Inverse-Limit and Topological Aspects of Abstract Interpretation

David A. Schmidt[a,1]

[a] *Computing and Information Sciences Dept., Kansas State University, Manhattan, KS 66506, USA*

Abstract

We develop abstract-interpretation domain construction in terms of the inverse-limit construction of denotational semantics and topological principles: We define an abstract domain as a "structural approximation" of a concrete domain if the former exists as a finite approximant in the inverse-limit construction of the latter, and we extract the appropriate Galois connection for sound and complete abstract interpretations. The elements of the abstract domain denote (basic) open sets from the concrete domain's Scott topology, and we hypothesize that every abstract domain, even non-structural approximations, defines a weakened form of topology on its corresponding concrete domain.

We implement this observation by relaxing the definitions of topological open set and continuity; key results still hold. We show that families of closed and open sets defined by abstract domains generate post- and pre-condition analyses, respectively, and Giacobazzi's forwards- and backwards-complete functions of abstract-interpretation theory are the topologically closed and continuous maps, respectively. Finally, we show that Smyth's upper and lower topologies for powerdomains induce the overapproximating and underapproximating transition functions used for abstract-model checking.

Key words: Abstract interpretation, denotational semantics, inverse-limit construction, Galois connection, Scott-topology

Email address: `das@ksu.edu` (David A. Schmidt)

[1]Supported by NSF ITR-0326577.

```
readInt(x)
x := negate(x)
while x < 0 :
  x := succ(x)
writeInt(x)
```

Transfer functions: For $i \in \mathit{Int}$,

$$\mathit{negate}(i) = -i$$

$$\mathit{filter}_{<0}(i) = \begin{cases} i & \text{if } i < 0 \\ \underline{\perp} & \text{if } i \geq 0 \end{cases}$$

$$\mathit{filter}_{\geq 0}(i) = \begin{cases} i & \text{if } i \geq 0 \\ \underline{\perp} & \text{if } i < 0 \end{cases}$$

$$\mathit{succ}(i) = i + 1$$

For functions $p, q \in \mathit{Int} \rightharpoonup \mathit{Int}$, their composition is *strict*, that is, if $p(k) = \underline{\perp}$, then $(q \circ p)(k) = \underline{\perp}$. The meaning of the program is

$$P = p_1 \circ p_0 \ : \ \mathit{Int} \rightharpoonup \mathit{Int}$$

where

$$\begin{aligned} p_0 &= \mathit{negate} \\ p_1(i) &= p_{11}(i) \sqcup (p_1 \circ p_2 \circ p_{12})(i) \\ p_{11} &= \mathit{filter}_{\geq 0} \\ p_{12} &= \mathit{filter}_{<0} \\ p_2 &= \mathit{succ} \end{aligned}$$

Figure 1: Example program, its transfer functions of arity $\mathit{Int} \rightharpoonup \mathit{Int}$, and its functional semantics

1. Introduction

Abstract intepretation performs finite computation of program properties [1, 2, 3]. As indicated by Cousot and Cousot [1, 4], for state set Σ and program $P : \Sigma \rightharpoonup \Sigma$,[2] the "program properties" are subsets of Σ. For example, for input property $S_0 \subseteq \Sigma$, P's postcondition property is $P[S_0] \subseteq \Sigma$ (where $P[S_0] = \{P(s) \in \Sigma \mid s \in S_0\}$). In general, it is impossible to calculate finitely $P[S_0]$, because S_0 might be infinite, or there might exist some $\sigma_0 \in S_0$ such that $P(\sigma_0)$ diverges. For this reason, abstract interpretation computes finitely an approximate answer, S', such that $P[S_0] \subseteq S'$.

Here is a motivating example. Figure 1 shows a small program in its upper left column, whose state consists of a single integer, named `x`. The meaning of the program is assembled from partial *transfer functions* of arity $\mathit{Int} \rightharpoonup \mathit{Int}$. The transfer functions are listed in the upper right column. The program's denotation is a composition of the transfer functions. (When $p(i)$

[2] Here, $\Sigma \rightharpoonup \Sigma$ denotes the partial functions from Σ to Σ.

is undefined, we write $p(i) = \underline{\perp}$ in the Figure.)

In particular, the while-loop is denoted by a recursively defined function, p_1, whose meaning is its least-fixed point within the cpo, $(\mathit{Int} \rightharpoonup \mathit{Int}) \rightarrow (\mathit{Int} \rightharpoonup \mathit{Int})$. The functions, $\mathit{filter}_{<0}$ and $\mathit{filter}_{\geq 0}$, guard the loop's body and exit, respectively, and the results are joined via $\sqcup$. Since the two filters are disjoint on the integer values they filter, $\sqcup$ is well defined.

