Logical Relations and Galois Connections

Kevin Backhouse [1] and Roland Backhouse [2]

[1] Computing Laboratory, University of Oxford, Oxford OX1 3QD, England, kevinb@comlab.ox.ac.uk

[2] School of Computer Science and Information Technology, University of Nottingham, Nottingham NG8 1BB, England, rcb@cs.nott.ac.uk

Abstract. Algebraic properties of logical relations on partially ordered sets are studied. It is shown how to construct a logical relation that extends a collection of base Galois connections to a Galois connection of arbitrary higher-order type. "Theorems-for-free" is used to show that the construction ensures safe abstract interpretation of parametrically polymorphic functions.

Logical relations were introduced by Plotkin [Plo80] and Reynolds [Rey83] as a basis for reasoning about possible implementations of the polymorphic lambda calculus and its models. Later, Wadler [Wad89] showed that Reynolds' "abstraction theorem" can be used to derive many useful properties of parametrically polymorphic functions from their types. This paper is about applying the algebraic properties of logical relations to constructing Galois connections of higher-order type.

The paper begins in section 1 with a review of the basic algebraic properties of the arrow operator on relations. Proofs are omitted in this section because most of the results are known. Section 2 contains the main results of the paper. The essential ideas have already been observed by Abramsky [Abr90]; our contribution is to specialise his "uniformization theorem" to Galois-connected functions. This enables us to give a concise calculational formulation of the construction of a Galois connection of any given type, given a collection of Galois connections on the base types.

The primary application of this work is to the construction of abstract interpretations [CC77,CC79]. The first author has successfully applied the theorems presented here to the construction of a definedness test for attribute grammars [Bac02]. In the literature on abstract interpretations, it would appear that there is incomplete understanding of the relevance of logical relations — Cousot and Cousot [CC94] claim that their guidelines for designing abstract interpretations exhibit "a definite advantage of the Galois connection approach to abstract interpretations over its variant formalization using logical relations". We show, however, that the construction of Galois connections of higher-order type is entirely equivalent to the construction of a logical relation.

1 Preliminaries

This section introduces the basic notions for future reference.

Types For the purposes of this paper we assume that types are partially ordered sets. Supposing $(A, \sqsubseteq)$ and $(B, \preceq)$ are posets, $(A, \sqsubseteq) \leftarrow (B, \preceq)$ denotes the set of *monotonic* functions with range A and domain B, ordered pointwise.

Relation Algebra A *binary relation of type* $A \sim B$ is a subset of $A \times B$. Given binary relations $R \in A \sim B$ and $S \in B \sim C$, $R \bullet S$ denotes their composition, the relation of type $A \sim C$ defined by

$$(x, z) \in R \bullet S \quad \equiv \quad \langle \exists y :: (x, y) \in R \wedge (y, z) \in S \rangle \quad . \tag{1}$$

We use $^{\cup}$ as a postfix operator to denote the converse operation on relations. So, if $R \in A \sim B$ then $R^{\cup}$ is the relation of type $B \sim A$ defined by

$$(x, y) \in R^{\cup} \quad \equiv \quad (y, x) \in R \quad . \tag{2}$$

Functions Functions are considered to be special sorts of relations. For function f of type $A \sim B$ we have, for all x in A and all y in B,

$$(x, y) \in f \equiv x = f.y \quad .$$

Galois Connections and Pair Algebras Galois connections are most often defined in terms of a pair of functions. A better starting point for a discussion of Galois connections is, arguably, relations — or so-called "pair algebras", as proposed by Hartmanis and Stearns [HS64,HS66]. A binary relation R on the posets $(A, \sqsubseteq)$ and $(B, \preceq)$ is called a *pair algebra* if there are functions $f \in A \leftarrow B$ and $g \in B \leftarrow A$ such that

$$f^{\cup} \bullet \sqsubseteq \quad = \quad R \quad = \quad \preceq \bullet\, g \quad .$$

If R is a pair algebra, the two functions f and g are said to be the *lower* and *upper adjoints* (respectively) of a *Galois connection* between the posets.

Theorems for Free Based on Reynolds' abstraction theorem [Rey83], Wadler [Wad89] showed how to derive a theorem about a polymorphic function from its type. The key to such "theorems for free" is, in Wadler's words, "that types may be read as relations". Briefly, the type A is read as the identity relation id_A on A and the function space constructor "$\leftarrow$" is read as a mapping from a pair of relations R and S of types $A \sim B$ and $C \sim D$, respectively, to a binary relation $R \leftarrow S$ on functions f and g of type $A \leftarrow C$ and $B \leftarrow D$, respectively. Formally, suppose R and S are binary relations of type $A \sim B$ and $C \sim D$, respectively. Then $R \leftarrow S$ is the binary relation of type $(A \leftarrow C) \sim (B \leftarrow D)$ defined by, for all functions $f \in A \leftarrow C$ and $g \in B \leftarrow D$,

$$(f, g) \in R \leftarrow S \quad \equiv \quad \langle \forall u,v :: (f.u\, ,\, g.v) \in R \Leftarrow (u, v) \in S \rangle \quad . \tag{3}$$

In words, f and g construct R-related values from S-related values.

As an example, suppose $(A, \sqsubseteq)$ and $(B, \preceq)$ are partially ordered sets. Then we can instantiate R to $\sqsubseteq$ and S to $\preceq$ getting a relation $\sqsubseteq \leftarrow \preceq$ between functions f and g of type $A \leftarrow B$. In particular, switching to the usual infix notation for membership of an ordering relation,

$$(f,f) \in \sqsubseteq\leftarrow\preceq \quad\equiv\quad \langle\forall u,v :: f.u \sqsubseteq f.v \Leftarrow u \preceq v\rangle \quad .$$

So $(f,f) \in \sqsubseteq\leftarrow\preceq$ is the statement that f is a monotonic function. In Wadler's words, f maps $\preceq$-related values to $\sqsubseteq$-related values.

