A Relational Abstraction for Functions

B. Jeannet[1], D. Gopan[2], and T. Reps[2]

[1] IRISA; Bertrand.Jeannet@irisa.fr
[2] Comp. Sci. Dept., Univ. of Wisconsin; {gopan,reps}@cs.wisc.edu

Abstract. This paper concerns the abstraction of sets of functions for use in abstract interpretation. The paper gives an overview of existing methods, which are illustrated with applications to shape analysis, and formalizes a new family of relational abstract domains that allows sets of functions to be abstracted more precisely than with known approaches, while being still machine-representable.

1 Introduction

A major strength of abstract interpretation is the ability create complex abstract domains from simpler ones [2]. In particular, (i) Galois connections can be composed, which allows a complex abstraction to be described as the composition of simpler ones, offering the ability to identify clearly the different kind of approximations that take place; (ii) given two abstractions for sets of elements, $\wp(D_i)$, $i = 1, 2$, there exist techniques for abstracting functions of signature $D_1 \to D_2$ [4].

The starting point for the paper is the abstraction method defined in [8], which presents a family of abstract domains that are useful when it is desired to connect storage elements (e.g., elements of arrays and lists) with numeric quantities. This paper reformulates that abstraction in a more general way—as a general method of abstracting a set of functions—which allows the basic idea from [8] to be applied more widely. Moreover, when the new formulation is compared with previously known ways of abstracting a set of functions, it yields more precise abstractions. We are just beginning to explore instantiations of the method that go beyond the ones used in [8].

We formalize a generic abstract-interpretation combinator, which abstract sets of functions of signature $D_1 \to D_2$ in a relational way, assuming the existence of abstractions A_1 and $A_2[n]$ for $\wp(D_1)$ and $\wp((D_2)^n)$, respectively (where A_1 is of finite cardinality n). The obtained abstract domain is precisely $A_2[n]$. In contrast to A_1, $A_2[n]$ may be a complex lattice (relational, infinite, and of infinite height), like the lattice of octagons [11] or convex polyhedra [5]. This relational function-abstraction is more precise than the classical approach described in the literature [4], because of its ability to represent relationships between the images of different elements mapped by a set of functions. For instance, consider a set of functions $F \subseteq U \to \mathbb{R}$ (that may represent a set of possible values for an array of reals). If all $f \in F$ satisfy $f(u_1) = f(u_2)$, our abstraction is able to preserve this information in the abstract domain; in that precise sense it may be qualified as relational (which differs from the usual definition of [10]).

In terms of precision, the relational function-abstraction $A_2[n]$ lies in-between the classical function-abstraction $A_1 \to A_2$ and its disjunctive completion [3]. The important point is that $A_2[n]$ is still finitely representable in the same circumstances as $A_1 \to A_2$ (i.e., when $A_1 \to A_2$ is finitely representable, assuming a tabulated representation).

The contribution of the paper are as follows:

- we give an overview of the existing approaches to abstracting functions and relations and analyze the loss of information induced by them;
- we state our new approach to abstracting functions and compare it to existing ones in terms of expressiveness and implementability;
- we illustrate these different abstractions by considering their use in shape analysis; as a side-effect, we show how canonical abstraction can be partially recast in terms of a powerful combination of elementary abstractions, without resorting to the logical framework of [14].

In contrast to the domain construction and refinement approach (e.g., [12, 6, 7]), which operates on general lattices, our approach explicitly exploits the functional structure of concrete states.

The remainder of the paper is organized into four sections: Section 2 introduces some terminology and notation. Section 3 reviews the classical abstractions of functions of signature $D_1 \to D_2$ and relations between elements of D_1 and D_2 that were described in [4]. Section 4 describes relational function-abstraction. Section 5 presents related work and draws some conclusions.

2 Preliminaries

2.1 Lattices and Galois connections

We denote by $L(\sqsubseteq, \perp, \top, \sqcup, \sqcap)$ a lattice defined by the set L and the partial order $\sqsubseteq$, where $\perp$, $\top$, $\sqcup$, and $\sqcap$ denote the smallest element, the greatest element, the least upper bound, and the greatest lower bound, respectively. Given any set D, the powerset $\wp(D)$ is a lattice ordered by set inclusion, and the set of functions $D \to L$ is a lattice ordered by the pointwise ordering: $f \sqsubseteq g \Leftrightarrow \forall d \in D : f(d) \sqsubseteq g(d)$. Given two lattices L_1 and L_2, $L_1 \times L_2$ is a lattice ordered componentwise, in which a pair (x_1, x_2) is identified with $\perp$ if either component is $\perp$. A function $f : L_1 \to L_2$ is strict and total if $f(\perp) = \perp \wedge f(x) = \perp \implies x = \perp$, monotonic if $x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$, and additive if $f(x \sqcup y) = f(x) \sqcup f(y)$. We denote these sets of functions by $L_1 \xrightarrow{\perp} L_2$, $L_1 \xrightarrow{\sqsubseteq} L_2$, and $L_1 \xrightarrow{\sqcup} L_2$, respectively. A lattice will be called a flat lattice if it is formed by a set of unordered elements to which a smallest element and a greatest element are added.

A Galois connection $C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ between two lattices C and A is defined by abstraction and concretization functions $\alpha : C \to A$ and $\gamma : A \to C$ that satisfy $\forall x \in C, \forall y \in A : x \sqsubseteq_C \gamma(y) \iff \alpha(x) \sqsubseteq_A y$. In program analysis, C is most often the powerset of states $\wp(S)$. For any Galois connection: (i) $\gamma \circ \alpha$ is extensive (i.e., greater than the identity function) and represents the information lost by the abstraction; (ii) α preserves $\sqcup$, and γ preserves $\sqcap$; (iii) α is one-to-one iff γ is onto iff $\gamma \circ \alpha$ is the identity; in this case, we use the notation $C \underset{\alpha}{\overset{\gamma}{\twoheadleftarrow\!\!\rightarrow}} A$. If $\gamma \circ \alpha$ is the identity, α loses no information and we will consider that C and A are isomorphic from the information standpoint, which is denoted by $C \simeq A$ (although γ may not be one-to-one).

Set-theoretic model			
Set of cells	Pointer variable z	Pointer field n	Real-valued field x
Cell	$z \in \text{Cell} \cup \{\text{nil}\}$	$n : \text{Cell} \to \text{Cell} \cup \{\text{nil}\}$	$x : \text{Cell} \to \mathbb{R}$
U	$z : U \to \mathbb{B}$	$n : U^2 \to \mathbb{B}$	$x : U \to \mathbb{R}$
Universe	Unary predicate	Binary predicate	Real-valued function
Logical model			

Fig. 1. Two models of a program state.

Any Galois connection $C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ can be refined by considering the disjunctive completion of A [2], which corresponds to $\wp(A)$ equipped with an inclusion order that takes into account the order of A. This refinement allows the disjunction of abstract properties in A to be represented exactly, instead of using $\sqcup_A$, which usually loses information (i.e., in general, one has $\gamma(x \sqcup_A y) \sqsupseteq \gamma(x) \sqcup_C \gamma(y)$). We denote by $\wp(A) \underset{\alpha_\vee}{\overset{\gamma_\vee}{\leftrightarrows}} A$ the Galois connection between the disjunctive completion of a lattice and itself. A is said to be disjunctive if it isomorphic to its disjunctive completion.

