Extracting program logics from abstract interpretations defined by logical relations

David A. Schmidt

Computing and Information Sciences Dept.
Kansas State University
Manhattan, KS 66506 USA

Abstract

We connect the activity of *defining* an abstract-interpretation-based static analysis with *synthesizing* its appropriate programming logic by applying logical relations as demonstrated by Abramsky. We begin with approximation relations of base type, which relate concrete computational values to their approximations, and we lift the relations to function space and upper- and lower-powerset. The resulting family's properties let us synthesize an appropriate logic for reasoning about the outcome of a static analysis. The relations need not generate Galois connections, but when they do, we show that the relational notions of soundness and completeness coincide with the Galois-connection-based notions.

Keywords: Abstract interpretation, logical relation, Galois connection, temporal logic

---

1 Introduction

Static analysis — the automated extraction of program properties — relies upon a suitably chosen *programming logic* for stating and validating the properties. For example, the static analysis of a nondeterministic state-transition system typically employs a variant of dynamic [16] or Hennessy-Milner [18] logic to state and validate properties: for states, $c \in C$:

$$
\begin{aligned}
&c \models p \text{ is given, for primitive properties, } p,\\
&c \models [f]\phi, \text{ if for all } c' \in f(c),\ c' \models \phi\\
&c \models \langle f\rangle\phi, \text{ if there exists } c' \in f(c) \text{ such that } c' \models \phi
\end{aligned}
$$

where $f : C \to \mathcal{P}(C)$ denotes a nondeterministic transition function/action.

From where does this logic arise? We can "deconstruct" the logic to discover its origin: first we untangle f from the universal/existential properties defined by $[\cdot]$

---

[1] Supported by NSF ITR-0326577.

[2] Email: `schmidt@cis.ksu.edu`

and $\langle\cdot\rangle$. Let $c \in C$ and $S \subseteq C$:

$$S \models \forall\phi,\ \text{if for all } c \in S,\ c \models \phi$$
$$S \models \exists\phi,\ \text{if there exists } c \in S \text{ such that } c \models \phi$$
$$c \models f;\phi,\ \text{if } f(c) \models \phi$$

This exposes the set domains implicit in the original logic. Now, $[f]\phi$ should be read as an abbreviation of $f;\forall\phi$.

This logic is itself an instance of another logic, where the universal quantifier quantifies disjunctions; there are conjunctions of existential quantifiers; and both domain and codomain properties of transfer functions can be described:

$$S \models \forall(\bigvee_{i<k} \phi_i),\ \text{if for all } c \in S,\ \text{there exists } j < k \text{ such that } c \models \phi_j$$
$$S \models \bigwedge_{i<k}(\exists\phi_i),\ \text{if for all } i < k,\ \text{there exists } c \in S \text{ such that } c \models \phi_i$$
$$c \models f;\phi,\ \text{if } f(c) \models \phi$$
$$f(c) \models f(\phi),\ \text{if } c \models \phi$$

This logic exposes that the set domains are lower- and upper-powerset constructions and distinguishes between function pre- and post-images. This paper will show that the last set of judgements are extracted from Plotkin-style *logical relations* for the types, $\mathcal{P}_L(\tau)$, $\mathcal{P}_U(\tau)$, and $\tau_1 \to \tau_2$ [28]; the relations themselves generate a Cousot-Cousot-style *abstract interpretation* [1,7,8]:

(i) We show how to define a static analysis based on abstract interpretation in terms of an approximation relation on base types, and we show how to lift the relation to compound types via logical relations, as first proposed by Abramsky [1].

(ii) We restate the coincidence between Galois-connection-based approximation and relational approximation regarding best approximation and soundness, and we extend the coincidence to functional completeness.

(iii) We show that every abstract domain has an *internal logic*, and we show how the logical relations generate logical operators within the internal logic.

(iv) When there are logical operators that do not fall within an abstract domain's internal logic, we show how to approximate them soundly by means of an *external logic* generated with the aid of the logical relations.

(v) We demonstrate how the generated external logic produces the above example logic.

Aside from its obvious debt to the abstract-interpretation theory of Cousot and Cousot [7,8,9,11], this paper builds on groundbreaking work by Abramsky [1], who extracted approximation relations from abstraction maps on base type and generated maps on higher type via logical relations; by Backhouse and Backhouse [4], who axiomatized many of Abramsky's results within relational algebra; and by Dams [13], who applied abstract interpretation to a rigorous development of safety

```
readInt(x)
if x>0 :
   x:= pred(x)
x:= succ(x)
writeInt(x)
```

Q: Is output pos?

A: abstractly interpret domain Int by $Sign = \{neg, zero, pos, any\}$:

$readSign(\mathtt{x})$
$\mathtt{if}\ isPos(\mathtt{x})\mathtt{:}$
$\quad \mathtt{x:=}\ pred^\sharp(\mathtt{x})$
$\mathtt{x:=}\ succ^\sharp(\mathtt{x})$
$writeSign(\mathtt{x})$

$$\text{where}\quad \begin{array}{l} succ^\sharp(pos) = pos \\ succ^\sharp(zero) = pos \\ succ^\sharp(neg) = any \\ succ^\sharp(any) = any \end{array} \quad\text{and}\quad \begin{array}{l} pred^\sharp(neg) = neg \\ pred^\sharp(zero) = neg \\ pred^\sharp(pos) = any \\ pred^\sharp(any) = any \end{array}$$

Calculate the static analysis:

$$\{zero \mapsto pos,\ neg \mapsto any,\ pos \mapsto any,\ any \mapsto any\}$$

The Question is decided only for $zero$ — the static analysis is *sound* but *incomplete*.

Fig. 1. Abstract interpretation: *computing on properties*

and liveness checking in abstract model checking.

The present paper's contribution is its use of logical relations to *generate* a static analysis — even in the absence of Galois connections — and to *synthesize* a logic appropriate for reasoning about the results of the analysis.

2 Static analysis and logical properties

Figure 1 displays a small program and a Question: Upon termination, is the output a positive integer? Rather than exhaustively test the program to answer the question, we might employ a static analysis, which in the Figure uses an *abstract domain* of sign properties, $Sign$, as approximate values for computation. When the program's transition functions, `succ` and `pred`, are abstracted to compute on $Sign$, we obtain an *abstract interpretation* of the program that can be applied to the abstract-test cases. The results, displayed in the Figure, let us conclude that an input of 0 results in a positive output, but the loss in precision within $Sign$ prevents decisions for positive, negative, and arbitrary integer inputs.[3]

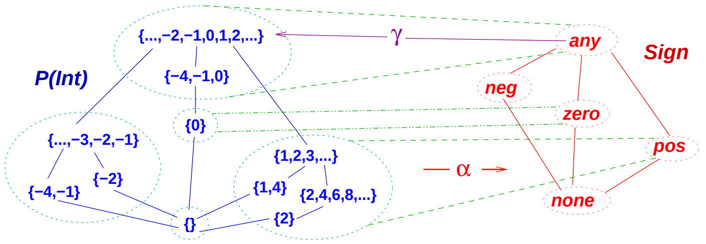
3 Galois connections

Galois connections underlie most static analyses [7,20,26]: For complete lattices, $(C, \subseteq, \cup, \cap)$ and $(A, \sqsubseteq, \sqcup, \sqcap)$, a pair of monotone maps, $\alpha : C \to A$ and $\gamma : A \to C$, define a *Galois connection*, written $C\langle\alpha,\gamma\rangle A$ for short, iff $\alpha \circ \gamma \sqsubseteq_{A\to A} id_A$ and $\gamma \circ \alpha \sqsupseteq_{C\to C} id_C$.[4] As we will see, Galois-connection structure lets us define precisely

---

[3] If we improve $Sign$ by adding the properties, $\leq zero$ and $\geq zero$, then the improved definitions of $succ^\sharp$ and $pred^\sharp$ will decide the Question for pos and neg as well.