Particularly relevant to this paper is that, if f and g are both *monotonic* functions of type $A \leftarrow B$ where $(A,\sqsubseteq)$ and $(B,\preceq)$ are posets,

$$(f,g) \in \sqsubseteq\leftarrow\preceq \quad\equiv\quad \langle\forall u :: f.u \sqsubseteq g.u\rangle \quad .$$

(The easy proof is left to the reader.) We thus recognize $\sqsubseteq\leftarrow\preceq$ as the usual pointwise ordering on monotonic functions of type $(A,\sqsubseteq)\leftarrow(B,\preceq)$. Equivalently, in point-free relation algebra,

$$\sqsubseteq\leftarrow\preceq \;=\; \sqsubseteq\leftarrow\mathsf{id}_{\preceq} \tag{4}$$

where $\mathsf{id}_{\preceq}$ denotes the identity relation on the carrier of the poset ordering $\preceq$. See lemma 11 for the generalisation of this property to higher order types.

"Theorems for free" is the property that, if θ is a parametrically polymorphic function of type t, where t is a type expression parameterised by type variables $a,\ldots,c$, and, for each type variable a, R_a is a relation, then $(\theta,\theta)\in t[a,\ldots,c := R_a,\ldots,R_c]$. For numerous examples of "free" theorems, see Wadler's paper [Wad89].

Note that we should distinguish between the different instances of θ. The free theorem is actually that the pair $(\theta(A),\theta(B))$ is an element of relation $t[a,\ldots,c := R_a,\ldots,R_c]$ where A and B indicate how the particular instances of θ are determined (depending on the types of the relations R). We omit these details for the moment but include them later.

An instance of $R\leftarrow S$ that is used extensively in this paper is when R is a function and S is the converse of a function. For all monotonic functions $f\in(A,\sqsubseteq)\leftarrow(C,\unlhd)$ and $g\in(B,\preceq)\leftarrow(D,\leq)$, $f\leftarrow g^{\cup}$ is the monotonic function of type $(A\leftarrow D)\leftarrow(B\leftarrow C)$ defined by

$$(f\leftarrow g^{\cup}).h \;=\; f{\bullet}h{\bullet}g \quad . \tag{5}$$

(The easy proof is left to the reader.)

Properties The following *distributivity* properties of the $\leftarrow$ operator are easily derived from its definition. For all relations R and S,

$$(R\leftarrow S)^{\cup} \;=\; R^{\cup}\leftarrow S^{\cup} \quad . \tag{6}$$

For all relations R and S, and all functions f and g,

$$(R{\bullet}f)\leftarrow(S\bullet g^{\cup}) \;=\; (R\leftarrow S)\bullet(f\leftarrow g^{\cup}) \quad \text{, and} \tag{7}$$

$$(f^{\cup}\bullet R)\leftarrow(g{\bullet}S) \;=\; (f^{\cup}\leftarrow g)\bullet(R\leftarrow S) \quad . \tag{8}$$

Note carefully that (7) and (8) require f and g to be functions. Using id_X to denote the identity relation on set X, we also have:

$$\mathsf{id}_A\leftarrow\mathsf{id}_B \;=\; \mathsf{id}_{A\leftarrow B} \quad . \tag{9}$$

From now on we assume that composition has precedence over the arrow operator (so that the parentheses can be omitted in the lhs of (7) but not in the rhs).

Relators Another element of "theorems for free" is that type constructors, like disjoint sum $+$, cartesian product $\times$ and $List$ have to be extended to map relations to relations. This led Roland Backhouse to propose the notion of a *relator* as the basis for a relational theory of datatypes [BBH+92,BVW92]. Briefly, a relator is a monotonic functor that commutes with converse. Formally, a relator, F, is a pair of mappings from a source allegory to a target allegory. The first element of the pair is from objects (types) of the source allegory to objects of the target allegory and the second element is from arrows (relations) of the source allegory to arrows of the target allegory. The mappings are required to have the properties that they form a functor from the underlying source category to the underlying target category and, for all arrows R and S,

$$F.R \subseteq F.S \quad \Leftarrow \quad R \subseteq S \quad ,$$

and

$$(F.R)^{\cup} \quad = \quad F.(R^{\cup}) \quad .$$

An example of a unary relator is $List$. Suppose R relates values of type A to values of type B. Then $List.R$ relates values of type $List.A$ to values of type $List.B$. Specifically, the list as is related to the list bs by $List.R$ if as and bs have the same length and corresponding elements are related by R. An example of a binary relator is cartesian product. Suppose R relates values of type A to values of type B and S relates values of type C to values of type D. Then $R{\times}S$ relates values of type $A{\times}C$ to values of type $B{\times}D$. Specifically, the pair (a,c) is related to the pair (b,d) by $R{\times}S$ if a is related by R to b and c is related by S to d. An example of a zero-ary relator is $\mathbb{N}$; this relates two natural numbers exactly when they are equal. (Any type defines a zero-ary relator in this way.)

By design, it is easy to show that relators preserve functions. That is, if f is a function then $F.f$ is also a function.

Note that the formal definition of a relator encompasses the possibility of relators with different source and target allegories. We will be somewhat informal in our treatment of relators. Typically, we assume that the source allegory is the n-fold product of the target allegory with itself, for some unspecified number n ($n \geq 0$) and will write $F.(u \ldots v)$ for the application of the relator to arguments u, $\ldots$, v.

In this paper, we also require that relators preserve partial orderings (so that if R is reflexive, antisymmetric and transitive then so too is $F.R$). For this we need the additional property that relators distribute through binary intersections. That is, we assume that, for all relations R and S, $F.(R{\cap}S) = F.R \cap F.S$. This assumption is satisfied by all "regular" datatypes [Hoo97]. Indeed, it is easily verified for disjoint sum and cartesian product and then a straightforward proof shows that the property is preserved by the usual fixpoint construction of datatypes like $List$. So the assumption is a reasonable one in the context of modern programming languages.

2 Extending Galois Connections

In this section, we give a brief introduction to logical relations (the extension of the arrow operator discussed in section 1 to arbitrary types) and then introduce a *binary* logical-relation operator. (See definitions (14), (15) and (16).) The operator is a key ingredient in the construction of Galois connections of higher-order type. (See lemma 3 and theorem 1.)