As mentioned in the introduction, Galois connections can be composed. We use $[\sigma, \varsigma]$ to denote the composition of a connection σ followed by a connection ς (so that we have $\alpha_{[\sigma,\varsigma]} = \alpha_\varsigma \circ \alpha_\sigma$).

The existence of a Galois connection between two lattices L_1 and L_2 defines the following pre-order: $L_1 \succeq L_2 \Leftrightarrow L_1 \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} L_2$. Given a concrete lattice C, the set of all (equivalence class of) lattices that abstract it, ordered by $\succeq$, is itself a lattice with top element C (see for instance Fig. 2).

2.2 Shape Analysis and Modeling Program States

This section provides background on the semantic domains used in shape analysis; this material will be used later in the paper to illustrate several aspects of the different approaches to abstracting sets of functions.

The aim of shape analysis is to analyze the properties of programs that manipulate heap-allocated storage and perform destructive updating of pointer-valued fields [14]. The goal is to recover shape descriptors that provide information about the characteristics of the data structures that a program's pointer variables can point to. Typically, work on shape analysis considers an imperative language that is equipped with an operational semantics defined using a transition relation between program states.

At a given control point, a program state $s \in S$ is defined by the values of the local variables and the heap. At each control point, the set of possible concrete properties on states is thus $\wp(S)$. The collecting semantics of programs is defined as a system of equations on the lattice of concrete properties.

We now describe two ways in which a state s can be modeled (cf. Fig. 1).

- The set-theoretic model is perhaps more intuitive. We consider a fixed set Cell of memory cells. The value of a pointer variable z is modeled by an element $z \in \text{Cell} \cup \{\text{nil}\}$, where nil denotes the null value. If cells have a pointer-valued field n, the values of n-fields are modeled by a function

n : Cell $\rightarrow$ Cell $\cup$ {nil} that associates with each memory cell the value of the corresponding field.
- [14] models a state using the tools of logic: the set of cells is replaced by a universe U of individuals; the value of a program variable z is defined by a unary predicate on U; and the value of a field n is defined by a binary predicate on U^2. Integrity constraints are used to capture the fact that, for instance, a unary predicate z that represents what program variable z points to can have the value "true" for at most one memory cell [14].

  A real-valued field x can be modeled by a real-valued function on U.

We use the term "predicate of arity n" for a Boolean function $U^n \rightarrow \mathbb{B}$. A predicate can also be seen as a relation belonging to $\wp(U^n)$.

We use $\mathcal{P}_n$ to denote the set of predicates symbols of arity n, and $\mathcal{R}$ to denote the set of real-valued function symbols. With such notation, the concrete state-space considered is:[3]

$$S = (U \rightarrow \mathbb{B})^{|\mathcal{P}_1|} \times (U^2 \rightarrow \mathbb{B})^{|\mathcal{P}_2|} \times (U \rightarrow \mathbb{R})^{|\mathcal{R}|} \tag{1}$$

A concrete property in $\wp(S)$ is thus a relation between functions.

Because U is of unbounded size, concrete properties belonging to $C = \wp(S)$ have to be abstracted. The idea behind canonical abstraction [14] is to partition U into a finite set of equivalence classes $U^\sharp$, and to introduce an unknown value $1/2$ (or top element) to the Boolean set, yielding $\mathbb{T} = \{0, 1, 1/2\}$, so that a predicate $p : U \rightarrow \mathbb{B}$ is abstracted by an object $p^\sharp : U^\sharp \rightarrow \mathbb{T}$.

Example 1. In [14] and in Eqn. (1), the basic sets in use are the universe $D_1 = U$, the set of Booleans $D_2 = \mathbb{B}$, and the set of reals $D_3 = \mathbb{R}$. The universe U is partitioned using an equivalence relation $\simeq$, resulting in $U^\sharp = U/\simeq$. We use $\pi : U \rightarrow U^\sharp$ to denote the corresponding projection function. We then obtain a Galois connection

$$\wp(U) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (U^\sharp)_\bot^\top$$

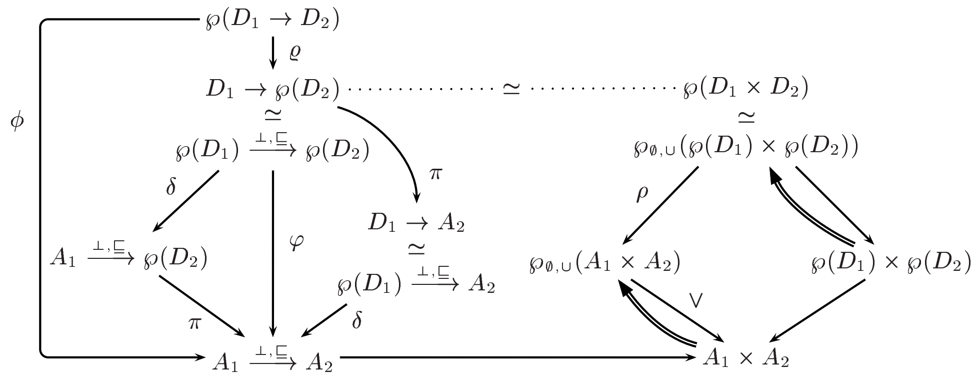
where $(U^\sharp)_\bot^\top$ is the flat domain $U^\sharp$ completed with top and bottom elements, $\gamma(\bot) = \emptyset$, $\gamma(\top) = U$, and $\gamma(u^\sharp) = \pi^{-1}(u^\sharp)$. Booleans are not abstracted, i.e., the domain $\wp(\mathbb{B})$ is abstracted by itself. In most cases, we will not consider the abstractions of reals. In the sequel, $U^\sharp$ will implicitely denote $(U^\sharp)_\bot^\top$. □

3 Classical abstractions of functions and relations

We recall from [4] the classical abstractions of functions of signature $D_1 \rightarrow D_2$ and relations between elements of D_1 and D_2 that can be built from two Galois connections $\wp(D_1) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} A_1$ and $\wp(D_2) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} A_2$. Our notation is mainly taken from [4]. We also make some observations in Sec. 3.3 about exploiting the interplay between functions and relations to obtain suitable abstractions.

We first describe two useful isomorphisms used in the sequel.

[3] Eqn. (1) is really the concrete state-space that one would have if the techniques of [14] were combined with those of [8]. To simplify Eqn. (1), we have omitted nullary predicates, which would be used to model Boolean-valued variables, and nullary functions, which would be used to model real-valued variables.



(a) for sets of functions (b) for relations

$L_1 \longrightarrow L_2$ means that $L_1 \succeq L_2$

$L_1 \Longleftarrow L_2$ means that L_1 is the disjunctive completion of L_2.

Fig. 2. Lattice of abstract domains for functions and relations.

– For any set D and lattice L, we have $(D \to L) \underset{\alpha}{\overset{\gamma}{\leftleftarrows\!\!\rightarrow}} (\wp(D) \xrightarrow{\perp,\sqsubseteq} L)$:

$$\alpha(F)(X) = \bigsqcup_{d \in X} F(d) \quad , \quad \gamma(F^\sharp)(d) = F^\sharp(\{d\})$$

This isomorphism allows to use directly the Galois connection $\wp(D_1) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} A_1$ to abstract the domain of functions $D_1 \to D_2$.