[4] Equivalently stated, the functions α and γ form a Galois connection when, for all $c \in C$ and $a \in A$, $c \subseteq_C \gamma(a)$ iff $\alpha(c) \sqsubseteq a$. When the lattices are treated as categories and the functions are treated as functors,



$$\begin{aligned} &\gamma : \mathit{Sign} \rightarrow \mathcal{P}(\mathit{Int}) \\ &\quad \gamma(\mathit{none}) = \{\} \\ &\quad \gamma(\mathit{neg}) = \{\cdots, -3, -2, -1\} \\ &\quad \gamma(\mathit{zero}) = \{0\} \\ &\quad \gamma(\mathit{pos}) = \{1, 2, 3, \cdots\} \\ &\quad \gamma(\mathit{any}) = \mathit{Int} \end{aligned}$$

$$\begin{aligned} &\alpha : \mathcal{P}(\mathit{Int}) \rightarrow \mathit{Sign} \\ &\alpha(S) = \sqcap\{a \mid \gamma(a) \subseteq S\} \\ &\quad \text{e.g., } \alpha\{2, 4, 6, 8, ...\} = \mathit{pos} \\ &\qquad \alpha\{-4, -1, 0\} = \mathit{any} \\ &\qquad \alpha\{0\} = \mathit{zero} \\ &\qquad \alpha\{\} = \mathit{none}, \text{ etc.} \end{aligned}$$

$(\mathcal{P}(\mathit{Int}), \subseteq)\langle\alpha, \gamma\rangle(\mathit{Sign}, \sqsubseteq)$ is a *Galois connection*: γ interprets the properties in Sign, and α maps each concrete set to the property that best describes the set.

Fig. 2. Galois connection between $\mathcal{P}(\mathit{Int})$ and Sign

notions of *sound*, *most-precise*, and *complete* approximation of programs and logics.

Figure 2 shows the Galois connection usually associated with the abstraction of integers by their signs, as used in Figure 1. The Galois connection in the Figure is a "completion" of the primitive abstraction relation, $\rho_{\mathit{Sign}} \subseteq \mathit{Int} \times \mathit{Sign}$, which matches concrete values to their primitive logical properties [24].

Let $n > 0$ and define $\rho_{\mathit{Sign}} \subseteq \mathit{Int} \times \mathit{Sign}$ as follows:

$$\begin{array}{ll} -n \, \rho_{\mathit{Sign}} \, \mathit{neg} & +n \, \rho_{\mathit{Sign}} \, \mathit{pos} \\ 0 \, \rho_{\mathit{Sign}} \, \mathit{zero} & m \, \rho_{\mathit{Sign}} \, \mathit{any}, \text{ for all } m \in \mathit{Int} \end{array}$$

For example, $+3$ has property pos, because $+3 \, \rho_{\mathit{Sign}} \, \mathit{pos}$.

Let A be a complete lattice (required for static analysis [20]) and C be a (partially ordered) set. For all $c, c' \in C$, for all $a, a' \in A$, a binary relation, $\rho \subseteq C \times A$, is

(i) *U-closed* iff $c \, \rho \, a$ and $a \sqsubseteq a'$ imply $c \, \rho \, a'$

(ii) *GLB-closed* iff $c \, \rho \sqcap \{a \mid c \, \rho \, a\}$

(iii) *L-closed* iff $c \, \rho \, a$ and $c' \sqsubseteq c$ imply $c' \, \rho \, a$

(iv) *LUB-closed* iff $\sqcup\{c \mid c \, \rho \, a\} \, \rho \, a$.

U- and L-closure ensure the soundness of approximation relation ρ [9,24], and GLB- and LUB-closure ensure the existence of most precise abstractions (α) and concretizations (γ), respectively — we have that [1,4,36,38]

U-GLB-L-LUB-closed $\rho \subseteq C \times A$ *defines the Galois connection,*
$C\langle\alpha_\rho, \gamma_\rho\rangle A$, *where* $\alpha_\rho(c) = \sqcap\{a \mid c \, \rho \, a\}$ *and* $\gamma_\rho(a) = \cup\{c \mid c \, \rho \, a\}$.
Further, every Galois connection defines the U-GLB-L-LUB-closed relation,
$c \, \rho \, a$ *iff* $c \subseteq_C \gamma(a)$ *(iff* $\alpha(c) \sqsubseteq a$*).*

---

the Galois connection defines an adjunction [1].

Every static analysis is based on an approximation relation, and most such relations possess U-GLB-L-LUB-closure (but not all, e.g., [9,23,41]). Relation $\rho_{\mathit{Sign}} \subseteq \mathit{Int} \times \mathit{Sign}$ above (where Int is discretely ordered) is U-L-GLB-closed but not LUB-closed. In this case, the Galois connection in Figure 2 can be constructed by completing Int to $\mathcal{P}(\mathit{Int})$. We do so by "lifting" ρ_{Sign} to *logical relation* $\rho_{L(\mathit{Sign})}$, as explained in the section that follows.

4 Logical relations and Galois connections

Approximation relations on compound types are correctly defined by logical relations [28]. For base types, b, function types, and lower and upper powerset types,

$$\tau \ ::= \ b \mid \tau_1 \rightarrow \tau_2 \mid L(\tau) \mid U(\tau)$$

we define these domains:

- D_b is given (e.g., Int and Sign)
- $D_{\tau_1 \rightarrow \tau_2} = D_{\tau_1} \rightarrow D_{\tau_2}$, the partially ordered set of monotone functions from D_{τ_1} to D_{τ_2}. (Monotonicity suffices for static analysis [7].)
- $D_{L(\tau)} = \mathcal{P}_L(D_\tau)$, a *lower powerset* of D_τ, which is a collection of downclosed subsets of D that includes all $\downarrow d$ for all $d \in D$, partially ordered by $\subseteq$, and closed under $\cap$.[5] (This includes $\mathcal{P}_\downarrow(D)$, the collection of *all* downclosed subsets of D.)
- $D_{U(\tau)} = \mathcal{P}_U(D_\tau)$, an *upper powerset* of D_τ, which is a collection of upclosed subsets of D that includes all $\uparrow d$ for all $d \in D$, partially ordered by $\supseteq$, and closed under $\cup$. (This includes $\mathcal{P}_\uparrow(D)$, the collection of *all* upclosed subsets of D.)

The family of approximating logical relations is defined as usual, for $\rho_\tau \subseteq C_\tau \times A_\tau$:

ρ_b is given, for base type b (e.g., $\rho_{\mathit{Sign}} \subseteq \mathit{Int} \times \mathit{Sign}$)

$f \, \rho_{\tau_1 \rightarrow \tau_2} \, f^\sharp$ iff for all $c \in C_{\tau_1}$ and $a \in A_{\tau_1}, c \, \rho_{\tau_1} \, a$ implies $f(c) \, \rho_{\tau_2} \, f^\sharp(a)$

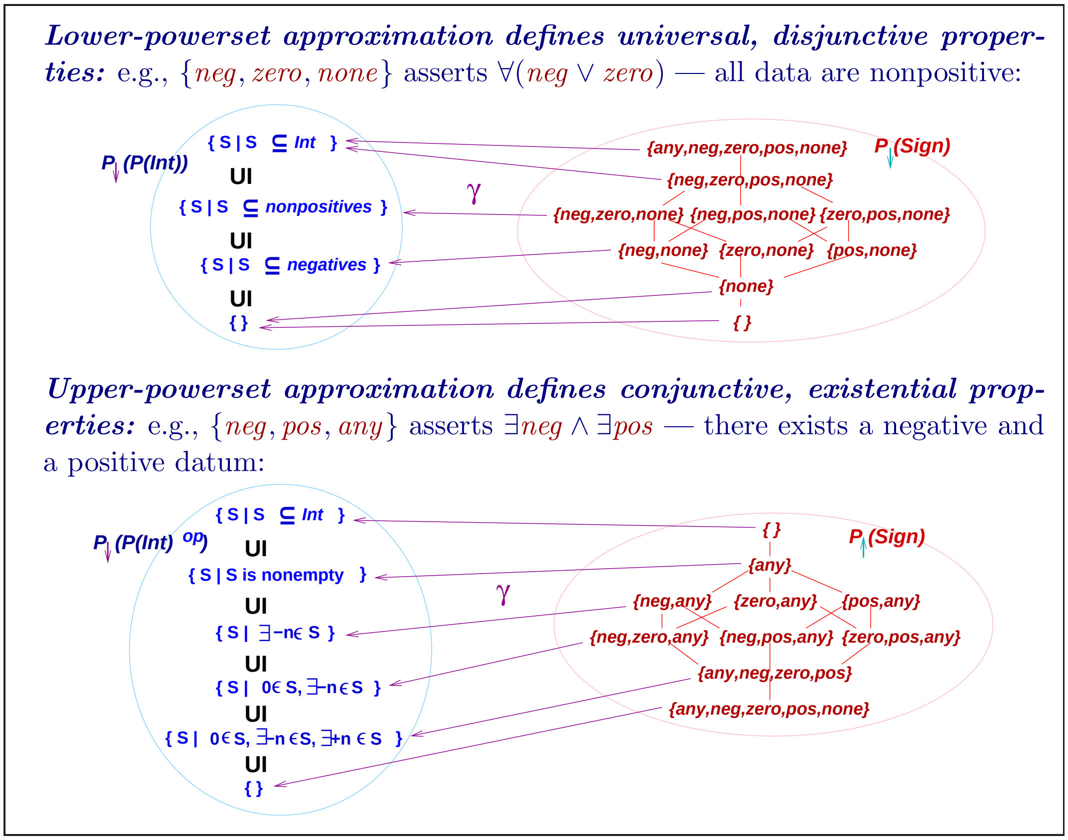
$S \, \rho_{L(\tau)} \, T$ iff for all $c \in S \in C_{L(\tau)}$, there exists $a \in T \in A_{L(\tau)}$ such that $c \, \rho_\tau \, a$