2.1 Types and Assignments to Variables

We consider the following (extended BNF) grammar of type expressions:

$$Exp \quad ::= \quad (Relator\ Exp^*) \mid (Exp \leftarrow Exp) \mid Variable$$

where $Variable$ and $Relator$ are (disjoint) finite sets. Variables act as placeholders for types and relations, as explained in detail later. (A common name for them is "polymorphic type variable".) Each element of $Relator$ denotes a relator of a certain arity.

According to the syntax above, the application of a binary relator, say $+$, to types t and u would be denoted by the prefix notation $+tu$. We will, however, deviate from the formal syntax in our examples and write $t{+}u$ instead. Also, the formal syntax stipulates that subtypes are parenthesised (in order to avoid specifying precedence and associativity rules); again we will ignore this requirement when presenting examples. Instead we assume that $\leftarrow$ has the lowest precedence and associates to the left.

An example of a type expression is

$$List.a \leftarrow (Bool \leftarrow a) \times List.a \quad .$$

This uses the the unary relator $List$, binary product relator, $\times$, and the zero-ary relator $Bool$. It describes the type of the (uncurried) $filter$ function on lists, which is parametrically polymorphic in the variable a.

A *variable assignment* is a function with domain $Variable$, the set of type variables. We consider two kinds of variable assignments, one with range posets and the other with range (binary) relations on posets. A variable assignment will be denoted by an assignment statement as in, for example,

$$a{,}b := integer{,}boolean \quad .$$

Given a variable assignment V, its application to type variable a is denoted by V_a.

In the case of variable assignments whose ranges are sets of relations we need to define the converse and composition operators. The definitions are:

$$(V^{\cup})_a \;=\; (V_a)^{\cup} \tag{10}$$

and

$$(V \bullet W)_a \;=\; V_a \bullet W_a \quad . \tag{11}$$

We extend variable assignments to type expressions in two ways. The simpler of the two extends a variable assignment V inductively to all type expressions t: for all type expressions u, v

$$V_{u \leftarrow v} \;=\; V_u \leftarrow V_v \tag{12}$$

and, for all type expressions u, $\ldots$, v,

$$V_{F.(u \ldots v)} \;=\; F.(V_u \ldots V_v) \quad . \tag{13}$$

Note that these definitions make sense whenever the range of V is an assignment of posets or relations to the type variables. In the case that V is a poset assignment we define the $\leftarrow$ operator to map posets A and B into the poset of *monotonic* functions mapping values of poset B into values of poset A, ordered pointwise. Correspondingly, in the case that V assigns relations to the type variables, we restrict the arrow operator to relate *monotonic* functions of the appropriate type. The requirement that the arrow operator relates monotonic functions is essential to the main results in this paper although many of its properties do not depend on this requirement.

In the case that V assigns relations to the type variables, the function mapping type expression t to V_t is called a *logical relation*[1].

The more complicated extension uses two assignments, one for the positive and the other for the negative occurrences of type variables. Specifically, suppose V and W are two variable assignments of the same kind. Then the variable assignment $[V{,}W]$ is defined inductively as follows:

$$[V{,}W]_a \;=\; V_a \quad , \tag{14}$$

$$[V{,}W]_{u \leftarrow v} \;=\; [V{,}W]_u \leftarrow [W{,}V]_v \quad , \tag{15}$$

$$[V{,}W]_{F.(u \ldots v)} \;=\; F.([V{,}W]_u \ldots [V{,}W]_v) \quad . \tag{16}$$

For example,

$$\begin{aligned} & [(a := R),(a := S)]_{a \leftarrow a \leftarrow (a \leftarrow a \times a) \leftarrow List.a} \\ = \;& R \leftarrow S \leftarrow (S \leftarrow R \times R) \leftarrow List.S \quad . \end{aligned}$$

From the above definitions, the following properties are easily proved for all type expressions t by induction on the structure of type expressions.

$$V_t \;=\; [V{,}V]_t \quad , \tag{17}$$

$$(V_t)^{\cup} \;=\; (V^{\cup})_t \quad , \tag{18}$$

$$([V{,}W]_t)^{\cup} \;=\; \left[V^{\cup}{,}W^{\cup}\right]_t \quad . \tag{19}$$

[1] The standard definition of a logical relation is a bit more general: a logical relation is a family of relations indexed by types such that properties (12) and (13) hold for all types u, $\ldots$, v. Clearly, a logical relation in our sense uniquely defines a logical relation according to the standard definition.

Because of (18) we write $V_t^{\cup}$, omitting the parentheses. We also write $[V,W]_t^{\cup}$; here the justification is that we define $[V,W]^{\cup}$ to be $[V^{\cup}, W^{\cup}]$.

Note that it is not the case that $(V \bullet W)_t = V_t \bullet W_t$ for all type expressions t . Property (11) is only valid for type variables a .

2.2 Basic Constructions

The following lemma generalises what is known in the literature as the "independent attribute method" [NNH98, page 247]. Specifically, the independent attribute method is the case that the relator F is the product relator. We state and prove the lemma only for a unary relator. The extension to non-unary relators is easy but requires extra apparatus. (Replace posets, relations and functions everywhere by vectors of posets, relations and functions.)

Lemma 1. Let $(A, \leq)$ be a poset. Suppose F is a relator that preserves binary intersections. (That is, for all relations R and S , $F.(R \cap S) = F.R \cap F.S$.) Then $(F.A\ , F.\leq)$ is a poset.

Proof It is easy to verify that transitivity and reflexivity of $F.\leq$ follows from transitivity and reflexivity of $\leq$. Antisymmetry of $F.\leq$ is the property that

$$F.\leq \cap (F.\leq)^{\cup} \subseteq F.\mathsf{id}_A \quad .$$

This follows from the antisymmetry of $\leq$ ($\leq \cap \leq^{\cup} \subseteq \mathsf{id}_A$) and that F commutes with converse and is monotonic, and the assumption that F preserves binary intersections.

