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

Overview. This paper concerns the abstraction of sets of functions for use in abstract interpretation. The paper gives an overview of abstract domains for functions 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.

A major strength of Abstract Interpretation is the ability create complex abstract domains from simpler ones [1]. 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 $D_1 \rightarrow D_2$ [2].

The starting point for the paper is the abstraction method defined in [5], 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 [5] 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 [5].

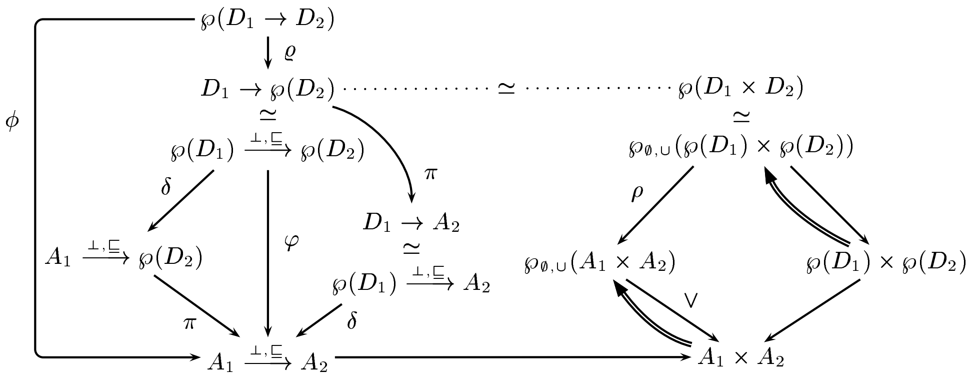
We formalize a generic abstract-interpretation combinator, which abstract sets of functions of signature $D_1 \rightarrow D_2$ in a relational way, assuming the existence of abstractions A_1 and $A_2[n]$ for $\wp(D_1)$ and $\wp((D_2)^n)$, respectively (where A_1 is of finite cardinality n). The obtained 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 [4] or convex polyhedra [3]. This relational function-abstraction is more precise than the classical approach described in the literature [2], because of its ability to represent relationships between the images of different elements mapped by a set of functions.

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

Classical abstractions of functions and relations. Fig. 1 depicts the classical methods for abstracting functions of signature $D_1 \rightarrow D_2$ and relations $R \subseteq D_1 \times D_2$, that can be built from two Galois connections $\wp(D_1) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}}$ and $\wp(D_2) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} A_2$. These abstractions are described in [2] and we use the same notation. Some of the abstractions are illustrated on Fig. 3, where $D_1 = U$,

$A_1 = (U^\sharp)^\top_\bot$ (a flat lattice completed by $\top$ and $\bot$), $D_2 = \mathbb{R}$, and $A_2 = \wp(D_2)$ (no abstraction of the codomain in the example).

In summary, ϱ abstracts a set of functions $F \subseteq D_1 \to D_2$ by a single function $F^\sharp : D_1 \to \wp(D_2)$. One can then abstract the equivalent transformer $F^\sharp : \wp(D_1) \xrightarrow{\bot,\sqsubseteq} \wp(D_2)$ with a function $F^{\sharp\sharp} : A_1 \xrightarrow{\bot,\sqsubseteq} A_2$ by abstracting both the domain and the codomain: $\alpha_\varphi(F^\sharp) = \alpha_2 \circ F^\sharp \circ \gamma_1$, $\gamma_\varphi(F^{\sharp\sharp}) = \gamma_2 \circ F^{\sharp\sharp} \circ \alpha_1$. One can also abstract separately the domain (abstraction δ) and the codomain (pointwise abstraction π) to obtain two intermediate abstractions.



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

$L_1 \simeq L_2$ means lattice isomorphism
$L_1 \longrightarrow L_2$ means that $L_1 \succeq L_2$ (L_2 abstracts L_1)
$L_1 \Longleftarrow L_2$ means that L_1 is the disjunctive completion of L_2.