$S \, \rho_{U(\tau)} \, T$ iff for all $a \in T \in A_{U(\tau)}$, there exists $c \in S \in C_{U(\tau)}$ such that $c \, \rho_\tau \, a$

The definitions read as expected, e.g., $f \, \rho_{\tau_1 \rightarrow \tau_2} \, f^\sharp$ asserts that function f is approximated by function $f^\sharp$ because arguments related by an approximation relation map to answers related by an approximation relation. $S \, \rho_{L(\tau)} \, T$ defines an *overapproximation* relation: S is overapproximated by T because every element of S has an approximant in T. Dually, $S \, \rho_{U(\tau)} \, T$ defines an *underapproximation* relation, because every element in T is witnessed by a concrete element in S. See Figure 3 for examples of set approximation, which propose logical readings of the relations on lower and upper powersets [27,39], reminiscent of the modal language proposed by Winskel [42], adapted to approximation. The lower-powerset approximation is an example of Abramsky's *safety adjunction*, and the upper-powerset approximation is an example of his *liveness adjunction* [1].

---

[5] $S \subseteq D$ is *downclosed* if $S = \{d' \in D \mid \exists d \in S, d' \sqsubseteq_D d\}$; for $d \in D$, $\downarrow d = \{d' \in D \mid d' \sqsubseteq_D d\}$; $S \subseteq D$ is *upclosed* if $S = \{d' \in D \mid \exists d \in S, d \sqsubseteq_D d'\}$; and $\uparrow d = \{d' \in D \mid d \sqsubseteq_D d'\}$.



Lower-powerset approximation defines universal, disjunctive properties: e.g., $\{neg, zero, none\}$ asserts $\forall(neg \vee zero)$ — all data are nonpositive:

Upper-powerset approximation defines conjunctive, existential properties: e.g., $\{neg, pos, any\}$ asserts $\exists neg \wedge \exists pos$ — there exists a negative and a positive datum:

Fig. 3. Approximation by powersets

4.1 Closure properties of logical relations

Proposition 4.1 *For $\rho_\tau \subseteq C_\tau \times A_\tau$,*

(i) *$\rho_{L(\tau)}$ and $\rho_{U(\tau)}$ are L-closed; $\rho_{\tau' \to \tau}$ is L-closed iff ρ_τ is.*

(ii) *$\rho_{L(\tau)}$ and $\rho_{U(\tau)}$ are U-closed; $\rho_{\tau' \to \tau}$ is U-closed iff ρ_τ is.*

(iii) *If ρ_τ is U-GLB-closed, then so is $\rho_{L(\tau)}$; $\rho_{\tau' \to \tau}$ is U-GLB-closed iff ρ_τ is.*

(iv) *If ρ_τ is L-LUB-closed, then so is $\rho_{U(\tau)}$; $\rho_{\tau' \to \tau}$ is L-LUB-closed iff ρ_τ is.*

Missing are assurances of LUB-closure for $\rho_{L(\tau)}$ and GLB-closure for $\rho_{U(\tau)}$; these depend on the specific powersets used. But we do have [36]

- For any lower powerset, PA, of type $\mathcal{P}_L(\tau)$, $\rho_{L(\tau)} \subseteq \mathcal{P}_\downarrow(C_\tau) \times PA$ is LUB-closed.
- For any upper powerset, PC, of type $\mathcal{P}_U(\tau)$, $\rho_{U(\tau)} \subseteq PC \times \mathcal{P}_\uparrow(A_\tau)$, is GLB-closed.

Using these results, we can build Galois connections from the logical relations, as needed. One standard trick is completing a U-GLB-closed relation, like $\rho_{Sign} \subseteq Int \times Sign$, where Int is discretely ordered, to U-GLB-L-LUB-closed $\rho_{L(Sign)} \subseteq \mathcal{P}(Int) \times triv(Sign)$, where lower powerset $triv(Sign) = (\{\downarrow a \mid a \in Sign\}, \subseteq)$ is order-isomorphic to $Sign$. This produces the Galois connection in Figure 2.

Functional soundness: $f^\sharp : A \to A$ is *sound* for $f : C \to C$ iff

$$\alpha \circ f \sqsubseteq f^\sharp \circ \alpha \quad \text{iff} \quad f \circ \gamma \sqsubseteq \gamma \circ f^\sharp$$

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[dd, "\alpha"'] & f(S) \arrow[d, "\alpha"] \\
 & \sqsupseteq \\
\alpha(S) \arrow[r, "f^{\#}"] & \bullet
\end{tikzcd}
\qquad
\begin{tikzcd}
\gamma(a) \arrow[r, "f"] & \bullet \\
 & \sqsubseteq \\
a \arrow[uu, "\gamma"] \arrow[r, "f^{\#}"] & f^{\#}(a) \arrow[u, "\gamma"']
\end{tikzcd}$$

α and γ act as *semi-homomorphisms.*

Example: For $succ : \mathcal{P}(Int) \to \mathcal{P}(Int)$, $succ(S) = \{n+1 \mid n \in S\}$, $succ^\sharp$ is sound for $succ$.

γ***(forwards)-completeness***:

$$f \circ \gamma = \gamma \circ f^\sharp$$

$$\begin{tikzcd}
\gamma(a) \arrow[r, "f"] & \bullet \\
a \arrow[u, "\gamma"] \arrow[r, "f^{\#}"] & f^{\#}(a) \arrow[u, "\gamma"']
\end{tikzcd}$$

γ is a homomorphism from A to C: it preserves $f^\sharp$ as f.

α***(backwards)-completeness***:

$$\alpha \circ f = f^\sharp \circ \alpha$$

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[d, "\alpha"'] & f(S) \arrow[d, "\alpha"] \\
\alpha(S) \arrow[r, "f^{\#}"] & \bullet
\end{tikzcd}$$

α is a homomorphism from C to A: it preserves f as $f^\sharp$.

Examples: For $negate : \mathcal{P}(Int) \to \mathcal{P}(Int)$, $negate(S) = \{-n \mid n \in S\}$ and $negate^\sharp(neg) = pos$, $negate^\sharp(pos) = neg$, etc., $negate^\sharp$ is α- and γ-complete for $negate$; in contrast, $succ^\sharp$ is neither α- nor γ-complete for $succ$; finally, $square^\sharp$ is α- but not γ-complete for $square(S) = \{n^2 \mid n \in S\}$.

Fig. 4. Functional soundness and completeness expressed as semi- and full homomphisms

5 Functional soundness and completeness

Figure 1 showed that the concrete state-transition functions, $succ : Int \to Int$ and $pred : Int \to Int$, must be abstracted to $succ^\sharp : Sign \to Sign$ and $pred^\sharp : Sign \to Sign$ to conduct a static analysis.

A function, $f : C_\tau \to C_\tau$, is *soundly abstracted* by $f^\sharp : A_\tau \to A_\tau$, if $f\ \rho_{\tau\to\tau}\ f^\sharp$. This relational definition coincides with the classical definition of functional soundness from abstract interpretation [1,8,15]: If $f\ \rho_{\tau\to\tau}\ f^\sharp$ is U-GLB-L-LUB-closed, then the following are equivalent:

- $f\ \rho_{\tau\to\tau}\ f^\sharp$
- $\alpha_{\rho_\tau} \circ f \sqsubseteq_{C_\tau \to A_\tau} f^\sharp \circ \alpha_{\rho_\tau}$
- $f \circ \gamma_{\rho_\tau} \sqsubseteq_{A_\tau \to C_\tau} \gamma_{\rho_\tau} \circ f^\sharp$

α_{ρ_τ} and γ_{ρ_τ} are semi-homomorphisms with respect to f and $f^\sharp$; see Figure 4.

