe of rational terms: a direction for extending the work done in this paper.*

3.5 Ordering of set abstractions

We have shown the feasibility of set domains for abstracting standard terms and unification. We now prove that abstractions α_0–α_4 can be ordered, and that they constitute a chain. In order to do that we make use the characterization due to Cortesi et al. ([8]):

> Let $\langle D^\flat,\alpha_1,D_1^\sharp,\gamma_1\rangle$ and $\langle D^\flat,\alpha_2,D_2^\sharp,\gamma_2\rangle$ be Galois insertions. $D_1^\sharp$ (properly) $D^\flat$-abstracts $D_2^\sharp$ if and only if $\gamma_1(D_1^\sharp)\subseteq\gamma_2(D_2^\sharp)$ ($\gamma_1(D_1^\sharp)\subset\gamma_2(D_2^\sharp)$).

For the sake of simplicity, with a slight abuse of notation we will equivalently say that α_1 (properly) $D^\flat$-abstracts α_2.

Theorem 3.16 *Assume for all $n \in \mathbb{N}$ there is $f \in \Sigma$, $ar(f) = n$ (see Remark 2.2). Then, the abstraction function α_i $\mathcal{P}(T(\Sigma, \mathcal{V}))$-abstracts α_{i+1} for $i = 0, \ldots, 3$. The abstraction is proper for α_0 vs α_1, α_1 vs α_2, and α_2 vs α_3. Moreover, the abstraction α_3 vs α_4 is proper if there are two functional symbols $f, g \in \Sigma$ such that $ar(f) = ar(g) > 0$.*

Proof. Inclusions of counter-images is trivial to see.

Since, given a variable $X \in \mathcal{V}$, the set $\{t \in T(\Sigma, \mathcal{V}) : FV(t) = X\}$ belongs to $\gamma_1(\mathcal{P}(ACI1\text{-terms})) \setminus \gamma_0(\mathcal{P}(\{\emptyset, X\}))$, then α_0 properly abstracts α_1.

Since the set $\{f(X, \ldots, X) : f \in \Sigma, ar(f) > 0\}$ belongs to $\gamma_2(\mathcal{P}(\mathsf{closed\_set\text{-}terms})) \setminus \gamma_1(\mathcal{P}(ACI1\text{-terms}))$, then α_1 properly abstracts α_2.

Since the set $\{f(X_1, \ldots, X_n) : f \in \Sigma, ar(f) = n\}$ belongs to $\gamma_3(\mathcal{P}(\mathsf{ordered\_set\text{-}terms})) \setminus \gamma_2(\mathcal{P}(\mathsf{closed\_set\text{-}terms}))$, then α_2 properly abstracts α_3.

If there are no distinct functional symbols $f, g \in \Sigma$, $ar(f) = ar(g) > 0$, then there is no way to distinguish the two abstractions. Otherwise, for instance, the two sets $\{f(X, \ldots, X)\}, \{g(X, \ldots, X)\} \in \gamma_4(\mathcal{P}(\mathsf{funct\_set\text{-}terms})) \setminus \gamma_3(\mathcal{P}(\mathsf{ordered\_set\text{-}terms}))$. 3.16 ■

4 Further work

Practical applications of the presented domains (or of particular sub-domains) are under analysis. Moreover, we wish to generalize the approach in two directions: the first is to use simpler set domains to abstract more complex set domains; this would be useful to abstract set unification and/or handling of set constraints. The second is to try to abstract E unification for non trivial equational theories E using the power of set domains. Abstractions of infinite (rational) terms is also feasible, as sketched in Remark 3.15.

Acknowledgements

We wish to thank Agostino Cortesi, Moreno Falaschi, Gilberto Filè, Francesco Ranzato, and the two anonymous referees for their precious suggestions.

References

[1] Arenas-Sánchez, P., and Dovier, A. Minimal Set Unification. *Journal of Functional and Logic Programming* (1997). To appear.

[2] Baader, F., and Büttner, W. Unification in commutative and idempotent monoids. *Theoretical Computer Science 56* (1988), 345–352.

[3] Büttner, W., and Simonis, H. Embedding Boolean Expressions into Logic Programming. *Journal of Symbolic Computation 4* (1987), 191–205.

[4] Codish, M., Dams, D., Filè, G., and Bruynooghe, M. On the design of a correct freeness analysis for logic programs. *Journal of Logic Programming 28*, 3 (1996), 181–206.

[5] Codish, M., Dams, D., and Yardeni, E. Derivation and safety of an abstract unification algorithm for groundness analysis. In *Eight International Conference on Logic Programming* (1991), K. Furukawa, Ed., The MIT Press, Cambridge, Mass., pp. 79–96.

[6] Codish, M., Dams, D., and Yardeni, E. Bottom-up abstract interpretation of logic programs. *Theoretical Computer Science 124* (1994), 93–125.

[7] Codish, M., Lagoon, V., and Bueno, F. Sharing Analysis for Logic Programs using Set Logic programs. Technical report, Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, 1996.

[8] Cortesi, A., Filè, G., and Winsborough, W. Comparison of abstract interpretations. In *Automata, Languages and Programming, 19th International Colloquium, ICALP92* (1992), W. Kuich, Ed., vol. 623 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 521–532.

[9] Cortesi, A., Filè, G., and Winsborough, W. Optimal groundness analysis using propositional logic. *Journal of Logic Programming 27*, 2 (1996), 137–167.

[10] Cousot, P., and Cousot, R. Abstract interpretation: a unified framework for static analysis of programs by approximation of fixpoints. In *Proc. of 4th POPL* (1977), ACM, pp. 238–251.

[11] Cousot, P., and Cousot, R. Systematic design of program analysis frameworks. In *Proc. of 6th POPL* (1979), ACM, pp. 269–282.

[12] Cousot, P., and Cousot, R. Abstract Interpretation and Application to Logic Programs. *Journal of Logic Programming 13*, 2 and 3 (1992), 103–179.

[13] Cousot, P., and Cousot, R. Higher-order abstract interpretation. In *International Conference on Computer Languages* (1994), IEEE, pp. 95–112.

[14] Dovier, A., Omodeo, E. G., Pontelli, E., and Rossi, G. {log}: A Language for Programming in Logic with Finite Sets. *Journal of Logic Programming 28*, 1 (1996), 1–44.

[15] Filè, G., Giacobazzi, R., and Ranzato, F. A unifying view of abstract domain design. *Computing Surveys 28*, 2 (1996), 333–336.

[16] Filè, G., and Ranzato, F. Improving Abstract Interpretation by Systematic Lifting to the Powerset. In *Logic Programming, Proceedings of the 1994 International Symposium* (1994), M. Bruynooghe, Ed., The MIT Press, Cambridge, Mass., pp. 655–669.

[17] Levy, A. *Basic Set Theory*. Perspectives in Mathematical Logic. Springer Verlag, 1979.

[18] Marriott, K., and Søndergaard, H. Precise and efficient groundness analysis for logic programs. *LOPLAS 2*, 1–4 (1993), 181–196.

[19] Martelli, A., and Montanari, U. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems 4* (1982), 258–282.

[20] Muthukumar, K., and Hermenegildo, M. V. Compile-time derivation of variable dependency. *Journal of Logic Programming 13*, 2 and 3 (1992), 315–347.

[21] Omodeo, E. G., and Policriti, A. Solvable set/hyperset contexts: I. some decision procedures for the pure, finite case. *Communication on Pure and Applied Mathematics 9–10* (1995), 1123–1155. A special double issue dedicated to Jack Schwartz.