□

We now take the generalisation a step further, to arbitrary higher-order types.

Lemma 2. Let t be a type expression and let $\sqsubseteq$ and $\preceq$ assign partial ordering relations to the type variables in t . Let A and B be the corresponding poset assignments, respectively. (So for each type variable a , $\sqsubseteq_a$ is a partial ordering relation on A_a and $\preceq_a$ is a partial ordering relation on B_a .) Then $[\sqsubseteq, \preceq]_t$ is a partial ordering on $[A, B]_t$. As a corollary, $\sqsubseteq_t$ is a partial ordering relation on A_t .

Proof Straightforward induction on the stucture of type expressions using lemma 1 for the case that t is a relator application. (Reflexivity in the case of the arrow operator is by definition: it is the requirement mentioned above that if A and B are posets then $A \leftarrow B$ is the set of *monotonic* functions with target A and source B .) Property (17) is then used to derive the special case.

□

We remarked earlier that, if f is a total function of type $A \leftarrow B$ and g is a total function of type $C \leftarrow D$ then $f \leftarrow g^{\cup}$ is a total function of type $(A \leftarrow D) \leftarrow (B \leftarrow C)$. The following lemma generalises this property to arbitrary type expressions.

Lemma 3. Suppose f and g are type assignments such that, for all type variables a , f_a and g_a are total functions of type $A_a \leftarrow B_a$ and $C_a \leftarrow D_a$, respectively. Then, for all type expressions t, $[f,g^{\cup}]_t$ is a total function of type $[A,D]_t \leftarrow [B,C]_t$.

Proof Straightforward induction on the structure of type expressions using (19) and the fact that relators preserve total functions.

□

The function that maps type assignments f and g and type expression t to $[f,g^{\cup}]_t$ is parametrically polymorphic. The "theorem for free" that this observation yields is the following [Bru95]. (Compare lemma 4 with lemma 3 noting how types are replaced by relations.)

Lemma 4. Suppose that f, g, h and k assign total functions to type variables, and R and S assign relations to type variables. Suppose further that, for all type variables a, $(f_a,h_a) \in R_a \leftarrow S_a$ and $(g_a,k_a) \in T_a \leftarrow U_a$. Then, for all type expressions t,

$$([f,g^{\cup}]_t \,,\, [h,k^{\cup}]_t) \in [R,U]_t \leftarrow [S,T]_t \quad .$$

□

De Bruin [Bru95] shows how lemma 4 is used to derive a so-called "dinaturality" property from the type of a parametrically polymorphic function. Here, the following corollary is fundamental.

Lemma 5. Suppose that f and g assign total functions to type variables, and R and S assign relations to type variables. Suppose further that, for all type variables a, $(f_a,f_a) \in R_a \leftarrow S_a$ and $(g_a,g_a) \in S_a \leftarrow R_a$. Then, for all type expressions t,

$$([f,g^{\cup}]_t \,,\, [f,g^{\cup}]_t) \;\in\; R_t \leftarrow S_t \quad \text{, and}$$

$$([g,f^{\cup}]_t \,,\, [g,f^{\cup}]_t) \;\in\; S_t \leftarrow R_t \quad .$$

Thus, if, for each a, R_a and S_a are ordering relations, and f_a and g_a are monotonic functions, then $[f,g^{\cup}]_t$ and $[g,f^{\cup}]_t$ are monotonic functions.

Proof Make the instantiation $h,k,T,U \;:=\; f,g,S,R$ in lemma 4 and use (17).

□

Lemma 6. For all assignments f, g, h and k of functions to type variables and for all type expressions t,

$$[f,g^{\cup}]_t \bullet [h,k^{\cup}]_t \;=\; [f\bullet h,(k\bullet g)^{\cup}]_t \quad .$$

(Note the contravariance in the second argument.)

Proof Straightforward induction on the structure of type expressions. The basis follows immediately from (14) and the induction step uses (7) and (16).

□

2.3 Main Theorems

Lemma 7. For all partial orderings $\sqsubseteq$ and $\preceq$ and all functions g,

$$\sqsubseteq \leftarrow \preceq \bullet g \quad = \quad \sqsubseteq \leftarrow g \quad \text{and}$$

$$\sqsubseteq \leftarrow g^{\cup} \bullet \preceq \quad = \quad \sqsubseteq \leftarrow g^{\cup} \quad .$$

Proof We have, for all monotonic functions h and k,

$$
\begin{array}{ll}
 & (h,k) \in \sqsubseteq \leftarrow \preceq \bullet g \\
= & \{ \quad \text{definitions: (3), (1)} \quad \} \\
 & \langle \forall u,v :: h.u \sqsubseteq k.v \;\Leftarrow\; u \preceq g.v \rangle \\
= & \{ \quad (\Rightarrow)\ \text{reflexivity of } \preceq \\
 & \quad\ (\Leftarrow)\ h \text{ is monotonic, transitivity of } \sqsubseteq \quad \} \\
 & \langle \forall v :: h.(g.v) \sqsubseteq k.v \rangle \\
= & \{ \quad \text{definitions: (3), (1)} \quad \} \\
 & (h,k) \in \sqsubseteq \leftarrow g \quad .
\end{array}
$$

The second claim is proved similarly. (This is where monotonicity of k is used.)

□

Lemma 7 is often used in combination with the distributivity properties (7) and (8). Its most immediate application is property (4).

The next lemma can be seen as a special case of Abramsky's "uniformization theorem" [Abr90, proposition 6.4]. (Abramsky does not assume that the function g has an upper adjoint.)