– For any sets D_1 and D_2, we have the isomorphism $\wp(D_1 \times D_2) \underset{\alpha}{\overset{\gamma}{\leftleftarrows\!\!\rightarrow}} \wp_{\emptyset,\cup}(\wp(D_1) \times \wp(D_2))$, which allows to code relations on elements by relations on sets of elements:

$$\alpha(R) = \left\{ (X_1, X_2) \left| \begin{array}{l} \forall d_1 \in X_1, \exists d_2 \in X_2 : (d_1, d_2) \in R \\ \forall d_2 \in X_2, \exists d_1 \in X_1 : (d_1, d_2) \in R \end{array} \right. \right\}$$
$$\gamma(R^\sharp) = \{(x, y) \mid (\{x\}, \{y\}) \in R^\sharp\}$$

$\wp_{\emptyset,\cup}(L_1 \times L_2)$ denotes the set of relations $R \subseteq L_1 \times L_2$ that satisfies
- $(\perp, x_2) \in R \implies x_2 = \perp$, as well as the converse.
- $(x_1, x_2) \in R \wedge (y_1, y_2) \in R \implies (x_1 \sqcup y_1, x_2 \sqcup y_2) \in R$ (this corresponds to a kind of upward-closure).

3.1 Classical abstraction of a functional space

Fig. 2(a) shows classical abstract domains for sets of functions and their relationships. The first abstraction ϱ consists in abstracting a set of functions $F \subseteq D_1 \to D_2$ by a single function $F^\sharp : D_1 \to \wp(D_2)$ (or, equivalently, by a relation between D_1 and D_2):

$$\alpha_\varrho(F)(d_1) = \{f(d_1) \mid f \in F\} \quad , \quad f \in \gamma_\varrho(F^\sharp) \Leftrightarrow \forall d_1 : f(d_1) \in F^\sharp(d_1)$$

In words, $\alpha_\varrho(F)$ collects, for each argument in D_1, the set of its images by the functions in F. Consequently, this abstraction loses the possible relationship between $f(d_1)$ and $f(d_1')$ that may hold for $f \in F$.

Example 2. A set of real-valued functions of signature $U \to \mathbb{R}$ is abstracted with α_ϱ by an element of $U \to \wp(\mathbb{R})$. Fig. 4 of Section 4 depicts concrete value A1, which is abstracted by value D = α_ϱ(A1). The relationship $f(u_2) = f(u_1) + 1$ that holds in A1 is lost in D. $\square$

One can then abstract the equivalent transformer $F : \wp(D_1) \xrightarrow{\perp,\sqsubseteq} \wp(D_2)$ with a function $F^\sharp : A_1 \xrightarrow{\perp,\sqsubseteq} A_2$ by abstracting both the domain and the codomain:

$$\alpha_\varphi(F) = \alpha_2 \circ F \circ \gamma_1 \quad , \quad \gamma_\varphi(F) = \gamma_2 \circ F^\sharp \circ \alpha_1$$

One can also abstract separately the domain (abstraction δ) and the codomain (pointwise abstraction π) to obtain two intermediate abstractions. The full composition of these Galois connections is the Galois connection $\phi = [\varrho, \varphi] = [\varrho, [\delta, \pi]]$ $= [\varrho, [\pi, \delta]]$ between $\wp(D_1 \to D_2)$ and $A_1 \xrightarrow{\perp,\sqsubseteq} A_2$.

Let us take a closer look at the abstractions δ and π. Under δ, a function $F : D_1 \to \wp(D_2)$ is abstracted by a function $A_1 \to \wp(D_2)$ as follows: the image for an element $a \in A_1$ is computed by unioning together the images of the elements of D_1 represented by a:

$$\alpha_\delta(F)(a) = \bigcup_{\alpha_1(d)=a} F(d)$$

An application of this abstraction is illustrated in Fig. 4 ($E = \alpha_\delta(D)$). Under pointwise abstraction π, a function $F : D_1 \to \wp(D_2)$ is abstracted by representing the images of F by their abstraction in A_2: $\alpha_\pi(F) = \alpha_2 \circ F$.

Example 3. If we consider the basic Galois connections described in Example 1, a set of unary predicates of signature $U \to \mathbb{B}$ is abstracted with ϕ by an element of $(U^\sharp)_\perp^\top \xrightarrow{\perp,\sqsubseteq} \wp(\mathbb{B})$. The fact that canonical abstraction [14] can represent exactly both false and true values of predicates (i.e., it is conservative in both values) can be understood if you see unary predicates as ordinary functions abstracted this way. The same holds for binary predicates in $U^2 \to \mathbb{B}$. $\square$

Remark 1. In practice, the abstraction A_1 of the domain D_1 of functions is often a flat lattice induced by a partitioning of D_1.

Remark 2. If $A_1 \xrightarrow{\perp,\sqsubseteq} A_2$ is to be implemented, each of its elements has to be finitely representable. This implies that A_2 should be finitely representable, and A_1 should be finite, as in Example 3; in this case $A_1 \xrightarrow{\perp,\sqsubseteq} A_2 \simeq (A_2)^n$, where n is the size of the partition used to define A_1.

It is not necessary to have an $n+1^{\text{st}}$ dimension to represent the image of $\perp_{A_1}$, which is $\perp_{A_2}$. When A_1 is built as in Example 3, the image of $\top_{A_1}$ does not carry any additional information, because $\gamma_1(\bigsqcup_{a_1 \in A_1 \setminus \{\top_{A_1}\}}) = \bigcup_{a_1 \in A_1 \setminus \{\top_{A_1}\}} \gamma_1(a_1) =$ D_1. However, when the latter property does not hold, one should introduce one additional dimension to the abstract domain for the image of $\top_{A_1}$

3.2 Classical abstraction of a relation

A binary relation between elements belonging to D_1 and D_2, respectively, is an element of $\wp(D_1 \times D_2)$. Fig. 2(b) shows classical abstractions for relations. Roughly speaking, the right-hand side of Fig. 2(b) abstracts a relation between concrete elements as a pair of sets, and then abstracts the pair of sets componentwise.

The left-hand side of Fig. 2(b) abstracts a relation between concrete elements using a relation between abstract elements. The abstraction $\wp_{\emptyset,\cup}(\wp(D_1) \times \wp(D_2)) \underset{\alpha_\rho}{\overset{\gamma_\rho}{\leftrightarrows}} \wp_{\emptyset,\cup}(A_1 \times A_2)$ is defined by:

$$\alpha_\rho(R) = \{(\alpha_1(X_1), \alpha_2(X_2)) \mid (X_1, X_2) \in R\} \tag{2}$$

$$\gamma_\rho(R^\sharp) = \{(X_1, X_2) \mid \exists (a_1, a_2) \in R^\sharp : X_1 \subseteq \gamma_1(a_1) \wedge X_2 \subseteq \gamma_2(a_2)\} \tag{3}$$

$\wp_{\emptyset,\cup}(A_1 \times A_2)$ can also be obtained by disjunctive completion of $A_1 \times A_2$. An observation similar to Remark 1 holds when choosing A_1 and A_2 for building $\wp_{\emptyset,\cup}(A_1 \times A_2)$.

Those principles seem natural when D_1 and D_2 are simple sets without structure, but the notation may seem rather heavy. However, the power of these combinators for relations is that they can be used when, for instance, D_1 and D_2 are sets of functions that are in turn abstracted using the principles described in Section 3.1.