For $P = p_1 \circ p_0$, we can prove that $P(2) = 0$, $P(3) = 0$, and indeed, $P(i) = 0$ for all $i \geq 0$. Likewise, we can prove $P(j) = -j$, for all $j < 0$. These properties are postcondition properties, and they are defined by "lifting" each $p : \mathit{Int} \rightharpoonup \mathit{Int}$ in Figure 1 to a total function, $p : \mathcal{P}(\mathit{Int}) \rightarrow \mathcal{P}(\mathit{Int})$, $p[S] = \{p(i) \in \mathit{Int} \mid i \in S\}$. This is the *forwards collecting interpretation* [1] of p, and it is a partial-correctness interpretation, ignoring instances of $p(i) = \underline{\perp}$. The meaning of $P = p_1 \circ p_0 : \mathcal{P}(\mathit{Int}) \rightarrow \mathcal{P}(\mathit{Int})$ lifts accordingly, where the recursion is solved within the domain, $(\mathcal{P}(\mathit{Int}) \rightarrow \mathcal{P}(\mathit{Int})) \rightarrow (\mathcal{P}(\mathit{Int}) \rightarrow \mathcal{P}(\mathit{Int}))$, and $\sqcup$ is computed on $\mathcal{P}(\mathit{Int})$ as set union.

Now, we can prove that $P\{i \mid i \geq 0\} = \{0\}$ and $P\{j \mid j < 0\} = \{j \mid j > 0\}$, which are the strongest postconditions of the two input properties.

The forwards collecting semantics of P is well defined, but it is not finitely computable, and a key insight of abstract-interpretation theory is to limit to a finite number the calls to the p_i functions when computing $P[S]$. To accomplish this, we limit to a finite number the sets that are allowed as arguments and answers to the p_is. For state set, Σ, let the *abstract domain*, $A \subseteq \mathcal{P}(\Sigma)$, be a finite subcollection of $\mathcal{P}(\Sigma)$ such that $\{\} \in A$, and for all sets, $a_1, a_2 \in A$, there exists $a_3 \in A$ such that $a_1 \cup a_2 \subseteq a_3$, that is, A is a finite, bounded cpo.

When A is defined so that, for every $S \in \mathcal{P}(\Sigma)$, there exists a least $a \in A$ such that $S \subseteq a$, then there is a *Galois connection* between A and $\mathcal{P}(\Sigma)$, which we develop in the next section.

We compute upon the elements of abstract domain A. A function, $p : \mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)$, is *overapproximated* by $p^\sharp : A \rightarrow A$ when $p^\sharp(a) \supseteq p[a]$ for all $a \in A$. Since $A = \{a_0, a_1, \cdots, a_m\}$ has finite cardinality $m > 0$, each function $p^\sharp(x) = e_x$ is expanded into its m first-order equational instances, $\{p^\sharp(a') = e_{a'} \mid a' \in A\}$, and the equations are solved simultaneously.

For the example in Figure 1, perhaps we choose the finite collection, $\mathit{Sign}_0 = \{\mathit{none}, \mathit{neg}, \mathit{zero}, \leq 0, \mathit{pos}, \mathit{any}\}$, where the names denote, respectively, the sets $\{\}, \{i \in \mathit{Int} \mid i < 0\}, \{0\}, \{i \in \mathit{Int} \mid i \leq 0\}, \{i \in \mathit{Int} \mid i > 0\}$, and Int.