Lemma 8 (Uniformization). If f is a monotonic function with range $(A, \sqsubseteq)$ and the pair $(g, g^{\sharp})$ is a Galois connection between the posets $(B, \preceq)$ and $(C, \leq)$ then

$$f^{\cup} \bullet \sqsubseteq \leftarrow g^{\cup} \bullet \preceq \quad = \quad (f^{\cup} \leftarrow g^{\sharp}) \bullet (\sqsubseteq \leftarrow \preceq) \quad .$$

Also,

$$\sqsubseteq \bullet f \leftarrow \leq \bullet g^{\sharp} \quad = \quad (\sqsubseteq \leftarrow \leq) \bullet (f \leftarrow g^{\cup}) \quad .$$

Proof

$$
\begin{array}{ll}
 & f^{\cup} \bullet \sqsubseteq \leftarrow g^{\cup} \bullet \preceq \\
= & \{ \quad (g, g^{\sharp}) \text{ is a Galois connection.} \\
 & \quad\ \text{So, } g^{\cup} \bullet \preceq \;=\; \leq \bullet g^{\sharp} . \quad \} \\
 & f^{\cup} \bullet \sqsubseteq \leftarrow \leq \bullet g^{\sharp} \\
= & \{ \quad \text{distributivity: (8) and lemma 7} \quad \}
\end{array}
$$

$$(f^{\cup} \leftarrow id_C) \bullet (\sqsubseteq \leftarrow g^{\sharp})$$

$= \quad \{ \quad$ distributivity: (8) (applied twice) $\quad \}$

$$(f^{\cup} \leftarrow g^{\sharp}) \bullet (\sqsubseteq \leftarrow id_B)$$

$= \quad \{ \quad$ lemma 7 $\quad \}$

$$(f^{\cup} \leftarrow g^{\sharp}) \bullet (\sqsubseteq \leftarrow \preceq) \quad .$$

The second property is proved similarly.
□

Lemma 9. Suppose we are given two assignments of Galois connections (f_a, g_a) between the posets $(A_a, \sqsubseteq_a)$ and $(B_a, \preceq_a)$ and (h_a, k_a) between the posets $(C_a, \trianglelefteq_a)$ and $(D_a, \leq_a)$, for each type variable a. Then, for all type expressions t,

$$[f, k^{\cup}]_t^{\cup} \bullet [\sqsubseteq, \trianglelefteq]_t \;=\; [\preceq, \leq]_t \bullet [g, h^{\cup}]_t \quad .$$

In particular, the pair of functions $([f, k^{\cup}]_t \;,\; [g, h^{\cup}]_t)$ forms a Galois connection between the partially ordered sets $([A, C]_t \;,\; [\sqsubseteq, \trianglelefteq]_t)$ and $([B, D]_t \;,\; [\preceq, \leq]_t)$.

Proof Lemma 3 establishes that $[f, k^{\cup}]_t$ and $[g\;, h^{\cup}]_t$ are functions (of the right type) and lemma 2 that $[\sqsubseteq, \trianglelefteq]_t$ and $[\preceq, \leq]_t$ are partial ordering relations. So it suffices only to prove the equality. This we do as follows.

The induction hypothesis is the equality stated above, together with the symmetric equality:

$$[h, g^{\cup}]_t^{\cup} \bullet [\trianglelefteq, \sqsubseteq]_t \;=\; [\leq, \preceq]_t \bullet [k, f^{\cup}]_t \quad .$$

We are given the equalities: for all type variables a,

$$f_a^{\cup} \bullet \sqsubseteq_a \;=\; \preceq_a \bullet g_a \quad \text{, and}$$

$$h_a^{\cup} \bullet \trianglelefteq_a \;=\; \leq_a \bullet k_a \quad .$$

Combined with (14), these establish the basis of the proof.

For the case $t = u \leftarrow v$ we have:

$$[f, k^{\cup}]_t^{\cup} \bullet [\sqsubseteq, \trianglelefteq]_t$$

$= \quad \{ \quad t = u \leftarrow v$, definition (15) $\quad \}$

$$([f, k^{\cup}]_u \leftarrow [k^{\cup}, f]_v)^{\cup} \bullet ([\sqsubseteq, \trianglelefteq]_u \leftarrow [\trianglelefteq, \sqsubseteq]_v)$$

$= \quad \{ \quad$ converse: (6) and (19) $\quad \}$

$$([f, k^{\cup}]_u^{\cup} \leftarrow [k, f^{\cup}]_v) \bullet ([\sqsubseteq, \trianglelefteq]_u \leftarrow [\trianglelefteq, \sqsubseteq]_v)$$

$= \quad \{ \quad$ uniformization (lemma 8) and induction hypothesis

$\qquad ([k, f^{\cup}]_v$ is the upper adjoint of $[h, g^{\cup}]_v$) $\quad \}$

$$[f, k^{\cup}]_u^{\cup} \bullet [\sqsubseteq, \trianglelefteq]_u \;\leftarrow\; [h, g^{\cup}]_v^{\cup} \bullet [\trianglelefteq, \sqsubseteq]_v$$

$$
\begin{array}{ll}
= & \{ \quad \text{induction hypothesis} \quad \} \\
& [\preceq,\leq]_u \bullet [g, h^\cup]_u \;\leftarrow\; [\leq,\preceq]_v \bullet [k, f^\cup]_v \\
= & \{ \quad \text{uniformization (lemma 8) and induction hypothesis} \\
& \quad\; ([k, f^\cup]_v \text{ is the upper adjoint of } [h, g^\cup]_v\,) \quad \} \\
& ([\preceq,\leq]_u \leftarrow [\leq,\preceq]_v) \bullet ([g, h^\cup]_u \leftarrow [h, g^\cup]_v^\cup) \\
= & \{ \quad \text{converse (19)},\ t = u \leftarrow v\,,\ \text{definition (15)} \quad \} \\
& [\preceq,\leq]_t \bullet [g, h^\cup]_t \quad .
\end{array}
$$

The proof of the second equality is completely symmetric.

Finally, the case $t = F.(u \ldots v)$ is a straightforward application of the fact that relators distribute through composition and commute with converse.

□

The pair $(g\,,\,g^\sharp)$ is a *perfect* Galois connection between the posets $(B\,,\preceq)$ and $(C\,,\leq)$ if $g \bullet g^\sharp$ is the identity relation on B.

Lemma 10. If the base Galois connections defined in lemma 9 are all perfect then so are the constructed Galois connections.