Example 4. Considering the state-space S described in Equation (1), if we abstract functions with ϕ and relations over functions with ρ, we obtain the Galois connection

$$\wp(S) \leftrightarrows \wp_{\emptyset,\sqcup}\left(\left(U^\sharp \to \wp(\mathbb{B})\right)^{|\mathcal{P}_1|} \times \left((U^\sharp)^2 \to \wp(\mathbb{B})\right)^{|\mathcal{P}_2|} \times \left((U^\sharp \to \wp(\mathbb{R})\right)^{|\mathcal{R}|}\right)$$

(Codomains of functions are not abstracted here, and abstract values are not finitely representable.) □

3.3 Exploiting classical abstractions

There is an interplay between the abstraction methods for functions and the abstraction methods for relations, because functions can be coded as relations and conversely. For instance, a function $D_1 \to D_2$ can be coded as a relation in $\wp(D_1 \times D_2)$. A relation in $\wp(D_1 \times D_2)$ can in turn be viewed as a Boolean function $D_1 \times D_2 \to \mathbb{B}$. Each view induces a different abstract domain using the abstractions of the previous sections, as shown in Fig. 3.

Example 5. Coming back to Example 3, if we view a data-structure field as a function $U \to U$ rather than a binary relation $U \times U \to \mathbb{B}$, as in [13, 1], we will abstract $\wp(U \to U)$ by $U^\sharp \to \wp(U^\sharp) \simeq U^\sharp \times U^\sharp \to \mathbb{B}$.[4]

This abstraction is not conservative with respect to the value "true". That is, with this abstraction, "false" means "false", but "true" means "maybe true". An equivalent abstraction can be obtained when starting from $\wp(U^2 \to \mathbb{B})$ by abstracting $\wp(\mathbb{B})$ by conflating the values true and $\top$ [14, Section 8.2]. □

[4] Here the domain U is abstracted by $U^\sharp$, and the codomain U by the more precise disjunctive completion $\wp(U^\sharp)$.

$$
\begin{array}{lccccc}
\text{Viewing a function as:} & D_1 \times D_2 \to \mathbb{B} & & D_1 \to D_2 & & \wp(D_1 \times D_2) \\
\text{Abstracting sets of functions with:} & \Big\downarrow \phi & & \Big\downarrow \phi & & \Big\downarrow [\rho, \vee] \\
\text{Resulting abstract domains:} & A_1 \times A_2 \to \wp(\mathbb{B}) & \succeq & A_1 \to A_2 & \succeq & A_1 \times A_2
\end{array}
$$

Fig. 3. Different ways of coding a set of functions, and the resulting abstractions.

Generally speaking, abstraction methods for functions are more precise than abstraction methods for relations, as illustrated by Fig. 3. If we want to abstract relations in $\wp(D_1 \times D_2)$, viewing them as Boolean functions $D_1 \times D_2 \to \mathbb{B}$ and abstracting them with $A_1 \times A_2 \to \wp(\mathbb{B})$ instead of $\wp(A_1 \times A_2)$ allows to specify that some pairs of elements are definitely related.

Of course, the most suitable coding does not depend only on the induced abstraction, but also on the operations on functions (or relations) that are used for specifying the fixpoint equations to be solved. For instance, both concrete and abstract function application operations are defined in the most straightforward way when viewing a function as an element in $D_1 \to D_2$.

4 Relational Function-Abstraction

If we consider a set of functions of type $f : U \to \mathbb{R}$, and if we abstract it with the technique of Section 3.1 using convex polyhedra [5] for abstracting $\wp(\mathbb{R})$, we would obtain $U^\sharp \to \mathsf{Pol}[1] \simeq U^\sharp \to \mathsf{Interval}$. That is, we would associate to each abstract individual an interval. This is not what is proposed in [8], where an abstract value is a convex polyhedron, where each dimension corresponds to an abstract individual.

In this section, we formalize such an approach in more general terms; we analyze carefully the different kinds of abstractions that are performed; and we compare the approach to the classical approach described in Section 3.1.

4.1 Real-valued functions

To provide some intuition, we first instantiate the abstraction scheme for functions $f : U \to \mathbb{R}$. The universe U is partitioned by a projection function $\pi : U \to U^\sharp$. The cardinality of $U^\sharp$ is denoted by $|U^\sharp| = n$.

Our aim is the following: starting from the lattice $\wp(U \to \mathbb{R})$, we want to abstract it by a lattice of the form $\wp(U^\sharp \to \mathbb{R})$, for which many relational abstractions exist, instead of considering the lattice $U^\sharp \to \wp(\mathbb{R})$ obtained by the classical technique, which is not relational at all.

The abstraction proposed in [8] can be decomposed as follows:

$$
\begin{array}{ccccccc}
\wp(U \to \mathbb{R}) & \xrightarrow{\mu} & \wp(U^\sharp \to \wp(\mathbb{R})) & \xrightarrow{\eta} & \wp(U^\sharp \to \mathbb{R}) & \to & \mathsf{Pol}[n] \\
 & & \simeq & & \simeq & & \\
 & & \wp\big(\wp(\mathbb{R})^n\big) & & \wp(\mathbb{R}^n) & &
\end{array}
$$

The two isomorphisms mentioned in the above equation will be used later to encode functions as vectors: a function $f \in (U^\sharp \to D) \simeq D^n$ can be seen as a vector of elements of D by rewriting it as $\langle f(u_1^\sharp), \ldots, f(u_n^\sharp) \rangle$.

Fig. 4 illustrates the three abstraction steps, which are defined in detail below.

1. $\wp(U \to \mathbb{R}) \xleftarrow[\alpha_\mu]{\gamma_\mu} \wp(U^\sharp \to \wp(\mathbb{R}))$ is defined by

$$\alpha_\mu(\{f\}) = \{\langle X_1, \ldots, X_n\rangle \mid X_i = f \circ \pi^{-1}(u_i^\sharp)\}\ ,\ \alpha_\mu(F) = \bigcup_{f \in F} \alpha_\mu(\{f\})$$
$$\gamma_\mu(\{f^\sharp\}) = \{f \mid \forall u : f(u) \in f^\sharp(\pi(u))\} \qquad ,\ \gamma_\mu(F^\sharp) = \bigcup_{f^\sharp \in F^\sharp} \gamma_\mu(\{f^\sharp\})$$

   where f is lifted to sets in the expression $f \circ \pi^{-1}$. This Galois connection can be seen as the composition $\wp(U \to \mathbb{R}) \xrightarrow{[\varrho,\delta]} U^\sharp \to \wp(\mathbb{R})$ depicted in Fig. 2(a) refined by a disjunctive completion. This explains the fact that γ_μ preserves $\sqcup$.