Figure 2 shows the program's *abstract interpretation* upon Sign_0, which

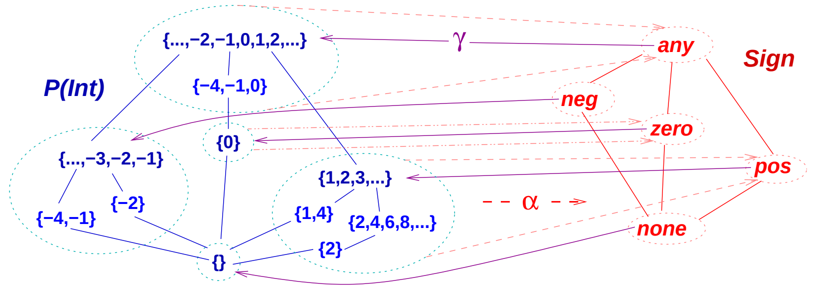
$$\begin{aligned}
&p^\sharp(\mathit{pos}) = p_1^\sharp(p_0^\sharp(\mathit{pos})) = p_1^\sharp(\mathit{neg}) = \cdots(\text{see below})\cdots = \mathit{zero}\\
&\quad p_0^\sharp(\mathit{pos}) = \mathit{negate}^\sharp(\mathit{pos}) = \mathit{neg}\\
&\quad p_1^\sharp(\mathit{neg}) = p_{11}^\sharp(\mathit{neg}) \sqcup (p_1^\sharp \circ p_2^\sharp \circ p_{12}^\sharp)(\mathit{neg}) = \mathit{none} \sqcup p_1^\sharp(\leq 0)\\
&\qquad\quad = \mathit{none} \sqcup \mathit{zero} = \mathit{zero}\\
&\qquad p_{11}^\sharp(\mathit{neg}) = \mathit{filter}_{\geq 0}^\sharp(\mathit{neg}) = \mathit{none}\\
&\qquad p_{12}^\sharp(\mathit{neg}) = \mathit{filter}_{<0}^\sharp(\mathit{neg}) = \mathit{neg}\\
&\qquad p_2^\sharp(\mathit{neg}) = \mathit{succ}^\sharp(\mathit{neg}) = \leq 0\\
&\quad p_1^\sharp(\leq 0) = p_{11}^\sharp(\leq 0) \sqcup (p_1^\sharp \circ p_2^\sharp \circ p_{12}^\sharp)(\leq 0) = \mathit{zero} \sqcup p_1^\sharp(\leq 0)\\
&\qquad\quad = \cdots(\text{least fixed point})\cdots = \mathit{zero}\\
&\qquad p_{11}^\sharp(\leq 0) = \mathit{filter}_{\geq 0}^\sharp(\leq 0) = \mathit{zero}\\
&\qquad p_{12}^\sharp(\leq 0) = \mathit{filter}_{<0}^\sharp(\leq 0) = \mathit{neg}
\end{aligned}$$

Figure 2: Abstract intepretation, $p^\sharp(\mathit{pos})$, of program p using abstract domain, $\mathit{Sign}_0 = \{\mathit{none}, \mathit{neg}, \mathit{zero}, \leq 0, \mathit{pos}, \mathit{any}\}$

calculates for precondition, pos, that postcondition $p^\sharp(\mathit{pos})$ is zero. How did we know in advance to choose Sign_0 to calculate this postcondition? In practice, either one chooses in advance the abstract domain, A, based on "structural" considerations of Σ [1, 2, 5], or one dynamically generates A on the fly, based on "relational" considerations [6, 7].

In this paper, we draw from precedents from denotational semantics and topology to understand better the choice of abstract domain, $A \subseteq \mathcal{P}(\Sigma)$:

1. "Structural" approximations of Σ are extracted from the inverse-limit construction of the Scott domain, $\Sigma = D^\infty$. Within the inverse-limit chain, each D_k serves as a structural approximation of its limit, D^∞, in the sense that the elements of D_k name subsets of D^∞. Indeed, these named subsets are open sets from D^∞'s Scott topology, consistent with Smyth's hypothesis that open sets are "semicomputable properties" [8], in this case, for program analysis.
2. When an abstract domain is defined on-the-fly, usually "relationally," based on the relation between values in Σ and the program analyzed (e.g., intervals [1], polyhedra [6], and predicate abstractions [7]), we relate the abstract domain's elements to Σ *as if the former define a topology on the latter*. The resulting, "weak topology" (not always closed under union, not always closed under binary intersection) preserves basic topological concepts, and we prove that the notions of forwards-complete and backwards-complete functions, introduced by



Define $Sign = \{none, neg, zero, pos, any\}$ and $\gamma : Sign \rightarrow \mathcal{P}(Int)$ as
$\gamma(none) = \emptyset$; $\gamma(neg) = \{\cdots, -2, -1\}$; $\gamma(zero) = \{0\}$;
$\gamma(pos) = \{1, 2, \cdots\}$; $\gamma(any) = Int$

Define $\alpha : \mathcal{P}(Int) \rightarrow Sign$ as $\alpha(S) = \sqcap\{a \in Sign \mid S \subseteq \gamma(a)\}$. For example, $\alpha\{-4,-1\} = neg$, $\alpha\{-4,-1,0\} = any$, $\alpha\{0\} = zero$, etc.