Given Galois connection, $C_\tau\langle\alpha_{\rho_\tau}, \gamma_{\rho_\tau}\rangle A_\tau$, the most precise, sound abstraction of $f : C_\tau \to C_\tau$ with respect to the Galois connection is $f^\sharp_{best} = \alpha_{\rho_\tau} \circ f \circ \gamma_{\rho_\tau} = \sqcap\{f^\sharp \mid f\ \rho_{\tau\to\tau}\ f^\sharp\}$ [8]. As indicated by the last equality and Proposition 4.1, if ρ_τ lacks U-GLB-closure, then there is no Galois connection and no most-precise abstraction.

Exact preservation of f's mappings within A by $f^\sharp$ yields *functional completeness*; it is characterized in two independent ways:

(i) When α acts as a homomorphism from f to $f^\sharp$, then $f^\sharp$ is α*(backwards)-*

complete for f [8,15].

(ii) When γ acts as a homomorphism from $f^\sharp$ to f, then $f^\sharp$ is γ *(forwards)-complete* for f [14].

See Figure 4. If some $f^\sharp$ is (α- or γ-) complete for f, then so is $f^\sharp_{best}$ [15]. The consequences of completeness are developed in the next section.

There is one important example of soundness: For a nondeterministic state transition system, $(\Sigma,\ R \subseteq \Sigma \times \Sigma)$, we characterize transition relation R as $f_R : \Sigma \rightarrow \mathcal{P}(\Sigma)$. Say there is an approximation relation, $\rho_{State} \subseteq \Sigma \times A$, and an abstract transition system, $(A,\ R^\sharp \subseteq A \times A)$, as used in "abstract model checking" [5,13]. Using the standard definition of simulation [18]: *$R^\sharp$ ρ_{State}-simulates R iff for all $c, c' \in \Sigma, a \in A$,*

$$c\, \rho_{State}\, a \textit{ and } c\, R\, c' \textit{ imply there exists } a' \in A \textit{ such that } a\, R^\sharp\, a' \textit{ and } c'\, \rho_{State}\, a',$$

we have that, if $f_R : \Sigma \rightarrow \mathcal{P}(\Sigma)$ and $f_{R^\sharp} : A \rightarrow PA$ are monotone, where PA is a lower powerset, then $R^\sharp$ ρ_{State}-simulates R iff $f_R\ \ \rho_{State \rightarrow L(State)}\ \ f_{R^\sharp}$.

A dual simulation, where $R^\flat$ ρ^{-1}_{State}-simulates R, is characterized with upper powersets as $f_R\ \ \rho_{State \rightarrow U(State)}\ \ f_{R^\flat}$ (cf. "liveness analysis" [1,13]).

6 Program logic

Given an abstraction, $\rho \subseteq C \times A$, that generates a static analysis (e.g., Figures 1 and 2), we require an assertion language to define the properties that the static analysis must check and validate for program correctness or code improvement.

The simplest assertion language is merely the elements of A itself (e.g., *Sign*, as used in Figure 1), and its "logical semantics" is $[\![a]\!]_\rho = \{c \mid c\, \rho\, a\}$, for each $a \in A$.

One immediate benefit is that every $f^\sharp : A \rightarrow A$ that is sound for $f : C \rightarrow C$ is also a sound *postcondition transformer* for f with respect to the assertion language, A: for all $a \in A$ and $c \in C$:

$$c \in [\![a]\!]_\rho \text{ implies } f(c) \in [\![f^\sharp(a)]\!]_\rho$$

Indeed, $f^\sharp_{best}$ is the *strongest* postcondition transformer for f in A.

A typical static analysis uses such A and $f^\sharp$ to compute postconditions for an abstracted program. At the program's exit (or at a key internal program point), there is some assertion to check. Say the assertion is stated as $a_{out} \in A$. Using $c_{in}\, \rho\, a_{in}$, the static analysis computes $f^\sharp(a_{in})$ and checks whether $f^\sharp(a_{in}) \sqsubseteq a_{out}$ holds true. If yes, then $f(c_{in}) \in [\![a_{out}]\!]_\rho$ holds by U-closure. This is how data-flow analysis, type checking, and program validation are usually implemented.

The previous technique is sound but "incomplete" (cf. Figure 1). We would prefer a decision procedure: Say that $\rho \subseteq C \times A$ is U-GLB-closed and define $\alpha_\rho : C \rightarrow A$ as $\alpha_\rho(c) = \sqcap \{a \mid c\, \rho\, a\}$, that is, α_ρ maps c to its best approximant. We say that $f^\sharp$ *ρ-decides* f if, for all $c \in C$, $a \in A$,

$$f^\sharp(\alpha_\rho(c)) \sqsubseteq a \text{ iff } f(c) \in [\![a]\!]_\rho$$

This means all f's A-logical properties can be decided by $f^\sharp$ within A. When ρ defines a Galois connection, decidability coincides with α_ρ-functional completeness:

Proposition 6.1 *For U-GLB-L-LUB-closed ρ, $f^\sharp$ ρ-decides f iff $f^\sharp$ is α_ρ-complete for f.*

This is why α-completeness is important in practice.

6.1 Internal logic

Assertion language A possesses an *internal logic* in the sense that there exist logical connectives that are expressed as functions on A. Here is an important example.

If $\rho \subseteq C \times A$ is U-GLB closed, then $\sqcap : A \times A \to A$ is logical conjunction in A: for all $c \in C$, $a_0, a_1 \in A$:

$$c \in [\![a_0 \sqcap a_1]\!]_\rho \text{ iff } c \in [\![a_0]\!]_\rho \text{ and } c \in [\![a_1]\!]_\rho$$

This expands the assertion language based on A to

$$\phi ::= a \mid \phi \sqcap \phi, \quad \text{for all } a \in A,$$

and we can employ the usual inference rules for conjunction. For example, in Figure 2, $\sqcap$ is conjunction, and we can assert, say, $2 \in [\![\mathit{any} \sqcap \mathit{pos}]\!]_{\rho_{Sign}}$. Most important, when a logical connective exists in A's internal logic, we can soundly check it within A: For conjunction, if a static analysis verifies that $a_{out} \sqsubseteq \phi_1 \sqcap \phi_2$, then we safely conclude, for all $c \,\rho\, a_{out}$, that $c \in [\![\phi_1]\!]_\rho$ and $c \in [\![\phi_2]\!]_\rho$.

Not all propositional connectives exist: For Figure 2, disjunction fails, because $0 \in [\![\mathit{any}]\!]_{\rho_{Sign}} = [\![\mathit{neg} \sqcup \mathit{pos}]\!]_{\rho_{Sign}}$, yet $0 \notin [\![\mathit{neg}]\!]_{\rho_{Sign}}$ and $0 \notin [\![\mathit{pos}]\!]_{\rho_{Sign}}$.[6] Thus, $\mathit{zero} \sqsubseteq \mathit{neg} \sqcup \mathit{pos}$ does *not* imply $0 \in [\![\mathit{neg}]\!]_{\rho_{Sign}}$ or $0 \in [\![\mathit{pos}]\!]_{\rho_{Sign}}$.

The previous definition of conjunction is somewhat informal; a more precise statement reads

$$[\![\sqcap(a_0, a_1)]\!]_\rho = \mathit{and}([\![a_0]\!]_\rho, [\![a_1]\!]_\rho)$$

where $\mathit{and} : \mathcal{P}(C) \times \mathcal{P}(C) \to \mathcal{P}(C)$ is $\cap$. This makes clearer that the connective, and, is expressed in A by $\sqcap$.

For k-ary logical connective, $f : \mathcal{P}(C)^k \to \mathcal{P}(C)$, and k-ary function $f^\sharp : A^k \to A$, we say that $f^\sharp$ *ρ-expresses* f if

$$[\![f^\sharp(a_i)_{i<k}]\!]_\rho = f([\![a_i]\!]_\rho)_{i<k}$$

(See the conjunction example, where $f = \mathit{and}$ and $f^\sharp = \sqcap$.)

We connect this notion to functional completeness: For $\rho \subseteq C \times A$, define $\overline{\rho} \subseteq \mathcal{P}(C) \times A$ as $S \,\overline{\rho}\, a$ iff for all $c \in S$, $c \,\rho\, a$.[7] $\overline{\rho}$ is L-LUB-closed, hence $\gamma_{\overline{\rho}} : A \to \mathcal{P}(C)$ is $\gamma_{\overline{\rho}}(a) = \cup\{S \mid S \,\overline{\rho}\, a\} = \{c \mid c \,\rho\, a\} = [\![a]\!]_\rho$.