   Intuitively, this abstraction does not merge functions together; as illustrated by abstract value B1 in Fig. 4, it only merges the values $f(u)$ and $f(u')$ when u and u' are projected to the same abstract individual $u^\sharp$. Because $U^\sharp \to \wp(\mathbb{R})$ is ordered, a value in $\wp(U^\sharp \to \wp(\mathbb{R}))$ is completely characterized by its maximal elements with respect to the $U^\sharp \to \wp(\mathbb{R})$ ordering. (In Fig. 4, we only show maximal elements.)
2. $\underset{\wp(\wp(\mathbb{R})^n)}{\wp(U^\sharp \to \wp(\mathbb{R}))} \xleftarrow[\alpha_\eta]{\gamma_\eta} \underset{\wp(\mathbb{R}^n)}{\wp(U^\sharp \to \mathbb{R})}$ is somewhat subtle:

$$\alpha_\eta(F) = \bigcup_{\langle X_1,\ldots,X_n\rangle \in F} X_1 \times \ldots \times X_n \tag{4}$$
$$\gamma_\eta(F^\sharp) = \{\langle X_1, \ldots, X_n\rangle \mid X_1 \times \ldots \times X_n \subseteq F^\sharp\} \tag{5}$$

   Note that in the right-hand-side lattice, $\wp(U^\sharp \to \mathbb{R})$, each dimension corresponds to a real instead of a set of reals. The subtle point in Eqn. (5) is that the set of vectors $F^\sharp$ is undercovered by a union of Cartesian products.
3. $\wp(\mathbb{R}^n) \overset{\gamma_{\mathsf{Pol}}}{\leftarrow} \mathsf{Pol}[n]$ is the abstraction of sets of vectors by convex polyhedra (in this particular case, because $\mathsf{Pol}[n]$ is not a complete lattice, the abstraction must be formalized using a weaker relationship than Galois connection). Other numerical lattices could be used in place of $\mathsf{Pol}[n]$.

To focus on the abstraction of the domain U, in the following discussion and in Fig. 4 we ignore the abstraction of codomain $\wp(\mathbb{R})$—i.e., we assume that the codomain is not abstracted. (Using the notation of Fig. 2(a), we are redefining ϕ to be the abstraction $[\varrho, \delta]$.)

Intuitively, the composition of abstractions 1. and 2. allows capturing relationships between the images of the different arguments of the functions, when they belong to different equivalence classes. In contrast, with the abstraction ϕ of Fig. 2(a), such relationships are lost (due to the abstraction ϱ).

Example 6. In Fig. 4, the abstraction of concrete value A1 using relational function-abstraction is abstract value C. C concretizes to concrete value A3. Inspection of A3 reveals that abstract value C preserves from A1 the properties

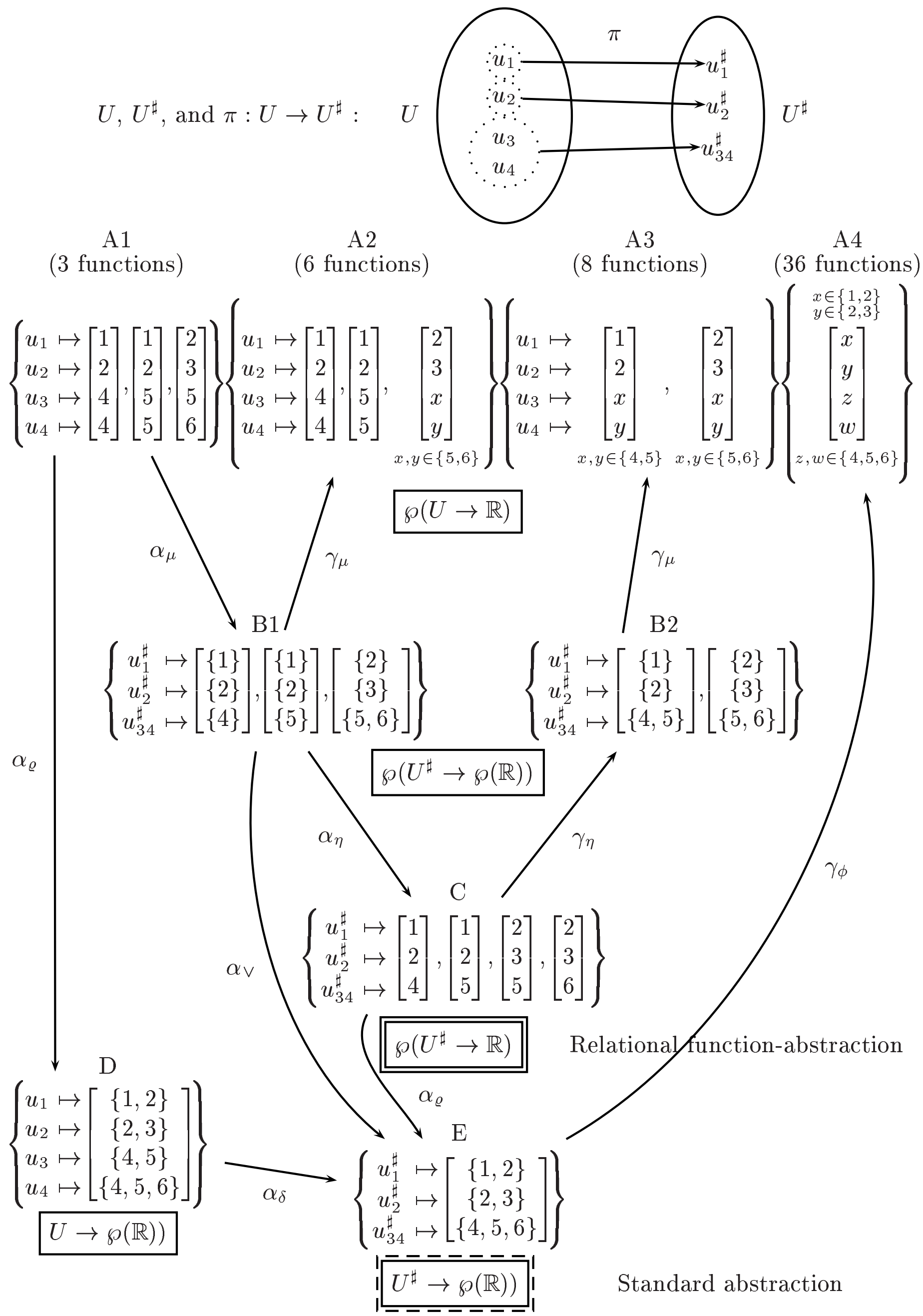


Fig. 4. Different abstractions of the concrete set of functions A1 and the loss of information induced by them (shown by concrete values A2, A3, and A4). Abstract value C (whose concretization is A3) is the abstract value obtained with relational function-abstraction Φ, whereas abstract value E (whose concretization is A4) is the abstract value obtained by the abstraction ϕ of Section 3.1.

$f(u_2) = f(u_1) + 1$ and $\forall u \in \{u_3, u_4\} : f(u_1) + 3 \leq f(u) \leq f(u_1) + 4$. However, C loses the property of A1 that whenever $f(u_1) = 1$, $f(u_3) = f(u_4)$.

In contrast, by using the abstraction ϕ of Section 3.1, we obtain the abstract (functional) value E, from which one can only deduce weaker properties, as shown by its concretization A4. A3 is of cardinality 8, whereas A4 is of cardinality 36. □

To study the loss of information induced by abstraction η (i.e., abstraction-step 2. above), let us consider the expression $\gamma_\eta \circ \alpha_\eta$:

$$\gamma_\eta \circ \alpha_\eta(F) = \left\{ \langle X_1, \ldots, X_n \rangle \;\middle|\; X_1 \times \ldots \times X_n \subseteq \bigcup_{\langle Y_1, \ldots, Y_n \rangle \in F} Y_1 \times \ldots \times Y_n \right\}$$

In words, this means that $(\gamma_\eta \circ \alpha_\eta)(F)$ adds to F Cartesian products that underapproximate unions of Cartesian products.