Figure 3: Abstract domain, *Sign*, concretization map, γ, and its adjoint, α

Giacobazzi, et al. [9, 10] to formalize most-precise abstract interpretations, are characterized as the topologically closed and topologically continuous maps on the weak topology.

2. Background: Abstract interpretation

We first review the classical notions used in abstract interpretation.

For concrete-data domain, Σ, we select a set of property names, A, that denote subsets of Σ: Each $a \in A$ names the set $\gamma(a) \subseteq \Sigma$, for $\gamma : A \rightarrow \mathcal{P}(\Sigma)$. We order abstract domain A so that $a \sqsubseteq a'$ iff $\gamma(a) \subseteq \gamma(a')$ — to be useful, A should be a bounded cpo, and better still, it should have binary joins. Figure 3 uses the property names, *neg*, *zero*, and *pos*, to partition the integers, *Int*, within a complete lattice named *Sign*.

When γ possesses an adjoint, $\alpha : \mathcal{P}(\Sigma) \rightarrow Sign$, then there is a *Galois connection* (that is, $S \subseteq \gamma(a)$ iff $\alpha(S) \sqsubseteq a$, for all $S \in \mathcal{P}(\Sigma)$ and $a \in A$). α is the *lower adjoint* and γ is the *upper adjoint*, and we write this as $\mathcal{P}(\Sigma)\langle\alpha,\gamma\rangle A$. This situation ensures that every $S \subseteq \Sigma$ can be closed into a least property, $\alpha(S)$, such that $S \subseteq \gamma(\alpha(S))$. We define $\rho = \gamma \circ \alpha$ to embed a set into its most-precise property set: $\rho : \mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)$ is an *upper closure operator*: it is monotone, extensive ($S \subseteq \rho(S)$), and idempotent ($\rho \circ \rho = \rho$). ρ's image is closed under intersection.

Let $f : \Sigma \rightharpoonup \Sigma$ be a partial function whose properties we wish to express within abstract domain, A. As before, we define its lift, $f : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$, as $f[S] = \{f(\sigma) \in \Sigma \mid \sigma \in S\}$. $f^\sharp : A \to A$ *soundly approximates* f if, for all $a \in A$, $f[\gamma(a)] \subseteq \gamma(f^\sharp(a))$. When γ has an adjoint, α, this is equivalent to $\alpha(f[S]) \sqsubseteq f^\sharp(\alpha(S))$, for all $S \in \mathcal{P}(\Sigma)$. It is always the case that $f_0^\sharp = \alpha \circ f \circ \gamma$ soundly approximates f. Indeed, $f_0^\sharp(a)$ calculates *strongest postconditions for f within in A*: for all $a, a' \in A$, if $f[\gamma(a)] \subseteq \gamma(a')$, then $f_0^\sharp(a) \sqsubseteq a'$. (That is, $f[\gamma(a)] \subseteq \gamma(f_0^\sharp(a)) \subseteq \gamma(a')$.)

Figure 4 displays some sample functions on Sign, their lifts, and sound approximating functions. All the approximating functions compute strongest postconditions on Sign.

When f is approximated exactly by $f^\sharp$ such that $f \circ \gamma = \gamma \circ f^\sharp$, we say $f^\sharp$ is *forwards complete for f* [10]. When f is approximated exactly such that $\alpha \circ f = f^\sharp \circ \alpha$, we say $f^\sharp$ is *backwards complete for f* [9, 11]. The two completeness notions are homomorphism properties, as illustrated in Figure 5.

It is easy to prove that when some $f^\sharp$ is forwards complete for f, then $f^\sharp = f_0^\sharp$ (similar for backwards complete). Since $f_0^\sharp$ is defined from f, we say that "f is forwards complete" when $f_0^\sharp$ is forwards complete for f (similar for "backwards complete"). For example, in Figure 3, $\mathit{filter}_{<0}$ is forwards but not backwards complete; negate is both backwards and forwards complete, and succ and $\mathit{filter}_{\geq 0}$ are neither.

Since $\rho[\mathcal{P}(\Sigma)] = \gamma[A]$ lists the properties named by A, we can understand $f^\sharp : A \to A$ as if it had arity, $\rho[\mathcal{P}(\Sigma)] \to \rho[\mathcal{P}(\Sigma)]$. In particular, $(\rho \circ f) : \rho[\mathcal{P}(\Sigma)] \to \rho[\mathcal{P}(\Sigma)]$ soundly approximates f in $\rho[\mathcal{P}(\Sigma)]$ (that is, for $\phi \in \rho[\mathcal{P}(\Sigma)]$, $f[\phi] \subseteq (\rho \circ f)[\phi]$), and it computes computes strongest postconditions for f (that is, for all $\phi, \psi \in \rho[\mathcal{P}(\Sigma)]$, if $f[\phi] \subseteq \psi$, then $(\rho \circ f)[\phi] \subseteq \psi$).