Proposition 6.2 *When $\rho \subseteq C \times A$ is U-GLB-closed, $f^\sharp : A \to A$ ρ-expresses $f : \mathcal{P}(C) \to \mathcal{P}(C)$ iff $f^\sharp$ is $\gamma_{\overline{\rho}}$-complete for f.*

This is why γ-completeness is important in practice.

---

[6] If disjunction would exist in Sign, it must equal $\sqcup$.

[7] This is the trick described at the end of Section 4 for "lifting" a relation to make it L-LUB-closed.

7 Logical relations generate logical connectives

Starting from base type, τ, and approximation relation, $\rho_\tau \subseteq C_\tau \times A_\tau$, we use the logical relations on compound types to generate logical operators in assertion language A_τ.

Please review the definition of lower powerset from the start of Section 4; recall, for a concrete lower powerset $\mathcal{P}_L(C_\tau)$ and an abstract lower powerset $\mathcal{P}_L(A_\tau)$, for downclosed sets $S \in \mathcal{P}_L(C_\tau)$ and $T \in \mathcal{P}_L(A_\tau)$, that

$$S\,\rho_{L(\tau)}\,T \text{ iff for all } c \in S, \text{ there exists } a \in T \text{ such that } c\,\rho_\tau\,a$$

Downclosed sets in $\mathcal{P}_L(A_\tau)$ might be written as expressions, $\downarrow\!\{a_i\}_{i<k}$. We treat $\downarrow$ as if it were a k-ary logical connective for the a_is: $\downarrow\!\{\cdot\} : {A_\tau}^k \to \mathcal{P}_L(A_\tau)$, defining its semantics from the logical relation:

$$\begin{aligned}\llbracket \downarrow\!\{a_i\}_{i<k} \rrbracket_{\rho_{L(\tau)}} &= \{S' \in \mathcal{P}_L(C_\tau) \mid \text{for all } c \in S', \text{ there exists } j < k \text{ such that } c \in \llbracket a_j \rrbracket_{\rho_\tau}\} \\ &= f_L\{\llbracket a_i \rrbracket_{\rho_\tau}\}_{i<k},\end{aligned}$$

where $f_L : \mathcal{P}(C_\tau)^k \to \mathcal{P}(\mathcal{P}_L(C_\tau))$ is defined

$$f_L\{S_i\}_{i<k} = \{S' \in \mathcal{P}_L(C_\tau) \mid \text{for all } c \in S', \text{ there exists } j < k \text{ such that } c \in S_j\}$$

By definition, $\downarrow$ $\rho_{L(\tau)}$-expresses f_L. What's more, we can use $\downarrow$ to ρ_τ-express disjunction: Define

$$\begin{aligned}c \in \llbracket \textstyle\bigvee_{i<k}\{a_i\} \rrbracket_{\rho_\tau} \quad &\text{iff } \downarrow\!c \in \llbracket \downarrow\!\{a_i\}_{i<k} \rrbracket_{\rho_{L(\tau)}} \\ &\text{iff there exists some } j < k \text{ such that } c \in \llbracket a_j \rrbracket_{\rho_\tau}\end{aligned}$$

This requires that ρ_τ be U-L-closed. The use of a lower powerset to express disjunction is known as the *disjunctive completion* of ρ_τ, where $\mathcal{P}_L(A) = \mathcal{P}_\downarrow(A)$ [15].

We can soundly check disjunction in A_τ: we check that $\downarrow a \sqsubseteq \downarrow\!\{a_i\}_{i<k}$, that is, we check whether there exists some $j < k$ such that $a \sqsubseteq a_j$; this implies $c \in \llbracket \bigvee_{i<k}\{a_i\} \rrbracket_{\rho_\tau}$, for all $c\,\rho_\tau\,a$. This is hardly a surprise, but it shows that one must steer to lower-powerset constructions to express disjunction in a static analysis.

Dually, we use the logical relation on upper powersets to express conjunction (when ρ_τ is not already U-GLB-closed):

$$\llbracket \uparrow\!\{a_i\}_{i<k} \rrbracket_{\rho_{U(\tau)}} = f_U\{\llbracket a_i \rrbracket_{\rho_\tau}\}_{i<k}, \text{ where } f_U : \mathcal{P}(C_\tau)^k \to \mathcal{P}(\mathcal{P}_U(C_\tau)) \text{ is defined}$$

$$f_U\{S_i\}_{i<k} = \{S' \in \mathcal{P}_U(C_\tau) \mid \text{for all } i < k, \text{ there exists } c \in S' \text{ such that } c \in S_i\}$$

By definition, $\uparrow$ $\rho_{U(\tau)}$-expresses f_U, and we define conjunction in A_τ as

$$c \in \llbracket \textstyle\bigwedge_{i<k}\{a_i\} \rrbracket_{\rho_\tau} \text{ iff } \uparrow\!c \in \llbracket \uparrow\!\{a_i\}_{i<k} \rrbracket_{\rho_{U(\tau)}} \text{ iff for all } i < k, c \in \llbracket a_i \rrbracket_{\rho_\tau}$$

The logical relation for $\tau_1 \to \tau_2$ does not readily surrender a logical connective. From

$$f\,\rho_{\tau_1\to\tau_2}\,f^\sharp \text{ iff for all } c \in C_{\tau_1}, a \in A_{\tau_1}, c\,\rho_{\tau_1}\,a \text{ implies } f(c)\,\rho_{\tau_2}\,f^\sharp(a)$$

we define merely a higher-order constant,

$$
\begin{aligned}
[\![f^\sharp]\!]_{\rho_{\tau_1 \rightarrow \tau_2}} = \{f \in C_{\tau_1} \rightarrow C_{\tau_2} \mid \ & \text{for all } c \in C_{\tau_1}, a \in A_{\tau_1}, \\
& c \in [\![a]\!]_{\rho_{\tau_1}} \text{ implies } f(c) \in [\![f^\sharp(a)]\!]_{\rho_{\tau_2}}\}
\end{aligned}
$$

We must work to extract a logical connective for ρ_{τ_1} and one for ρ_{τ_2}. For the latter, we propose the postimage function, $post_f : \mathcal{P}(C_{\tau_1}) \rightarrow \mathcal{P}(C_{\tau_2})$, which we hope to express by some $f^\sharp$:

$$[\![f^\sharp(a)]\!]_{\rho_{\tau_2}} = post_f [\![a]\!]_{\rho_{\tau_1}}, \text{where } post_f(S) = \{f(c) \in C_{\tau_2} \mid c \in S\}$$

By Proposition 6.2, we know that an $f^\sharp : A \rightarrow A$ ρ-expresses $post_f$ iff $f^\sharp$ is $\gamma_{\overline{\rho}}$-complete for $post_f$.

A logical connective that defines function preimage is defined as

$$[\![f^\sharp_{pre}; a]\!]_{\rho_{\tau_1}} = \widetilde{pre}_f [\![a]\!]_{\rho_{\tau_2}}, \text{where } \widetilde{pre}_f(S) = \{c \in C_{\tau_1} \mid f(c) \in S\}$$

Say we have some $f^\sharp : A \rightarrow A$ such that $f \; \rho_{\tau_1 \rightarrow \tau_2} \; f^\sharp$. To express $\widetilde{pre}_f : \mathcal{P}(C) \rightarrow \mathcal{P}(C)$, we want some $f^\sharp_{pre} : A \rightarrow \mathcal{P}_L(A_\tau)$, and the obvious candidate is

$$f^\sharp_{pre}(a) = \{a' \mid f^\sharp(a') \sqsubseteq a\}$$

If ρ_{τ_2} is U-closed, then we have soundness:[8] $\widetilde{pre}_f \; \rho_{\tau_1 \rightarrow L(\tau_2)} \; f^\sharp_{pre}$.

Proposition 7.1 *For* $f : C_\tau \rightarrow C_\tau$ *and* $f^\sharp : A_\tau \rightarrow A_\tau$ *if* ρ_τ *is U-GLB-closed and* $f^\sharp$ *is* α_{ρ_τ}*-complete for* f*, then* $f^\sharp_{pre}$ $\rho_{L(\tau)}$*-expresses* $\widetilde{pre}_f$*.*

When $f^\sharp_{pre}$ $\rho_{L(\tau)}$-expresses f_{pre}, we check $a' \in f^\sharp_{pre}(a)$, that is, $f^\sharp(a') \sqsubseteq a$, to validate that $c' \in \widetilde{pre}_f [\![a]\!]_{\rho_\tau}$, for all $c' \; \rho_\tau \; a'$.