Example 7. In Fig. 4, we have B2 $= (\gamma_\eta \circ \alpha_\eta)$ (B1). The information that whenever $f(u_1) = 1$, $f(u_3) = f(u_4)$ is lost. More generally, η will lose relational information about $f(u)$ and $f(u')$ when u and u' are merged together. □

4.2 Generalization

In this section, we generalize the results of Section 4.1 by considering the abstraction of functions of signature $D_1 \to D_2$. We also analyze the effect of the abstracting codomain D_2.

Our assumptions are as follows: Suppose that we have a Galois connection $\wp(D_1) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} A_1$, where A_1 is a finite lattice with $|A_1| = n$. Actually, this assumption is only needed for the third abstraction step of Def. 1. In addition, suppose that for any k, we have a Galois connection $\wp((D_2)^k) \leftrightarrows A_2[k]$, where $A_2[k]$ is a k-dimensional abstract domain. We use A_2 to denote $A_2[1]$. We also assume that $\forall k \geq 1 : A_2[k+1] \succeq A_2[k] \times A_2[1]$. If the inequality is actually an equality, $A_2[k] = (A_2)^k$ is said to be not relational [10] (for instance, intervals on reals). If the inequality is strict, $A_2[k]$ is relational (for instance, convex polyhedra)

Definition 1 (Relational function-abstraction Φ). The relational function-abstraction Φ is defined by the composition of the following three abstractions:

$$\wp(D_1 \to D_2) \xrightarrow{\mu} \wp(A_1 \xrightarrow{\perp, \sqsubseteq} \wp(D_2)) \xrightarrow{\eta} \wp(A_1 \backslash \{\perp\} \to D_2) \xrightarrow{\alpha_2[|A_1|]} A_2[|A_1|]$$

where

1. $\wp(D_1 \to D_2) \underset{\alpha_\mu}{\overset{\gamma_\mu}{\leftrightarrows}} \wp(A_1 \xrightarrow{\perp, \sqsubseteq} \wp(D_2))$ is defined by

$$\alpha_\mu(F) = \bigcup_{f \in F} \{f^\sharp \mid \forall a_1 : f^\sharp(a_1) = f(\gamma_1(a_1))\} \tag{6}$$

$$\gamma_\mu(F^\sharp) = \bigcup_{f^\sharp \in F^\sharp} \{f \mid \forall d_1 : f(d_1) \in f^\sharp(\alpha_1(d_1))\} \tag{7}$$

2. $\wp(A_1 \xrightarrow{\perp,\sqsubseteq} \wp(D_2)) \xrightleftharpoons[\alpha_\eta]{\gamma_\eta} \wp(A_1\backslash\{\perp\} \to D_2)$ is defined by[5]

$$\alpha_\eta(F) = \bigcup_{f\in F} \{f^\sharp \mid \forall a_1 : f^\sharp(a_1) \in f(a_1)\} \tag{8}$$

$$\gamma_\eta(F^\sharp) = \{f \mid \forall f^\sharp \in (A_1 \to D_2) : \Big(\forall a_1, f^\sharp(a_1) \in f(a_1)\Big) \Rightarrow f^\sharp \in F^\sharp\} \tag{9}$$

3. The last abstraction is the abstraction of sets of D_2-valued vectors by $A_2[n]$.

Notice that if we drop the assumption that A_2 is finite, we still provide an original method for abstracting $\wp(D_1 \to D_2)$ with $\wp(A_1\backslash\{\perp\} \to D_2)$. We identify below an important class of properties preserved by this abstraction, generalizing the properties mentioned in Example 6.

Theorem 1 (Relational properties preserved by the abstraction $[\mu,\eta]$). Let $P_1 \subseteq (D_1)^2$ and $P_2 \subseteq (D_2)^2$ be binary relations on D_1 and D_2, and let $\Psi_{[P_1,P_2]} \in \wp(D_1 \to D_2)$ be a property on functions defined by

$$\Psi_{[P_1,P_2]} = \{f : D_1 \to D_2 \mid \forall(d_1, d_1') \in P_1, (f(d_1), f(d_1')) \in P_2\}.$$

Assume that P_1 is preserved by the abstraction $\wp(D_1) \xrightleftharpoons[\alpha_1]{\gamma_1} A_1$ as follows:

$$(d_1, d_1') \in P_1 \Rightarrow (\gamma_1 \circ \alpha_1)(d_1) \times (\gamma_1 \circ \alpha_1)(d_1') \subseteq P_1.$$

Let $F \subseteq \Psi_{[P_1,P_2]}$. Then $(\gamma_{[\mu,\eta]} \circ \alpha_{[\mu,\eta]})(F) \subseteq \Psi_{[P_1,P_2]}$; i.e., property $\Psi_{[P_1,P_2]}$ is preserved by the abstraction $[\mu,\eta]$.

Example 8. Coming back to Example 6 and Fig. 4, let $P_1 = \{(u_1,u_3),(u_1,u_4)\}$ and $P_2 = \{(x,y) \mid x+3 \le y \le x+4\}$. Then the property $\Psi_{[P_1,P_2]}$ is $\forall u \in \{u_3,u_4\} : f(u_1)+3 \le f(u) \le f(u_1)+4$, which is satisfied by the set of functions A1. Because P_1 is preserved by the abstraction $\wp(U) \leftrightarrows U^\sharp$, and A3 is equal to $(\gamma_{[\mu,\eta]} \circ \alpha_{[\mu,\eta]})(A1)$, A3 also satisfies $\Psi_{[P_1,P_2]}$.

The above theorem is generalizable to k-ary relations P_1 and P_2 for $k \le |A_1|$. Also, the theorem implies that if, in addition, the relation P_2 is preserved by the abstraction $\alpha_2[|A_1|]$, then the property $\Psi_{[P_1,P_2]}$ is preserved by the full relational function-abstraction Φ. This is the case in the example above if one uses octagons, for instance.

Proof. Let $P_1^\sharp = \{(a_1,a_1') \in (A_1)^2 \mid \exists d_1 \in \gamma_1(a_1), \exists d_1' \in \gamma_1(a_1') : (d_1,d_1') \in P_1\}$. Notice that, due to the assumption on P_1, P_1 is fully defined by $P_1^\sharp$. Then

$$\alpha_\mu(\Psi_{[P_1,P_2]}) = \left\{ f : A_1 \to \wp(D_2) \;\middle|\; \begin{array}{c} \forall(a_1,a_1') \in P_1^\sharp, \forall(d_2,d_2') \in f(a_1) \times f(a_1') : \\ (d_2,d_2') \in P_2 \end{array} \right\}$$

$$\alpha_{[\mu,\eta]}(\Psi_{[P_1,P_2]}) = \{f : A_1 \to D_2 \mid \forall(a_1,a_1') \in P_1^\sharp : (f(a_1), f(a_1')) \in P_2\}$$

$$\gamma_\eta \circ \alpha_{[\mu,\eta]}(\Psi_{[P_1,P_2]}) = \left\{ f : A_1 \to \wp(D_2) \;\middle|\; \begin{array}{c} \forall f^\sharp : A_1 \to D_2 : \\ \Big(\forall a_1, f^\sharp(a_1) \in f(a_1)\Big) \Rightarrow \\ \Big(\forall(a_1,a_1') \in P_1^\sharp : (f^\sharp(a_1), f^\sharp(a_1')) \in P_2\Big) \end{array} \right\}$$

[5] We confess that Eqn. (9), which is derived from the definition of α_η, is difficult to understand. However, this formulation is more general than Eqn. (5); i.e. it can be used when A_1 is not finite.