We define $f_0^\sharp = \rho \circ f : \mathcal{P}(\Sigma) \to \rho[\mathcal{P}(\Sigma)]$ in the propositions that follow:

Proposition 1. *[10] The following are equivalent:*

- *$f_0^\sharp$ is forwards complete for f*
- *for all $\phi \in \rho[\mathcal{P}(\Sigma)]$, $f[\phi] \in \rho[\mathcal{P}(\Sigma)]$*
- *$f \circ \rho = \rho \circ f \circ \rho$*

Proposition 2. *[2, 9] The following are equivalent:*

- *$f_0^\sharp$ is backwards complete for f*

$f : \mathit{Int} \rightharpoonup \mathit{Int}$	$f : \mathcal{P}(\mathit{Int}) \to \mathcal{P}(\mathit{Int})$	$f^\sharp : \mathit{Sign} \to \mathit{Sign}$
$\mathit{succ}(i) = i + 1$	$\mathit{succ}[S] = \{\mathit{succ}(i) \mid i \in S\}$ Not forwards complete (consider *neg*); not backwards complete (consider $\{-1\}$).	$\mathit{succ}^\sharp(\mathit{none}) = \mathit{none}$ $\mathit{succ}^\sharp(\mathit{zero}) = \mathit{pos}$ $\mathit{succ}^\sharp(\mathit{pos}) = \mathit{pos}$ $\mathit{succ}^\sharp(\mathit{neg}) = \mathit{any}$ (!) $\mathit{succ}^\sharp(\mathit{any}) = \mathit{any}$ (!)
$\mathit{negate}(i) = -i$	$\mathit{negate}[S] = \{\mathit{negate}(i) \mid i \in S\}$ Forwards complete; backwards complete	$\mathit{negate}^\sharp(\mathit{neg}) = \mathit{pos}$ $\mathit{negate}^\sharp(\mathit{zero}) = \mathit{zero}$ $\mathit{negate}^\sharp(\mathit{pos}) = \mathit{neg}$ $\mathit{negate}^\sharp(\mathit{any}) = \mathit{any}$ $\mathit{negate}^\sharp(\mathit{none}) = \mathit{none}$
$\mathit{filter}_{<0}(i) = \begin{cases} i \text{ if } i < 0 \\ \perp \text{ if } i \geq 0 \end{cases}$	$\mathit{filter}_{<0}[S] = \{\mathit{filter}_{<0}(i) \mid i \in S\}$ Forwards complete; not backwards complete (consider $\{0, 1\}$).	$\mathit{filter}^\sharp_{<0}(\mathit{neg}) = \mathit{neg}$ $\mathit{filter}^\sharp_{<0}(\mathit{any}) = \mathit{neg}$ $\mathit{filter}^\sharp_{<0}(\mathit{zero}) = \mathit{none}$ $\mathit{filter}^\sharp_{<0}(\mathit{pos}) = \mathit{none}$ $\mathit{filter}^\sharp_{<0}(\mathit{none}) = \mathit{none}$
$\mathit{filter}_{\geq 0}(i) = \begin{cases} i \text{ if } i \geq 0 \\ \perp \text{ if } i < 0 \end{cases}$	$\mathit{filter}_{\geq 0}[S] = \{\mathit{filter}_{\geq 0}(i) \mid i \in S\}$ Not forwards complete (consider *any*); not backwards complete (consider $\{-1, 1\}$)	$\mathit{filter}^\sharp_{\geq 0}(\mathit{neg}) = \mathit{none}$ $\mathit{filter}^\sharp_{\geq 0}(\mathit{none}) = \mathit{none}$ $\mathit{filter}^\sharp_{\geq 0}(\mathit{zero}) = \mathit{zero}$ $\mathit{filter}^\sharp_{\geq 0}(\mathit{pos}) = \mathit{pos}$ $\mathit{filter}^\sharp_{<0}(\mathit{any}) = \mathit{any}$ (!)