8 External logics

Returning to the example in Figures 1 and 2, we see that neither $succ^\sharp$ and $pred^\sharp$ are α- or γ-complete for their respective concrete functions. So, we cannot express the $post_f$ and $\widetilde{pre}_f$ connectives, for $f \in \{succ, pred\}$, and soundly check them within $Sign$.

This situation is the rule, rather than the exception — it is almost impossible to define an abstract domain that admits completeness for all the transition functions embedded in a program. For this reason, we must study how to define a less precise, "external" logic for A that admits sound checking of logical operators that might not be expressible in A's internal logic.

Figure 5 displays the logic we have in mind, which consists of the operators extracted from the logical relations.

Program properties are defined by the judgements, e.g., $2 \models_{Sign} pos$, $succ(2) = 3 \models_{Sign} succ(pos)$, $\{0, 3\} \models_{L(Sign)} succ(pos) \vee zero$, $0 \models_{Sign} succ; pos$, and so on.

To check $\models_\tau$ via an abstract interpretation, we must

- supply an abstract domain, A_τ, for each concrete domain, C_τ
- supply $f^\sharp : A_{\tau_1} \rightarrow A_{\tau_2}$ for each concrete transition function, $f : C_{\tau_1} \rightarrow C_{\tau_2}$, such that $f \; \rho_{\tau_1 \rightarrow \tau_2} \; f^\sharp$.

---

[8] In Abramsky's terminology [1], $f^\sharp_{pre}$ defines a *safety relation*.

Types: $\tau ::= b \mid L(\tau) \mid U(\tau) \mid \tau_1 \to \tau_2$

Typed function symbols: $f : \tau_1 \to \tau_2$

Assertions: $\phi ::= a \mid \bigvee_{i<k} \phi_i \mid \bigwedge_{i<k} \phi_i \mid f(\phi) \mid f;\phi$

Judgement typing:

$$a : b \qquad \frac{\phi_i : \tau,\ \text{for all } i < k}{\bigvee\limits_{i<k} \phi_i : L(\tau)} \qquad \frac{\phi_i : \tau,\ \text{for all } i < k}{\bigwedge\limits_{i<k} \phi_i : U(\tau)}$$

$$\frac{f : \tau_1 \to \tau_2 \quad \phi : \tau_1}{f(\phi) : \tau_2} \qquad \frac{f : \tau_1 \to \tau_2 \quad \phi : \tau_2}{f;\phi : \tau_1}$$

Concrete judgements: have form, $c \models_\tau \phi$, where $c \in C_\tau$ and $\phi : \tau$

$c \models_b a$ is given by $\rho_b \subseteq C_b \times A_b$, e.g., $n \models_{\mathit{Sign}} a$ if $n\, \rho_{\mathit{Sign}}\, a$

$S \models_{L(\tau)} \bigvee_{i<k} \phi_i$, if for all $c \in S$, there exists $j < k$ such that $c \models_\tau \phi_j$

$S \models_{U(\tau)} \bigwedge_{i<k} \phi_i$, if for all $i < k$, there exists $c \in S$ such that $c \models_\tau \phi_i$

$c \models_{\tau_2} f(\phi)$, if there exists $c' \in C_{\tau_1}$ such that $c' \models_{\tau_1} \phi$ and $f(c') = c$,

for $f \in C_{\tau_1} \to C_{\tau_2}$

$c \models_{\tau_1} f;\phi$, if $f(c) \models_{\tau_2} \phi$, for $f \in C_{\tau_1} \to C_{\tau_2}$

Fig. 5. Concrete external logic based on logical relations

Abstract judgements: have form, $a \models^{\mathcal{A}}_\tau \phi$, where $a \in A_\tau$ and $\phi : \tau$

$a \models^{\mathcal{A}}_b a'$, if $a \sqsubseteq_b a'$, for $a, a' \in A_b$ (e.g., $\mathit{pos} \sqsubseteq_{\mathit{Sign}} \mathit{any}$)

$T \models^{\mathcal{A}}_{L(\tau)} \bigvee_{i<k} \phi_i$, if for all $a \in T$, there exists $j < k$ such that $a \models^{\mathcal{A}}_\tau \phi_j$

$T \models^{\mathcal{A}}_{U(\tau)} \bigwedge_{i<k} \phi_i$, if for all $i < k$, there exists $a \in T$ such that $a \models^{\mathcal{A}}_\tau \phi_i$

$a \models^{\mathcal{A}}_{\tau_2} f(\phi)$, if *... to come ...*

$a \models^{\mathcal{A}}_{\tau_1} f;\phi$, if $f^\sharp(a) \models^{\mathcal{A}}_{\tau_2} \phi$, for $f^\sharp \in A_{\tau_1} \to A_{\tau_2}$

Fig. 6. Abstract external logic

Given the output, $a_{out} \in A_\tau$, of a program's static analysis, we attempt to validate judgements of form, $a_{out} \models^{\mathcal{A}}_\tau \phi$, where abstract judgements based on $\models^{\mathcal{A}}_\tau$ are defined in Figure 6. We require that $\models^{\mathcal{A}}_\tau$ is *sound for* $\models_\tau$: for all ϕ and $a \in A_\tau$,

$$a \models^{\mathcal{A}}_\tau \phi \text{ implies } c \models_\tau \phi, \text{ for all } c\, \rho_\tau\, a$$

When the above implication is strengthened to an equivalence, we have a form of logical completeness known as *best preservation* [11,34]: for all $a \in A_\tau$,

$$a \models^{\mathcal{A}}_\tau \phi \text{ iff } c \models_\tau \phi, \text{ for all } c\, \rho_\tau\, a$$

Another form of completeness is stated in terms of concrete values and is known as *strong preservation* [29]: for all $c \in C_\tau$,

$$c \models_\tau \phi \text{ iff there exists } a \in A_\tau \text{ such that } a \models^{\mathcal{A}}_\tau \phi \text{ and } c\,\rho_\tau\, a$$

The two completeness forms are independent [14]. Returning to Figures 5 and 6, we have this result:

Theorem 8.1 *For all τ, $\models^{\mathcal{A}}_\tau$ in Figure 6 is sound for $\models_\tau$ in Figure 5.*

Missing from Figure 6 is a judgement form for $f(\phi)$, the postimage judgement. The reason is that the naive formulation, namely, $f^\sharp(a) \models^{\mathcal{A}}_{\tau_2} f(\phi)$, if $a \models^{\mathcal{A}}_{\tau_1} \phi$, for $f\,\rho_{\tau_1 \to \tau_2}\, f^\sharp$, is *unsound*. For example, $\mathit{any} = \mathit{pred}^\sharp(\mathit{pos}) \models^{\mathcal{A}}_{\mathit{Sign}} \mathit{pred}(\mathit{pos})$. Since $-2\,\rho_{\mathit{Sign}}\, \mathit{any}$, the abstract judgement appears to imply that $-2 \models_{\mathit{Sign}} \mathit{pred}(\mathit{pos})$, which fails. The problem is that $\mathit{pred}^\sharp$ overestimates the postimage defined by $\mathit{post}_{\mathit{pred}}$, whereas the judgement, $f^\sharp(\mathit{pos}) \models^{\mathcal{A}}_{\mathit{Sign}} \mathit{pred}(\mathit{pos})$ requires an $f^\sharp$ that *underestimates* it.

There is a repair, but it is not trivial [35]: First, treat a concrete transition function, f, to have arity, $f : C_1 \to \mathcal{P}(C_2)$.[9] Then, define $f^{-1} : C_2 \to \mathcal{P}(C_1)$ as $f^{-1}(c) = \{d \in C_1 \mid c \in f(d)\}$. This means $(f^{-1})^{-1} = f$, and more importantly, that $\mathit{post}_f = \mathit{pre}_{f^{-1}}$ [22]. The preimage function, $\mathit{pre}_g : \mathcal{P}(C_1) \to \mathcal{P}(C_2)$, for $g : C_2 \to \mathcal{P}(C_1)$, is defined

$$\mathit{pre}_g(S) = \{c \mid g(c) \cap S \neq \emptyset\}$$

Recall from Figure 3 that the upper-powerset construction defines an abstract domain of sets that *witness* concrete values. For $S\,\rho_{U(\tau)}\, T$, the set, $T = \{a_0, a_1, \cdots, a_i, \cdots\} \in \mathcal{P}_U(A)$, asserts existence of concrete values, $\{c_0, c_1, \cdots, c_i, \cdots\} \subseteq S \in \mathcal{P}(C)$, such that $c_i\,\rho\, a_i$, for $i \geq 0$. An upper powerset is the appropriate abstract domain for underapproximating a concrete function's image: For $f : C_{\tau_1} \to \mathcal{P}(C_{\tau_2})$ and $f^\flat : A_{\tau_1} \to \mathcal{P}_U(A_{\tau_2})$ such that $f\,\rho_{\tau_1 \to U(\tau_2)}\, f^\flat$, we know that $f(c)\,\rho_{U(\tau_2)}\, f^\flat(a)$, for $c\,\rho_{\tau_1}\, a$, meaning that every $a \in f^\flat(a)$ has a witness $c \in f(c)$.