Fig. 1. Lattice of abstract domains for functions and relations.

Fig. 2 shows also that viewing a function $D_1 \to D_2$ as a Boolean function $D_1 \times D_2 \to \mathbb{B}$, as canonical abstraction [5] does, leads $\simeq$ better precision.

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

Fig. 2. Different ways of coding (sets of) functions, and the resulting abstractions.

Relational function-abstraction. The principle of relational function-abstraction is illustrated in Fig. 3. In this figure, we abstract by different methods the set of functions $A1$, and we observe in $A2$, $A3$ and $A4$, which are images of $A1$ by different $\gamma \circ \alpha$ mappings, the loss of information induced by these methods.

The point of our abstraction is to avoid the abstraction ϱ, which loses (in D) the relation $f(u_1) = f(u_2)$ that holds for any $f \in A1$. Instead, we directly abstract the domain U separately on each element of $A1$, obtaining $B1$. This is equivalent to the disjunctive completion of $\alpha_\delta \circ \alpha_\varrho$.

We then apply the original abstraction η to obtain C. The point is that if $U^\sharp$ is finite, C is a set of $|U^\sharp|$-dimensional vectors and can in turn be abstracted by any numerical lattices (octagons, polyhedra). The definition of η, and particularly

γ_η, is somewhat subtle:

$$\alpha_\eta(F) = \bigcup_{f \in F} \{f^\sharp \mid \forall u^\sharp : f^\sharp(u^\sharp) \in f(u^\sharp)\} \tag{1}$$

$$\gamma_\eta(F^\sharp) = \{f \mid \forall f^\sharp \in (U^\sharp \to \mathbb{R}) : \left(\forall u^\sharp, f^\sharp(u^\sharp) \in f(u^\sharp)\right) \Rightarrow f^\sharp \in F^\sharp\} \tag{2}$$

Another formulation can be given by treating functions from (finite) $U^\sharp$ as vectors:

$$\alpha_\eta(F) = \bigcup_{\langle X_1,\ldots,X_n\rangle \in F} X_1 \times \ldots \times X_n \tag{3}$$

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

A comparison of $B2$ with $B1$ shows what is lost by η. Observe, however, that the relation $f(u_1^\sharp) = f(u_2^\sharp)$ is preserved. A comparison of $A3$ with $A4$ shows what is gained with relational function-abstraction Φ compared to the classical abstraction ϕ. For instance, we preserve in $A3$ the property $f(u_2) + 2 \leq f(u_3) \leq f(u_2) + 3$, unlike in $A4$.

Because γ_η is not trivial, our contribution can also be viewed as a way to give a meaning to a set $|U^\sharp|$-dimensional vectors as a set of functions $U \to \mathbb{R}$, where U may be infinite, using the concretization $\gamma^\vee_{\varrho,\delta} \circ \gamma_\eta$.

In full generality, if the codomain is also abstracted, assuming $\alpha_2[n]$ abstracts $\wp(D_2^n)$ with $A_2[n]$, relational function-abstraction α_Φ is defined as $\alpha_2[n] \circ \alpha_\eta \circ \alpha^\vee_{\varrho,\delta}$.

References

1. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In ACM POPL'79, San Antonio, January 1979.
2. 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.
3. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In ACM POPL'78, Tucson, January 1978.
4. A. Miné. The octagon abstract domain. In AST'01 in Working Conference on Reverse Engineering 2001. IEEE CS Press, October 2001.
5. M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. ACM Transactions on Programming Languages and Systems, 24(3), 2002.