Figure 4: Transfer functions, their collecting interpretations, and their sound approximations on *Sign*

$f^\sharp : A \to A$ is *sound* for $f : \Sigma \to \Sigma$ iff $f \circ \gamma \sqsubseteq \gamma \circ f^\sharp$ iff $\alpha \circ f \sqsubseteq f^\sharp \circ \alpha$

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

α and γ act as *semi-homomorphisms.*

Forwards completeness [10]:
$f \circ \gamma = \gamma \circ f^\sharp$

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

γ is a homomorphism from A to $\mathcal{P}(\Sigma)$ — it preserves $f^\sharp$ as f.

Backwards completeness [2, 9]:
$\alpha \circ f = f^\sharp \circ \alpha$

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

α is a homomorphism from $\mathcal{P}(\Sigma)$ to A — it preserves f as $f^\sharp$.

Figure 5: Sound and complete forms of abstract functions

- *for all* $S_1, S_2 \in \mathcal{P}(\Sigma)$, $\rho(S_1) = \rho(S_2)$ *implies* $\rho(f[S_1]) = \rho(f[S_2])$
- $\rho \circ f = \rho \circ f \circ \rho$.

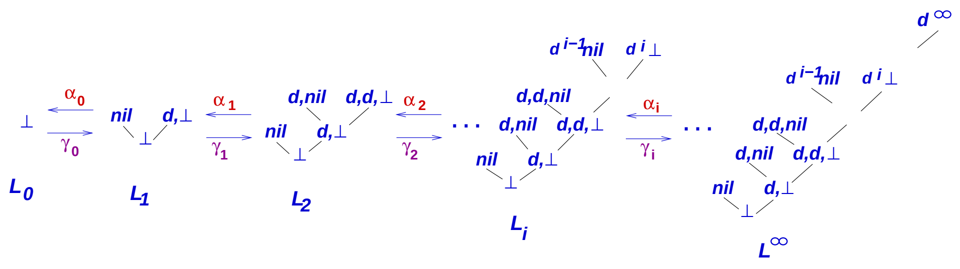
Both forwards- and backwards-complete functions calculate strongest post-conditions, even though the two notions are inequivalent [10]. Later, we will use topology to prove that a forwards-complete function preserves properties, whereas a backwards-complete function reflects them, cf. Figure 5.

3. Background: Denotational semantics

One might explain *denotational semantics* as the interpretation of a program's phrases as values from *Scott-domains*. We treat a Scott-domain as the inverse limit of a sequence of finite-cardinality bounded cpos, related by embedding-projection pairs (the "Sequence of Finite Posets" construction) [12, 13]. Figure 6 presents the Scott-domain of finite and infinite lists corresponding to the domain equation, $L = (\{nil\} + (D \times L))_\perp$.[3] For each

[3]As usual, $+$ represents disjoint union, $\times$ is product, and $\_\_\perp$ is lifting.

$L^\infty \approx (\{nil\} + (D \times L^\infty)_\perp$, where D is some fixed Scott-domain



For $L_0 = \{\perp\}$, $L_{i+1} = (\{nil\} + (D \times L_i)_\perp$,
the embedding, projection pairs, $L_i\langle\gamma_i, \alpha_i\rangle L_{i+1}$, are defined

$$\gamma_0(\perp) = \perp \qquad \gamma_{i+1} = F(\gamma_i) \qquad F(f)(\perp) = \perp$$
$$\alpha_0(\ell) = \perp \qquad \text{(that is, } \gamma_{i+1}(\ell) = \ell) \qquad \text{where} \quad F(f)(nil) = nil$$
$$\alpha_{i+1} = F(\alpha_i) \qquad F(f)(d, \ell) = (d, f(\ell))$$

For $i < j$, define $\gamma_{i,j} = \gamma_{j-1} \circ \cdots \circ \gamma_{i+1} \circ \gamma_i$
$\alpha_{j,i} = \alpha_i \circ \cdots \circ \alpha_{j-2} \circ \alpha_{j-1}$

The elements of L^∞ are tuples, $\langle \ell_i \rangle_{i \geq 0}$, such that each $\ell_i \in L_i$ and $\ell_i = \alpha_i(\ell_{i+1})$ for all $i \geq 0$.