We now show that $\gamma_\eta \circ \alpha_{[\mu,\eta]}(\Psi_{[P_1,P_2]}) \subseteq \alpha_\mu(\Psi_{[P_1,P_2]})$. Let $f \in \gamma_\eta \circ \alpha_{[\mu,\eta]}(\Psi_{[P_1,P_2]})$, and $(a_1, a_1') \in P_1^\sharp$. We have that for any $f^\sharp : A_1 \to D_2$ such that $\forall a, f^\sharp(a) \in f(a)$, then $(f^\sharp(a_1), f^\sharp(a_1')) \in P_2$. It follows that $\forall (d_2, d_2') \in f(a_1) \times f(a_1')) : (d_2, d_2') \in P_2$, which proves that $f \in \alpha_\mu(\Psi_{[P_1,P_2]})$. It is easy to show that $\gamma_\mu \circ \alpha_\mu(\Psi_{[P_1,P_2]}) \subseteq \Psi_{[P_1,P_2]}$. Now, because $F \subseteq \Psi_{[P_1,P_2]}$ and $(\gamma_{[\mu,\eta]} \circ \alpha_{[\mu,\eta]})$ is monotone, the desired relationship holds: $(\gamma_{[\mu,\eta]} \circ \alpha_{[\mu,\eta]})(F) \subseteq (\gamma_{[\mu,\eta]} \circ \alpha_{[\mu,\eta]})(\Psi_{[P_1,P_2]}) \subseteq \Psi_{[P_1,P_2]}$. □

Example 9. It is instructive to illustrate relational function-abstraction Φ for the case of Boolean functions in $U \to \mathbb{B}$, assuming that the codomain ($\mathbb{B}$) is not abstracted. We obtain $\wp(U^\sharp \to \mathbb{B}) \simeq \wp(\mathbb{B}^n)$; i.e., an abstract value is a set of bit-vectors, or, equivalently, a propositional formula. If we use the abstraction ϕ, we obtain instead $U^\sharp \to \wp(\mathbb{B}) \simeq \wp(\mathbb{B})^n$; i.e., an abstract value is a trivector or a single monomial. This means that in this specific case the abstraction $\Phi = [\mu, \eta]$ reduces to the disjunctive completion μ of the abstraction $[\varrho, \delta] = \phi$. In particular, the abstraction η does not lose any information. This property is not true in general, as shown by Example 7. □

We now compare the relational function-abstraction Φ defined above with ϕ, the traditional approach to abstracting sets of functions, and its disjunctive completion.

Theorem 2. We have the following relationships:

$$\wp(A_1 \xrightarrow{\perp,\sqsubseteq} A_2) \;\succeq\; A_2[|A_1|] \;\succeq\; (A_1 \xrightarrow{\perp,\sqsubseteq} A_2)$$

The first inequality reduces to an equality iff $A_2[|A_1|]$ is disjunctive. The second inequality reduces to an equality iff $A_2[|A_1|]$ is not relational.

Proof. Let $n = |A_1|$ be the cardinality of A_1. We have the isomorphism $(A_1 \xrightarrow{\perp,\sqsubseteq} A_2) \simeq (A_2)^n$. By hypothesis, $\forall k \geq 1 : A_2[k] \succeq (A_2)^k$.

As a consequence, $A_2[n] \succeq (A_2)^n$, which corresponds to the second inequality of the theorem. The equality and strict-inequality cases follow from the definition of a relational lattice.

We denote by $A_2[n] \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A_2)^n$ the Galois connection corresponding to the inequality $A_2[n] \succeq (A_2)^n$. We now define the Galois connection $\wp((A_2)^n) \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} A_2[n]$ with

$$\alpha'(X) = \sqcup\{\gamma(a) \mid a \in X\} \quad \text{and} \quad \gamma'(Y) = \{a \in (A_2)^n \mid \gamma(a) \sqsubseteq Y\}$$

One can easily check that this defines a Galois connection. This proves the first inequality of the theorem. The equality and strict-inequality cases follow from the definition of a disjunctive lattice (cf. Section 2.1) and the fact that $\wp(A_1 \xrightarrow{\perp,\sqsubseteq} A_2)$ is trivially disjunctive. □

Implementability issues. An abstract lattice, even if not implementable, is interesting in so far as it may be used as a semantic domain in an abstraction chain. Here, adopting a pragmatic standpoint, we are interested in knowing when the three abstract domains $\wp(A_1 \xrightarrow{\perp,\sqsubseteq} A_2)$, $A_2[|A_1|]$, and $(A_1 \xrightarrow{\perp,\sqsubseteq} A_2)$ can be used in practice, i.e., when their elements are finitely representable. For the sake of discussion, assume that $A_1 \xrightarrow{\bot,\sqsubseteq} A_2$ is finitely representable by an argument/value table (e.g., A_1 is finite and A_2 is finitely representable). The domain $\wp(A_1 \xrightarrow{\bot,\sqsubseteq} A_2)$ is finitely representable only if A_2 is a finite lattice, or an infinite lattice that does not contain infinite subsets of incomparable elements (or anti-chains). In contrast, the relational function-abstraction $A_2[|A_1|]$ is always finitely representable under our assumptions. In particular,

- $A_2[|A_1|]$ is finitely representable when A_2 is a finite-height lattice. In this case it is also finite-height.
- $A_2[|A_1|]$ is finitely representable even when A_2 is not a finite-height lattice, and can be used in practice, provided that A_2 is equipped with a widening operator.

For instance, if $D_2 = \mathbb{R}$, $A_2[n]$ can be the relational lattice of octagons [11] or convex polyhedra [5]. If $D_2 = \Sigma^*$ is a language over an alphabet Σ, $A_2[n]$ can be the relational lattice $\mathsf{Reg}(\Sigma^n)$ of regular languages over vectors of letters equipped with a suitable widening operator. Neither of these lattices are finite-height.

The disussion above assumed that only argument/value tabular representations are available for functions in $A_1 \to A_2$. Of course, in particular cases, more efficient representations may be available that exploit the underlying structure of the domain and/or codomain. For instance, regular transducers are a very effective representation of (a subset of) functions on $\Sigma^* \to \Sigma^*$, which has both infinite domain and codomain.