Proof That the base connections are perfect means that $f_a \bullet g_a \;=\; A_a$ and $h_a \bullet k_a \;=\; C_a$. (We are taking the liberty of using A_a and C_a to denote identity relations here.) We have to prove that, for all type expressions t,

$$[f, k^\cup]_t \bullet [g, h^\cup]_t \;=\; [A, C]_t \quad .$$

The proof is an application of lemma 6:

$$
\begin{array}{ll}
& [f, k^\cup]_t \bullet [g, h^\cup]_t \;=\; [A, C]_t \\
= & \{ \quad \text{lemma 6} \quad \} \\
& [f \bullet g, (h \bullet k)^\cup]_t \;=\; [A, C]_t \\
\Leftarrow & \{ \quad \text{definition: (14), (15), (16)} \quad \} \\
& \langle \forall a :: (f \bullet g)_a = A_a \;\;\wedge\;\; (h \bullet k)_a^\cup = C_a \rangle \\
= & \{ \quad \text{(11), assumption},\ C_a^\cup = C_a \quad \} \\
& \mathit{true} \quad .
\end{array}
$$

□

In the following theorems we suppose we are given an assignment of Galois connections $(\mathit{abs}_a\,, \mathit{con}_a)$ between the posets $(A_a\,, \sqsubseteq_a)$ and $(B_a\,, \preceq_a)$ for each type variable a. In other words, we are given six functions A, B, $\sqsubseteq$, $\preceq$, abs and con on the type variables. For each variable a, we define P_a to be the pair algebra corresponding to the given Galois connection. That is,

$$\mathit{abs}_a^\cup \bullet \sqsubseteq_a \;=\; P_a \;=\; \preceq_a \bullet \mathit{con}_a \quad . \tag{20}$$

Theorem 1 (Logical Pair Algebras). For all type expressions t,

$$[abs\,,con^{\cup}]_t^{\cup} \bullet \sqsubseteq_t \;=\; P_t \;=\; \preceq_t \bullet [con\,,abs^{\cup}]_t \quad .$$

In words, the logical relation P defines a Galois connection for all type expressions t, namely the pair of functions ($[abs\,,con^{\cup}]_t$, $[con\,,abs^{\cup}]_t$) connecting the partially ordered sets ($A_t\,,\sqsubseteq_t$) and ($B_t\,,\preceq_t$). Moreover, P_t is a pair algebra for all type expressions t.

Proof Lemma 9 establishes the equality between the outer terms. (Take f and h to be equal, and g and k also to be equal.) We, therefore, only have to establish the equality between the first two terms.

The proof is by induction on the structure of type expressions. The basis is the assumption (20) combined with (14). For $t = u \leftarrow v$ we have:

$$
\begin{array}{ll}
 & [abs\,,con^{\cup}]_t^{\cup} \bullet \sqsubseteq_t \\
= & \{ \quad t = u \leftarrow v \text{ , definition and (15)} \quad \} \\
 & ([abs\,,con^{\cup}]_u \leftarrow [con^{\cup}\,,abs]_v)^{\cup} \bullet (\sqsubseteq_u \leftarrow \sqsubseteq_v) \\
= & \{ \quad \text{converse: (6) and (19)} \quad \} \\
 & ([abs\,,con^{\cup}]_u^{\cup} \leftarrow [con\,,abs^{\cup}]_v) \bullet (\sqsubseteq_u \leftarrow \sqsubseteq_v) \\
= & \{ \quad \text{uniformization (lemma 8) and induction hypo} \quad \} \\
 & [abs\,,con^{\cup}]_u^{\cup} \bullet \sqsubseteq_u \;\leftarrow\; [abs\,,con^{\cup}]_v^{\cup} \bullet \sqsubseteq_v \\
= & \{ \quad \text{induction hypothesis, } t = u \leftarrow v \text{ , definition (12)} \quad \} \\
 & P_t \quad .
\end{array}
$$

Finally, the case $t = F.(u \ldots v)$ is a straightforward application of the fact that relators distribute through composition and commute with converse.

□

Theorem 1 expresses formally the fact that the extension of a family of Galois connections from a collection of base types to arbitrary higher-order types is precisely defined by a logical relation (the pair algebra P in the theorem), in contradiction of Cousot and Cousot's [CC94] claim of "a definite advantage of the Galois connection approach to abstract interpretations over its variant formalization using logical relations". An important corollary (already observed by Abramsky [Abr90]) is that the "safety" of this extension is a "free" theorem.

Theorem 2 (Safety for Free). If θ is a parametrically polymorphic function of type t, where t is a type expression parameterised by type variables $a,\ldots,c$, then

$$[abs\,,con^{\cup}]_t\,.\theta(B) \;\sqsubseteq_t\; \theta(A) \quad .$$

($\theta(X)$ denotes the instance of θ of type X_t .)

Proof The assumption that θ has type t and is parametrically polymorphic in all the type variables in t means that

$$(\theta(B)\,,\theta(A)) \in P_t \quad .$$

The theorem follows immediately from theorem 1.

□

3 Pointwise Orderings

In this section we specialise the theorems of the previous section in order to demonstrate more clearly their relevance to practical application. For example, we show that the logical relation $\sqsubseteq_t$ in the statement of theorem 2 is just a pointwise ordering relation. We first introduce a novel way of assigning to type variables.