For all $i \geq 0$, $L_i\langle\gamma_{i,\infty}, \alpha_{\infty,i}\rangle L^\infty$ are defined

$$\gamma_{i,\infty}(\ell) = \langle \alpha_{i-1,0}(\ell), \alpha_{i-1,1}(\ell), \cdots, \alpha_i(\ell), \ell, \gamma_i(\ell), \gamma_{i,i+2}(\ell), \gamma_{i,i+3}(\ell), \cdots \rangle$$
$$\alpha_{\infty,i}\langle \ell_0, \ell_1, \cdots, \ell_i, \cdots \rangle = \ell_i$$

Finally, $L^\infty\langle\gamma^\infty, \alpha^\infty\rangle(\{nil\} + (D \times L^\infty))_\perp$ forms an order-isomorphism, where

$$\gamma^\infty = \sqcup_{i \geq 0} \; F(\gamma_{i,\infty}) \circ \alpha_{\infty,i+1}$$
$$\alpha^\infty = \sqcup_{i \geq 0} \; \gamma_{i+1,\infty} \circ F(\alpha_{\infty,i})$$

Figure 6: Inverse limit of $L = (\{nil\} + (D \times L))_\perp$

$$\begin{array}{l}
\mathtt{d} \in \mathit{Data}(\text{atomic data}) \\
\mathtt{x} \in \mathit{Var}(\text{variable names}) \\
\mathtt{G} \in \mathit{Guard}(\text{boolean expressions}) \\
\mathtt{E} \in \mathit{Expression} ::= \mathtt{x} \mid \mathtt{tl\ E} \mid \mathtt{cons\ d\ E} \\
\mathtt{C} \in \mathit{Command} ::= \mathtt{x\ =\ E} \mid \mathtt{C_1; C_2} \mid \mathtt{if\ (G_i : C_i)}_{i \in I}\ \mathtt{fi} \mid \mathtt{while\ G\ do\ C}
\end{array}$$

Domain of stores: $\sigma \in \Sigma = \mathit{Var} \to L^\infty$

$$\begin{array}{l}
\mathcal{G} : \mathit{Guard} \to \Sigma \to \Sigma_{\underline{\perp}} \\
\quad \mathcal{G}[\![\mathtt{G}]\!]\sigma = \sigma \text{ when } \mathtt{G} \text{ holds true in } \sigma; \qquad \mathcal{G}[\![\mathtt{G}]\!]\sigma = \underline{\perp} \text{ otherwise}
\end{array}$$

$$\begin{array}{l}
\mathcal{E} : \mathit{Expression} \to \Sigma \to L^\infty \\
\quad \mathcal{E}[\![\mathtt{x}]\!]\sigma = \mathit{lookup}\ [\![\mathtt{x}]\!]\ \sigma \quad \text{where } \mathit{lookup}\ v\ \sigma = \sigma(v) \\
\quad \mathcal{E}[\![\mathtt{tl\ E}]\!]\sigma = \mathit{tail}\ (\mathcal{E}[\![\mathtt{E}]\!]\sigma) \quad \text{where} \quad \mathit{tail}(v) = \mathit{cases}\ \gamma^\infty(v)\ \mathit{of} \begin{cases} \perp : \alpha^\infty(\perp) \\ \mathit{nil} : \alpha^\infty(\perp) \\ (d, \ell) : \ell \end{cases} \\
\quad \mathcal{E}[\![\mathtt{cons\ d\ E}]\!]\sigma = \mathit{cons}\ \mathtt{d}\ (\mathcal{E}[\![\mathtt{E}]\!]\sigma) \quad \text{where} \quad \mathit{cons}\ d\ \ell = \alpha^\infty(d, \ell)
\end{array}$$