4.3 Abstracting relations over functions with relational function-abstraction

The strength of relational lattices is their ability to abstract a powerset of Cartesian products more precisely than the abstractions that were discussed in Section 3.2, as illustrated by the following example:

Example 10. Suppose that we want to abstract relations between vectors (i.e., the concrete domain is $C = \wp(\mathbb{R}^n \times \mathbb{R}^m)$), where sets of vectors $\wp(\mathbb{R}^n)$ and $\wp(\mathbb{R}^m)$ are abstracted by convex polyhedra. The abstraction ρ of Fig. 2(b) results in the abstract domain $\wp(\mathsf{Pol}[n] \times \mathsf{Pol}[m])$, which is not finitely representable. The abstraction $[\rho, \vee]$ results in $\mathsf{Pol}[n] \times \mathsf{Pol}[m]$, which does not capture relationships between pairs very precisely. It is well-known that the most precise way to abstract C is to use the relational lattice $\mathsf{Pol}[n + m]$. □

A similar phenomenon arises when abstracting relations over functions with relational function-abstraction. If we want to abstract a relation in $\wp\big((D_1 \to D_2) \times (D_1 \to D_2)\big)$ (i.e., a relation between functions that share the same domain and codomain), we should use $A_2[2 \cdot |A_1|]$. In other words, we can exploit the set-isomorphism $(D_1 \to D_2)^2 = D_1 \to (D_2)^2$, and then apply relational function-abstraction:

$$\wp\big((D_1 \to D_2) \times (D_1 \to D_2)\big) = \wp\big(D_1 \to (D_2)^2\big) \leftrightarrows A_2[2 \cdot |A_1|]$$

Example 11. Coming back to Eqn. (1), let us illustrate the principles described in this paper to obtain a finitely representable abstract domain for $\wp(S)$:

$$\wp(S) = \wp\left((U \to \mathbb{B})^{|\mathcal{P}_1|} \times (U^2 \to \mathbb{B})^{|\mathcal{P}_2|} \times (U \to \mathbb{R})^{|\mathcal{R}|}\right) \tag{10}$$

$$= \left((U \to \mathbb{B})^{|\mathcal{P}_1|} \times (U^2 \to \mathbb{B})^{|\mathcal{P}_2|}\right) \to \wp\left(U \to \mathbb{R}^{|\mathcal{R}|}\right) \tag{11}$$

$$\leftrightarrows \left(\left(U^\sharp \to \wp(\mathbb{B})\right)^{|\mathcal{P}_1|} \times \left((U^\sharp)^2 \to \wp(\mathbb{B})\right)^{|\mathcal{P}_2|}\right) \xrightarrow{\perp,\sqsubseteq} \mathsf{Pol}[\,|\mathcal{R}| \cdot |U^\sharp|\,] \tag{12}$$

The function in Eqn. (11) is abstracted using the function-abstraction φ of Fig. 2, the abstraction $[\rho, \vee]$ being used for the domain and the relational function-abstraction Φ being used for the codomain. Intuitively, we associate a convex polyhedra to each vector of abstract Boolean functions. □

5 Related Work and Conclusions

We formalized in this paper a generic abstract-interpretation combinator, which abstracts sets of functions $D_1 \to D_2$ in a relational way, assuming the existence of abstractions A_1 and $A_2[n]$ for $\wp(D_1)$ and $\wp((D_2)^n)$, respectively. Viewed from another angle, we have shown how to give a new semantics—in terms of sets of functions—to some previously known abstract lattices, such as octagons and convex polyhedra. This was achieved by developing nonstandard concretization functions for such domains. As an intermediate step, we formalized the abstraction of sets of functions $D_1 \to D_2$ by sets of functions $A_1 \to D_2$ and identified a class of properties preserved by this abstraction.

In terms of precision, the abstract domains that we obtain from relational function-abstraction lie in-between the classical function-abstraction $A_1 \to A_2$ and its disjunctive completion. The important point is that the abstract domains obtained from relational function-abstraction are finitely representable in more general circumstances than the disjunctive completions of classical function-abstractions. In fact, they are finitely representable in the same circumstances as classical function-abstraction (i.e., when $A_1 \to A_2$ is finitely representable with a tabulated representation).

We focused in this paper on the compositional construction of abstract domains and ignored algorithmic issues, as well as the choice of basic abstract domains. The relational function-abstraction described in the paper has actually been implemented in [8] in the shape-analysis framework of [14], for abstracting real-valued fields of dynamically allocated data structures. More recently, [9] has addressed the problem of abstracting arrays of reals, viewed as functions of signature $[0..n] \to \mathbb{R}$. Both [8] and [9] address the difficult problem of abstracting the domain U of a function space in a suitable way. It appears than using a fixed partition of U is useless; instead, [8] and [9] support dynamic partitioning of U.

We compared our solution to the classical solutions described in [4] and their refinement with the disjunctive-completion method. We review here other refinement methods. The tensor product of [12] combines in a relational way two different abstract domains L_1 and L_2 that abstract the same concrete domain C; it is denoted $L_1 \otimes L_2$. The tensor product satifies the equation $\wp(S_1) \otimes \wp(S_2) = \wp(S_1 \times S_2)$ for powersets, and extends such an operation to more general lattices.

The reduced cardinal power [2] and reduced relative power [6] combine L_1 and L_2 in a different way, by considering lattices of functions $L_1 \to L_2$, which captures dependencies, or in a more logical setting, implications [7]. In the particular case where $L_1 = L_2 = L$, this refinement allows to capture autodependencies, which is in general incomparable with the disjunctive completion of L. We did not fully explore whether the above refinements can be used to generate relational function-abstraction Φ from classical abstractions for functions. However, we believe that even if it were possible, the construction would be more complicated than our approach:

- the functional structure of concrete states would not be exploited;
- the "syntactic" structure of the obtained abstract lattice ($L_1 \otimes L_2$ or $L_1 \to L_2$) would be quite different from $A_2[n]$, even if an isomorphism exists.

References

1. D. R. Chase, M. Wegman, and F. K. Zadeck. Analysis of pointers and structures. In Proceedings of the ACM SIGPLAN 1990 conference on Programming language design and implementation. ACM Press, 1990.
2. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In 6th ACM Symposium on Principles of Programming Languages, POPL'79, San Antonio, January 1979.
3. P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. Journal of Logic Programming, 13(2–3):103–179, 1992.
4. P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In Proc. of the 1994 Int. Conf. on Computer Languages, Toulouse, France, May 1994. IEEE Computer Society Press.
5. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In 5th ACM Symposium on Principles of Programming Languages, POPL'78, Tucson (Arizona), January 1978.
6. R. Giacobazzi and F. Ranzato. The reduced relative power operation on abstract domains. Theoretical Computer Science, 216, 1999.
7. R. Giacobazzi and F. Scozzari. A logical model for relational abstract domains. ACM Trans. Program. Lang. Syst., 20(5), 1998.
8. D. Gopan, F. DiMaio, N.Dor, T. Reps, and M. Sagiv. Numeric domains with summarized dimensions. In Int. Conf. on Tools and Algs. for the Construction and Analysis of Systems (TACAS'04), volume 2988 of LNCS, 2004.
9. D. Gopan, T. Reps, and M. Sagiv. A framework for numeric analysis of array operations. In 32th ACM Symposium on Principles of Programming Languages, (POPL'05). ACM press, January 2005.
10. N. Jones and S. Muchnick. Complexity of flow analysis, inducive assertion synthesis, and a language due to Dijkstra. In N. Jones and S. Muchnick, editors, Program Flow Analysis: Theory and Applications. Prentice-Hall, 1981.
11. A. Miné. The octagon abstract domain. In AST'01 in Working Conference on Reverse Engineering 2001. IEEE CS Press, October 2001.
12. F. Nielson. Tensor products generalize the relational data flow analysis method. In Fourth Hungarian Computer Science Conference, 1985.
13. M. Sagiv, T. Reps, and R. Wilhelm. Solving shape-analysis problems in languages with destructive updating. In Symposium on Principles of Programming Languages (POPL'96), pages 16–31, New York, NY, January 1996. ACM Press.
14. M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. ACM Transactions on Programming Languages and Systems, 24(3), 2002.