We have this soundness result for approximating function preimages:[10]

Lemma 8.2 *Assume there exist two sets, $T_\phi \subseteq A_{\tau_2}$ and $S_\phi \subseteq C_{\tau_2}$, such that for all $a \in A_\tau$, $c \in C_\tau$, if $a \in T_\phi$ and $c\,\rho_{\tau_2}\, a$, then $c \in S_\phi$.*

Then, for $f : C_{\tau_1} \to \mathcal{P}_U(C_{\tau_2})$ and $f^\flat : A_{\tau_1} \to \mathcal{P}_U(A_{\tau_2})$ such that $f\,\rho_{\tau_1 \to U(\tau_2)}\, f^\flat$, for all $a \in A_{\tau_1}$, $c \in C_{\tau_1}$,

$$c\,\rho_{\tau_1}\, a \text{ and } a \in \mathit{pre}_{f^\flat}(T) \text{ imply } c \in \mathit{pre}_f(S).$$

Using the relationship, $\mathit{post}_f = \mathit{pre}_{f^{-1}}$, we apply Lemma 8.2 to fill the gap in Figure 6: Recall from Figure 5 that

$$\begin{aligned} c \models_{\tau_2} f(\phi), \quad & \text{if there exists } c' \in C_{\tau_1} \text{ such that } c \in f(c') \text{ and } c' \models_{\tau_1} \phi \\ & \text{iff } c \in \mathit{post}_f\{c' \mid c' \models_{\tau_1} \phi\} \\ & \text{iff } c \in \mathit{pre}_{f^{-1}}\{c' \mid c' \models_{\tau_1} \phi\} \end{aligned}$$

---

[9] Indeed, this representation is the usual one for nondeterministic state-transition relations.

[10] In Abramsky's terminology [1], $\mathit{pre}_{f^\flat}$ defines a *liveness relation*.

Now, add this abstract judgement to Figure 6 (assuming $f^{-1}\,\rho_{\tau_2 \to U(\tau_1)}\,f^\flat$):

$$\begin{aligned} a \models^{\mathcal{A}}_{\tau_2} f(\phi), &\ \text{ if } a \in pre_{f^\flat}\{a' \mid a' \models^{\mathcal{A}}_{\tau_1} \phi\} \\ &\ \text{ iff } \text{ there exists } a' \in A_{\tau_1} \text{ such that } a' \in f^\flat(a) \text{ and } a' \models^{\mathcal{A}}_{\tau_1} \phi \end{aligned}$$

By Lemma 8.2, Theorem 8.1 is preserved.[11]

We finish with some known results regarding expressibility and completeness for external logics. First, we write $[\![\phi]\!]_\tau$ to denote $\{c \mid c \models_\tau \phi\}$ (similarly for $[\![\phi]\!]^{\mathcal{A}}_\tau$). We can relate the sets, $[\![\phi]\!]_\tau$ and $[\![\phi]\!]^{\mathcal{A}}_\tau$, by means of the Galois connection, $(\mathcal{P}(C_\tau), \supseteq)\langle\overline{\alpha_u}, \overline{\gamma}\rangle(\mathcal{P}_\downarrow(A_\tau), \supseteq)$ [34], where $\overline{\gamma}(T) = \cup_{a \in T}\gamma(a)$ and $\overline{\alpha_u}(S) = \{a \mid \gamma(a) \subseteq S\}$, where $\gamma(a) = \{c \mid c\,\rho_\tau\,a\}$. We have that

- $\models^{\mathcal{A}}_\tau$ is best-preserving for $\models_\tau$ iff $\overline{\alpha_u}[\![\phi]\!]_\tau = [\![\phi]\!]^{\mathcal{A}}_\tau$ [34]
- $\models^{\mathcal{A}}_\tau$ is strongly-preserving for $\models_\tau$ iff $[\![\phi]\!]_\tau = \overline{\gamma}[\![\phi]\!]^{\mathcal{A}}_\tau$ [30]

The abstract external logic, $\models^{\mathcal{A}}_\tau$, achieves completeness for $\models_\tau$ when each of its logical operators possess completeness: First, rewrite each concrete judgement form in the format,

$$c \models_\tau op_f(\phi_i)_{i<k}, \text{ if } c \in f([\![\phi_i]\!]_{\tau_i})_{i<k},$$

for k-ary logical operator, $f : \mathcal{P}(C_{\tau_i})^k \to \mathcal{P}(C_\tau)$ (similarly for $\models^{\mathcal{A}}_\tau$). When the logical relations, ρ_τ, define Galois connections, we have these results:

- The abstract judgement set, $\models^{\mathcal{A}}_\tau$, that proves the most sound properties for concrete judgement set, $\models_\tau$, is the one that approximates each concrete logical operator, $f : \mathcal{P}(C_{\tau_i})^k \to \mathcal{P}(C_\tau)$, by $f^\sharp_{best} : \mathcal{P}_\downarrow(A_{\tau_i})^k \to \mathcal{P}_\downarrow(A_\tau)$ [6,13,36]
- $\models^{\mathcal{A}}_\tau$ is best-preserving for $\models_\tau$ if each abstract logical operator, $f^\sharp$, is α_{ρ_τ}-complete for each concrete logical operator, f [11].
- $\models^{\mathcal{A}}_\tau$ is strongly-preserving for $\models_\tau$ if each $f^\sharp$ is γ_{ρ_τ}-complete for f [29].

9 Conclusion

This paper showed how to extract an appropriate programming logic from a logical-relation family that also defines a static analysis. Figure 6 displays the logic that results from a classical family of logical relations. As noted in the Introduction, a variety of logics stem from the setup in Figure 6: First, it is common to limit the set-conjunction and set-disjunction connectives to one argument each, giving this logic:

$$\begin{aligned} & a \models^{\mathcal{A}}_b a', \text{ if } a \sqsubseteq_b a' \\ & T \models^{\mathcal{A}}_{L(\tau)} \forall\phi, \text{ if for all } a \in T, a \models^{\mathcal{A}}_\tau \phi \\ & T \models^{\mathcal{A}}_{U(\tau)} \exists\phi, \text{ if there exists } a \in T \text{ such that } a \models^{\mathcal{A}}_\tau \phi \\ & a \models^{\mathcal{A}}_{\tau_1} f;\phi, \text{ if } f^\sharp(a) \models^{\mathcal{A}}_{\tau_2} \phi, \text{ for } f^\sharp \in A_{\tau_1} \to A_{\tau_2} \end{aligned}$$

---

[11] Here is an obvious question: Why not approximate $f : C_{\tau_1} \to \mathcal{P}(C_{\tau_2})$ by some $f^\flat : A_{\tau_1} \to \mathcal{P}_U(A_{\tau_2})$ and approximate $post_f$ by $post_{f^\flat}$? As shown in [35], $post_{f^\flat}$ is antimonotone and unsound for underapproximating function postimage.

If we hide the typings attached to the judgements, which is usually done, then we restrict the logic to judgements on base type — we do so by applying the operator for function preimage to the ones for disjunction and conjunction:

$a \models^{\mathcal{A}} f;\forall\phi,$ if for all $a' \in f^{\sharp}(a),\ a' \models^{\mathcal{A}} \phi,$ for $f^{\sharp} \in A_\tau \to \mathcal{P}_L(A_\tau)$

$a \models^{\mathcal{A}} f;\exists\phi,$ if there exists $a' \in f^{\flat}(a)$ such that $a' \models^{\mathcal{A}} \phi,$ for $f^{\flat} \in A_\tau \to \mathcal{P}_U(A_\tau)$

We can abbreviate $d \models_\tau f;\forall\phi$ by $d \models_\tau \forall f.\phi$ (as in *description logic* [3]), or by $[f]\phi$ (*Hennessy-Milner logic* [18]), or by $\Box\phi$ when the system studied has only one transition function (*CTL* [5]). Similarly, $d \models_\tau f;\exists\phi$ is abbreviated by $d \models_\tau \exists f.\phi$, or by $\langle f\rangle\phi$, or merely by $\Diamond\phi$.