Suppose V and W are two variable assignments of the same kind. Then the variable assignment $[\![V,W]\!]$ is defined inductively as follows:

$$[\![V,W]\!]_a \;=\; V_a \quad , \tag{21}$$

$$[\![V,W]\!]_{u\leftarrow v} \;=\; [\![V,W]\!]_u \leftarrow W_v \quad , \tag{22}$$

$$[\![V,W]\!]_{F.(u\ldots\, v)} \;=\; F.([\![V,W]\!]_u \ldots [\![V,W]\!]_v) \quad . \tag{23}$$

The assignment $[\![V,W]\!]$ applies the assignment V to the highest positive occurrences of the variables and applies assignment W to all others. For example,

$$\begin{aligned} & [\![(a,b := \sqsubseteq,\trianglelefteq)\;,\;(a,b := I,J)]\!]_{List.((a\leftarrow b)\times b)\;\leftarrow\; a} \\ = \;& List.((\sqsubseteq\leftarrow J)\times\trianglelefteq) \leftarrow I \quad . \end{aligned}$$

If I and J are identity relations on posets A and B, respectively, the relation $List.((\sqsubseteq\leftarrow J)\times\trianglelefteq) \leftarrow I$ is the pointwise ordering relation on functions of type $List.((A\leftarrow B)\times B) \leftarrow B$. As the next lemma states, it is equal to the relation $List.((\sqsubseteq\leftarrow\trianglelefteq)\times\trianglelefteq) \leftarrow \sqsubseteq$.

Lemma 11 (Higher Order Pointwise Orderings). If for each variable a, $\sqsubseteq_a$ is a partial ordering relation on the set A_a and id_a is the identity relation on A_a then, for all type expressions t,

$$[\![\sqsubseteq , id]\!]_t \;=\; \sqsubseteq_t \quad .$$

Proof Straightforward induction on the structure of type expressions with lemma 7 providing the crucial inductive step.
□

In words, lemma 11 states that the pointwise ordering of functions, $[\![\sqsubseteq , id]\!]_t$, is the logical relation $\sqsubseteq_t$ generated by the given base orderings.

Lemma 12.

$$P_t \;=\; [\![abs^{\cup} \bullet \sqsubseteq \;,\; [abs\,,con^{\cup}]^{\cup}]\!]_t \quad .$$

Proof The proof is by induction on the structure of type expressions. The basis is trivial. For $t = u\leftarrow v$ we have:

$$\begin{aligned} & P_t \\ = \quad & \{ \quad t = u\leftarrow v \text{ , definition (12), theorem 1} \quad \} \end{aligned}$$

$$
\begin{array}{ll}
 & P_u \;\leftarrow\; [abs\,, con^{\cup}]_v^{\cup} \bullet \sqsubseteq_v \\
= & \quad \{ \quad \text{distributivity: (7) and lemma 7} \\
 & \quad\quad \text{(noting that } P_u = [abs\,, con^{\cup}]_u^{\cup} \bullet \sqsubseteq_u \text{)} \quad \} \\
 & P_u \leftarrow [abs\,, con^{\cup}]_v^{\cup} \\
= & \quad \{ \quad \text{induction hypothesis} \quad \} \\
 & [\![abs^{\cup} \bullet \sqsubseteq\,,\ [abs\,, con^{\cup}]^{\cup}]\!]_u \;\leftarrow\; [abs\,, con^{\cup}]_v^{\cup} \\
= & \quad \{ \quad \text{definition: (22)},\ t = u \leftarrow v \quad \} \\
 & [\![abs^{\cup} \bullet \sqsubseteq\,,\ [abs\,, con^{\cup}]^{\cup}]\!]_t \quad .
\end{array}
$$

□

Corollary 1 (Safety For Free, Special Case). If θ has type $t = a \leftarrow v$, where a is a type variable and v is a type expression, and is parametrically polymorphic in all the type variables in t then

$$abs_a \bullet \theta(B) \quad \dot{\sqsubseteq} \quad \theta(A) \bullet [abs\,, con^{\cup}]_v$$

where $\dot{\sqsubseteq}$ denotes the pointwise ordering $[\![\sqsubseteq\,, A]\!]_t$ on functions with range A_a.
Proof The assumption that θ has type t and is parametrically polymorphic in all the type variables in $a \leftarrow v$ means that

$$(\theta(B)\,,\, \theta(A)) \in P_{a \leftarrow v} \quad .$$

Now,

$$
\begin{array}{ll}
 & P_{a \leftarrow v} \\
= & \quad \{ \quad \text{lemma 12} \quad \} \\
 & [\![abs^{\cup} \bullet \sqsubseteq\,,\ [abs\,, con^{\cup}]^{\cup}]\!]_{a \leftarrow v} \\
= & \quad \{ \quad \text{definitions: (22) and (11)} \quad \} \\
 & abs_a^{\cup} \bullet \sqsubseteq_a \leftarrow [abs\,, con^{\cup}]_v^{\cup} \\
= & \quad \{ \quad \text{distributivity: (7) and (8)} \quad \} \\
 & (abs_a^{\cup} \leftarrow A_v) \bullet (\sqsubseteq_a \leftarrow A_v) \bullet (A_a \leftarrow [abs\,, con^{\cup}]_v^{\cup}) \\
= & \quad \{ \quad \dot{\sqsubseteq} \;=\; [\![\sqsubseteq\,, A]\!]_{a \leftarrow v} \;=\; \sqsubseteq_a \leftarrow A_v\,, \\
 & \quad\quad \text{converse: (6)} \quad \} \\
 & (abs_a \leftarrow A_v)^{\cup} \;\bullet\; \dot{\sqsubseteq} \;\bullet\; (A_a \leftarrow [abs\,, con^{\cup}]_v^{\cup}) \quad .
\end{array}
$$

Hence, using (5) and the definitions of composition and converse,

$$abs_a \bullet \theta(B) \quad \dot{\sqsubseteq} \quad \theta(A) \bullet [abs\,, con^{\cup}]_v \quad .$$

□

We conclude with a simple example. Consider the $fold$ function which, given a starting value m of some type a, a binary operator $\oslash$ of type $a \leftarrow a \times a$ and a list $[m_o, \ldots, m_{n-1}]$ of a's, evaluates $((m \oslash m_o) \oslash \ldots) \oslash m_{n-1}$. Suppose that we want to determine whether a product of numbers is divisible by k for some given positive number k. For brevity, define the function dk mapping integers to booleans by

$$dk.m \;=\; m/k$$

where m/k is read as "m is divisible by k". The function dk is the lower adjoint in a Galois connection (dk, kd) between $(\mathsf{Bool}, \Leftarrow)$ and $(\mathsf{PosInt}, /)$ where $dk \in \mathsf{Bool} \leftarrow \mathsf{PosInt}$, $kd \in \mathsf{PosInt} \leftarrow \mathsf{Bool}$ and $/$ is the is-divisible-by ordering on positive integers. Specifically, kd is defined by, for all booleans b,

$$kd.b \;=\; \textbf{if } b \textbf{ then } k \textbf{ else } 1$$

and satisfies, for all positive integers m and all booleans b,

$$dk.m \Leftarrow b \quad \equiv \quad m \,/\, kd.b \quad .$$

(Read "(m is divisible by k if b) is (m is divisible by $kd.b$)".) The pair algebra corresponding to this Galois connection relates all positive integers to the boolean $false$ and the positive integers divisible by k to the boolean $true$.