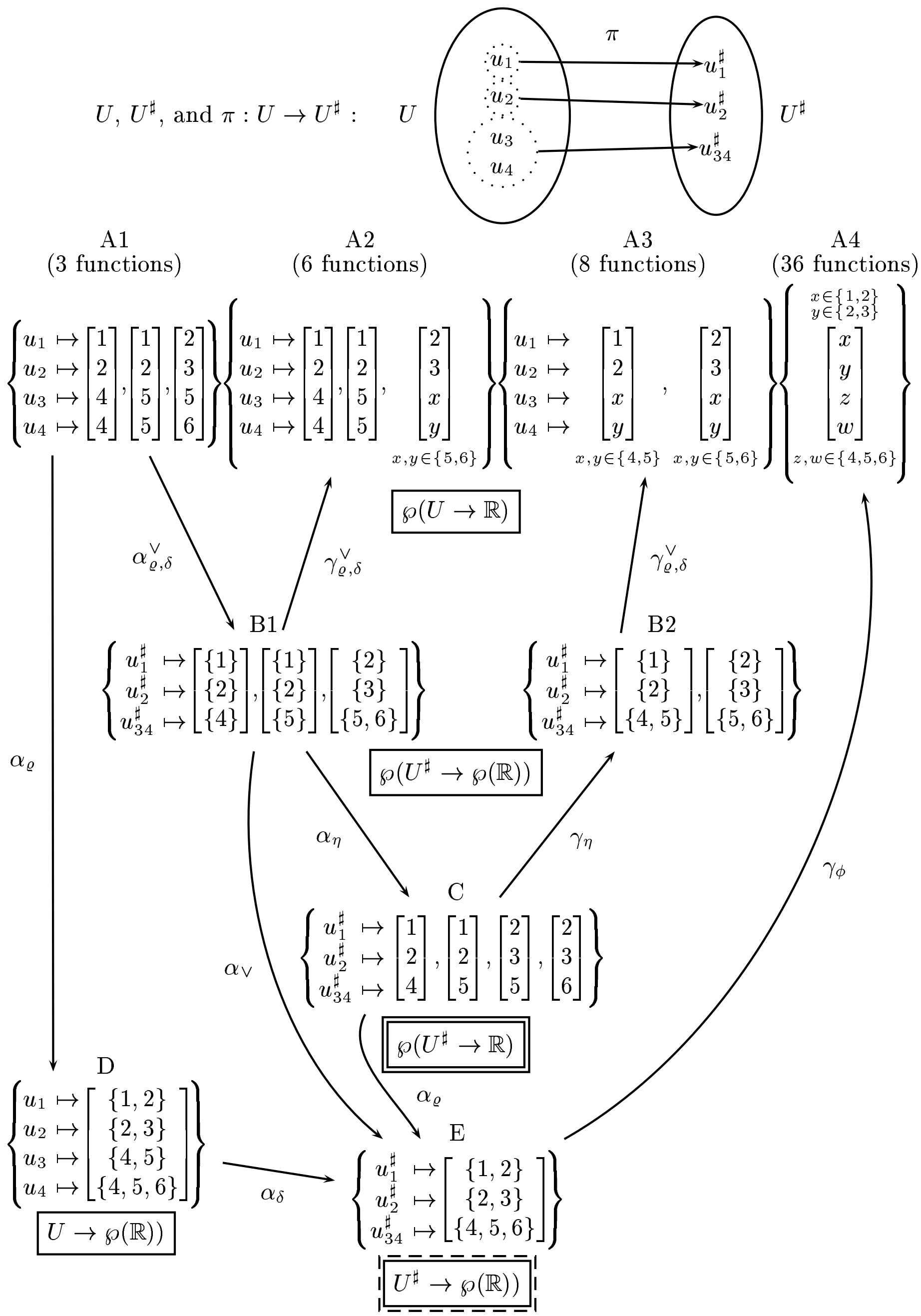


Fig. 3. Different abstractions of the concrete set of functions A1 and the loss of information induced by them (shown by concrete values A2, A3, and A4). Abstract value C (whose concretization is A3) is the abstract value obtained with relational function-abstraction Φ, whereas abstract value E (whose concretization is A4) is the abstract value obtained by the abstraction ϕ.