10 History and related work

Galois connections were first proposed by Patrick and Radhia Cousot as a formalization of program data-flow and static analysis [7]; the Cousots also defined the notion of best approximation of a transfer function [8]. The notion of a functionally complete approximate transfer function was proposed by Giacobazzi, et al. [14,15].

The lifting of Galois connections from base type to higher types was studied by Nielson [25] and the Cousots [10]. The characterization of a Galois connection by an approximation relation came from Shmuely [38] and Hartmanis and Stearns [17]. Mycroft and Jones connected the approximation relation to the soundness of static analysis [24], and the idea was formalized by Schmidt [32,33].

Abramsky formalized the connection between approximation relations and logical relations within category theory, and his paper [1] provided a categorical formulation where Kan extensions are used to characterize the notion of best approximating transition function. Backhouse and Backhouse adapted Abramsky's ideas to relational algebra [4].

Abramsky also defined Scott-domain theory in "logical form" [2], where domains are generated from a set of primitive propositions such that each domain element is a collection (conjunction) of the propositions that hold true for it. Jensen adapted this formulation to define "abstract interpretation in logical form" [19], where an abstract interpretation is defined as collecting some fixed subset of the primitive propositions used to generate the concrete-domain elements. This provides a simple characterization of completeness as the collection of *all* the propositions contained in a concrete-element's denotation.

Abramsky's and Jensen's efforts are the first towards extracting program logics from semantic domains, but in general, the connection between abstract-interpretation domains and logics for program validation is ill-developed (hence, this paper). The traditional logic used with an abstract-interpretation domain is a conjunction of primitive propositions (Jensen's "conjunctive logic" [19]), called in this paper the domain's internal logic.

Steffen was the first to observe a connection between branching-time temporal logic and the format of standard data-flow analysis problems [40] — a connection used by Schmidt in his slogan: "data-flow analysis is model checking of abstract interpretations" [31,37]. Lacey, et al. built on this idea to define both the static analysis *and* the program transformation triggered by its results in terms of a temporal logic enriched by Prolog-style logical variables [21], reinforcing the intuition that there exists a fundamental connection between temporal logic and abstract-interpretation domains.

One of the most striking pieces of evidence for this connection was produced by Dams, who showed how software "abstract model checking" could be formalized by means of sound abstract interpretations using domains of overapproximating ("may") and underapproximating ("must") denotations [12,13]. Schmidt formalized Dams's constructions within a theory of Galois connections generated from logical-relation-based, lower- and upper-powerset abstract domains [33,35,36].

The present paper combines these threads of work.

Acknowledgements In 1982 at Edinburgh University, I learned about powerdomains and logical relations from Gordon Plotkin's lectures and notes. The present work stems from that education and has benefitted from interactions with Radhia and Patrick Cousot, Roberto Giacobazzi, Michael Huth, Isabella Mastroeni, Francesco Ranzato, and Francesco Tapparo.

References

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

[2] S. Abramsky. Domain theory in logical form. *Ann.Pure Appl.Logic*, 51:1–77, 1991.

[3] F. Baader, et al. *The Description Logic Handbook*. Cambridge Univ. Press, 2003.

[4] K. Backhouse and R. Backhouse. Galois connections and logical relations. In *Mathematics of Program Construction*, LNCS 2386. Springer Verlag, 2002.

[5] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 2000.

[6] R. Cleaveland, P. Iyer, and D. Yankelevich. Optimality in abstractions of model checking. In *Proc. SAS'95*. Springer LNCS 983, 1995.

[7] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs. In *Proc. 4th ACM Symp. on Principles of Programming Languages*, pages 238–252. ACM Press, 1977.

[8] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. 6th ACM Symp. on Principles of Programming Languages*, pages 269–282. ACM Press, 1979.

[9] P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Computation*, 2:511–547, 1992.

[10] P. Cousot and R. Cousot. Higher-order abstract interpretation. In *Proceedings IEEE Int. Conf. Computer Lang.*, 1994.

[11] P. Cousot and R. Cousot. Temporal abstract interpretation. In *Proc. 27th ACM Symp. on Principles of Programming Languages*, pages 12–25. ACM Press, 2000.

[12] D. Dams. *Abstract interpretation and partition refinement for model checking*. PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 1996.

[13] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Prog. Lang. Systems*, 19:253–291, 1997.

[14] R. Giacobazzi and E. Quintarelli. Incompleteness, counterexamples, and refinements in abstract model checking. In *Static Analysis Symposium*, LNCS 2126, pages 356–373. Springer Verlag, 2001.

[15] R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. *J. ACM*, 47:361–416, 2000.

[16] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

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

[18] M.C.B. Hennessy and Robin Milner. Algebraic laws for non-determinism and concurrency. *JACM*, 32:137–161, 1985.

[19] T. Jensen. *Abstract Interpretation in Logical Form*. PhD thesis, Imperial College, London, 1992.

[20] N. Jones and F. Nielson. Abstract interpretation: a semantics-based tool for program analysis. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 4*, pages 527–636. Oxford Univ. Press, 1995.

[21] D. Lacey, N.D. Jones, E. VanWyk, and C.C. Frederiksen. Compiler optimization correctness by temporal logic. *J. Higher Order and Symbolic Comp.*, 17:173–206, 2004.

[22] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for verification of concurrent systems. *Formal Methods in System Design*, 6:1–36, 1995.

[23] A. Miné. The octagon abstract domain. *J. Higher-Order and Symbolic Computation*, 19:31–100, 2006.

[24] A. Mycroft and N.D. Jones. A relational framework for abstract interpretation. In *Programs as Data Objects*, LNCS 217, pages 156–171. Springer Verlag, 1985.

[25] F. Nielson. Two-level semantics and abstract interpretation. *Theoretical Computer Science*, 69(2):117–242, 1989.

[26] F. Nielson, H.R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer Verlag, 1999.

[27] G. Plotkin. Domains. Lecture notes, Univ. Pisa/Edinburgh, 1983.

[28] G. D. Plotkin. Lambda-definability in the full type hierarchy. In J. Seldin and J. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–374. Academic Press, 1980.

[29] F. Ranzato and F. Tapparo. Strong preservation as completeness in abstract interpretation. In *Proc. European Symp. Programming*, LNCS 2986, pages 18–32. Springer Verlag, 2004.

[30] F. Ranzato and F. Tapparo. Strong preservation of temporal fixpoint-based operators by abstract interpretation. In *Proc. Conf. VMCAI'06*, LNCS 3855, pages 332–347. Springer Verlag, 2006.

[31] D.A. Schmidt. Data-flow analysis is model checking of abstract interpretations. In *Proc. 25th ACM Symp. on Principles of Prog. Languages*. ACM Press, 1998.

[32] D.A. Schmidt. Structure-preserving binary relations for program abstraction. In *The Essence of Computation*, LNCS 2566, pages 246–266. Springer Verlag, 2002.

[33] D.A. Schmidt. Closed and logical relations for over- and under-approximation of powersets. In *Symp. Static Analysis (SAS'04)*, LNCS 3148, pages 22–37. Springer Verlag, 2004.

[34] D.A. Schmidt. Comparing completeness properties of static analyses and their logics. In *Asian Symp. Prog. Lang. Systems (APLAS'06)*, LNCS 4279, pages 183–199. Springer Verlag, 2006.

[35] D.A. Schmidt. Underapproximating predicate transformers. In *Proc. Symp. Static Analysis (SAS'06)*, LNCS 4134, pages 127–143. Springer Verlag, 2006.

[36] D.A. Schmidt. A calculus of logical relations for over- and underapproximating static analyses. *Science Comp. Programming*, 64:29–53, 2007.

[37] D.A. Schmidt and B. Steffen. Data-flow analysis as model checking of abstract interpretations. In G. Levi, editor, *Proc. 5th Static Analysis Symposium*. Springer LNCS 1503, 1998.

[38] Z. Shmuely. The structure of Galois connections. *Pacific J. Mathematics*, 54:209–225, 1974.

[39] M. Smyth. Powerdomains. *Journal of Computer and System Sciences*, 16:23–36, 1978.

[40] B. Steffen. Generating data-flow analysis algorithms for modal specifications. *Science of Computer Programming*, 21:115–139, 1993.

[41] A. Venet. Automatic determination of communication topologies in mobile systems. In *Symp. Static Analysis (SAS'98)*, pages 152–167. Springer Verlag, 1998.

[42] G. Winskel. On powerdomains and modality. *Theor. Comput. Sci.*, 36:127–137, 1985.