Theorem 1 predicts that the function

$$dk \;\leftarrow\; kd^{\cup} \;\leftarrow\; (kd^{\cup} \leftarrow dk \times dk) \;\leftarrow\; List.kd^{\cup}$$

is the lower adjoint in a Galois connection between the poset of monotonic functions of type

$$\mathsf{Bool} \leftarrow \mathsf{Bool} \leftarrow (\mathsf{Bool} \leftarrow \mathsf{Bool} \times \mathsf{Bool}) \leftarrow List.\mathsf{Bool}$$

(ordered pointwise by $\Leftarrow$) and monotonic functions of type

$$\mathsf{PosInt} \leftarrow \mathsf{PosInt} \leftarrow (\mathsf{PosInt} \leftarrow \mathsf{PosInt} \times \mathsf{PosInt}) \leftarrow List.\mathsf{PosInt}$$

(ordered pointwise by divisibility). The upper adjoint is the function

$$kd \;\leftarrow\; dk^{\cup} \;\leftarrow\; (dk^{\cup} \leftarrow kd \times kd) \;\leftarrow\; List.dk^{\cup} \quad .$$

Theorem 2 says that the Galois connection defines a safe abstract interpretation of the evaluation of the $fold$ function. This is made explicit by "uncurrying" the function and applying corollary 1. To be precise, assuming $fold$ has type

$$a \;\leftarrow\; a \times (a \leftarrow a \times a) \times List.a$$

the corollary states that

$$dk.ConcreteValue \;\;\Leftarrow\;\; AbstractValue$$

where

$$ConcreteValue \;\;=\;\; fold.(m\,,(\times)\,,ms)$$

and

$$AbstractValue \;=\; fold.(dk.m \;,\; (dk \leftarrow kd^{\cup} \times kd^{\cup}).(\times) \;,\; List.dk.ms) \quad .$$

(The argument $(\times)$ to $fold$ is normal arithmetic multiplication and is not to be confused with cartesian product as used in the type of $fold$.) Note that, for booleans b and c,

$$(dk \leftarrow kd^{\cup} \times kd^{\cup}).(\times).(b,c) \;=\; dk.(kd.b \times kd.c) \;=\; b \vee c \quad .$$

So the theorem predicts the (obvious) property that it is safe to evaluate whether the product of a list of numbers is divisible by k by interpreting each number as the boolean value "the number is divisible by k" and interpreting multiplication as logical disjunction. "Safe" means that a $true$ answer can be relied upon.

For a more substantial example involving a definedness test on attribute grammars, see [Bac02].

4 Conclusion

We have shown how to extend a collection of Galois connections to a Galois connection of arbitrary higher-order type. The construction is more general than any that we are aware of in the literature. In addition, we have shown that the construction is defined by a logical relation thus hopefully clarifying misunderstandings about the relationship between higher-order Galois connections, their safety properties and logical relations.

References

[Abr90] Samson Abramsky. Abstract interpretation, logical relations, and Kan extensions. *J. Logic and Computation*, 1(1):5–41, 1990.

[Bac02] K. S. Backhouse. A functional semantics of attribute grammars. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer-Verlag, 2002. Available from: http://web.comlab.ox.ac.uk/oucl/research/areas/progtools/publications.htm.

[BBH + 92] R.C. Backhouse, P. de Bruin, P. Hoogendijk, G. Malcolm, T.S. Voermans, and J. van der Woude. Polynomial relators. In M. Nivat, C.S. Rattray, T. Rus, and G. Scollo, editors, *Proceedings of the 2nd Conference on Algebraic Methodology and Software Technology, AMAST'91*, pages 303–326. Springer-Verlag, Workshops in Computing, 1992.

[Bru95] Peter J. de Bruin. *Inductive Types in Constructive Languages.* PhD thesis, Rijksuniversiteit Groningen, 1995.

[BVW92] R.C. Backhouse, T.S. Voermans, and J. van der Woude. A relational theory of datatypes. Available via World-Wide Web at `http://www.cs.nott.ac.uk/~rcb/MPC/papers` Available via anonymous ftp from `ftp.win.tue.nl` in directory `pub/math.prog.construction`, December 1992.

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unifed lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, January 1977.

[CC79] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, January 1979.

[CC94] Patrick Cousot and Radhia Cousot. Higher-order abstract interpretations (and application to comportment analysis generalizing strictness, termination, projection and per analysis of functional languages). In *Procs. ICCL'94, IEEE*, pages 95–112, 1994.

[Hoo97] Paul Hoogendijk. *A Generic Theory of Datatypes*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1997.

[HS64] J. Hartmanis and R.E. Stearns. Pair algebras and their application to automata theory. *Information and Control*, 7(4):485–507, 1964.

[HS66] J. Hartmanis and R.E. Stearns. *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, 1966.

[NNH98] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer-Verlag, 1998.

[Plo80] Gordon D. Plotkin. Lambda-definability in the full type hierarchy. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, London, 1980.

[Rey83] J.C. Reynolds. Types, abstraction and parametric polymorphism. In R.E. Mason, editor, *IFIP '83*, pages 513–523. Elsevier Science Publishers, 1983.

[Wad89] P. Wadler. Theorems for free! In *4'th Symposium on Functional Programming Languages and Computer Architecture, ACM, London*, September 1989.