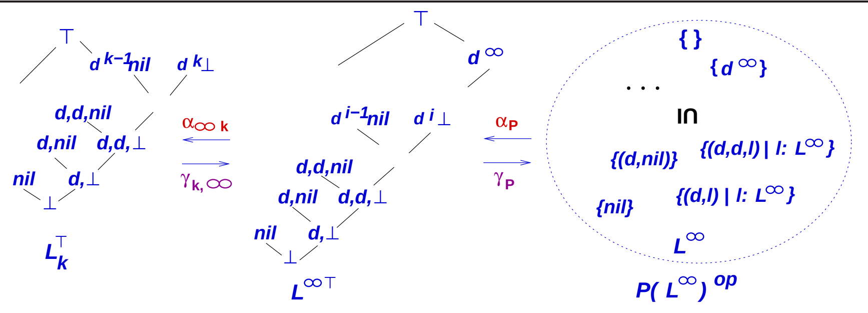
$$\begin{array}{l}
\mathcal{C} : \mathit{Command} \to \Sigma \to \Sigma_{\underline{\perp}} \\
\quad \mathcal{C}[\![\mathtt{x = E}]\!]\sigma = \mathit{update}\ [\![\mathtt{x}]\!]\ (\mathcal{E}[\![\mathtt{E}]\!]\sigma)\ \sigma \qquad \text{where } \mathit{update}\ v\ \ell\ \sigma = \sigma + [v \mapsto \ell] \\
\quad \mathcal{C}[\![\mathtt{C_1; C_2}]\!] = \mathcal{C}[\![\mathtt{C_2}]\!] \circ \mathcal{C}[\![\mathtt{C_1}]\!] \\
\qquad\qquad \text{Note} : \circ \text{ forces strictness: } g \circ f(\sigma) = \underline{\perp} \text{ when } f(\sigma) = \underline{\perp} \\
\quad \mathcal{C}[\![\mathtt{if\ (G}_i : \mathtt{C}_i)_{i \in I}\ \mathtt{fi}]\!] = \bigsqcup_{i \in I} \mathcal{C}[\![\mathtt{C}_i]\!] \circ \mathcal{G}[\![\mathtt{G}_i]\!] \\
\quad \mathcal{C}[\![\mathtt{while\ G\ do\ C}]\!] = \mathit{lfp}\ \lambda f.\ (\mathcal{G}[\![\neg\mathtt{G}]\!]) \sqcup (f \circ \mathcal{C}[\![\mathtt{C}]\!] \circ \mathcal{G}[\![\mathtt{G}]\!])
\end{array}$$

Figure 7: Denotational semantics for while-language based on L^∞

$i \geq 0$, the corresponding embedding-projection pair defines a Galois connection, $L_i\langle\gamma_i, \alpha_i\rangle L_{i+1}$, as does $L_i\langle\gamma_{i,\infty}, \alpha_{\infty,i}\rangle L^\infty$. (Here, the γ functions are *lower* adjoints.)

Figure 7 shows a denotational semantics for a while-language based on L^∞. A store is a mapping from a set of variables, *Var*, to values in L^∞. Absence of store is denoted by $\underline{\perp}$ (to distinguish it from $\perp \in L^\infty$). The language uses a guarded-if construction, where a guard, $\mathtt{G}_j$, filters the input store to its guarded command, $\mathtt{C}_j$, and the results of all $\mathtt{G_j} : \mathtt{C_j}$ pairs are joined. When the guards of an if-command are mutually exclusive, the semantics is the usual one. (We use this formulation to ease the transition into abstract interpretation, which treats software somewhat like flowcharts or circuits, cf. Figure 1).

The while-command is a tail-recursive guarded-if, such that `while B do C`



Define $L_k^\top \langle\gamma,\alpha\rangle \mathcal{P}(L^\infty)^{op}$ as $\begin{array}{l}\gamma = \gamma_P \circ \gamma_{k,\infty} \\ \alpha = \alpha_{\infty,k} \circ \alpha_P\end{array}$,

where $\begin{array}{l}\gamma_P(\ell) = \uparrow\ell \ = \{m \in L^\infty \mid \ell \sqsubseteq m\} \\ \alpha_P(S) = \sqcap S\end{array}$. We can rotate the above diagram and define the Galois connection, $\mathcal{P}(L^\infty)\langle\alpha,\gamma\rangle {L_k^\top}^{op}$

Figure 8: Collecting domain (data-test sets), $\mathcal{P}(L^\infty)^{op}$, for L^∞ and the associated Galois connections

has a denotation equal to $\texttt{if}\ (\neg\texttt{B} : \texttt{skip}),\ (\texttt{B} : (\texttt{C};\ \texttt{while B do C}))\ \texttt{fi}$.

Here is an example: let $\sigma_0 = [[\![\texttt{x}]\!] \mapsto nil]$. Then,

$$\begin{array}{l} \mathcal{C}[\![\texttt{if (isNil x : x = cons d0 x) (isNonNil x : x = x) fi}]\!]\sigma_0 \\ = \ (\mathcal{C}[\![\texttt{x = cons d0 x}]\!] \circ \mathcal{G}[\![\texttt{isNil x}]\!])\sigma_0 \ \sqcup \ (\mathcal{C}[\![\texttt{x = x}]\!] \circ \mathcal{G}[\![\texttt{isNonNil x}]\!])\sigma_0 \\ = \ \mathcal{C}[\![\texttt{x = cons d0 x}]\!]\sigma_0 \ \sqcup \ \mathcal{C}[\![\texttt{x = x}]\!]\underline{\bot} \\ = \ (update\ [\![\texttt{x}]\!]\ (\mathcal{E}[\![\texttt{cons d0 x}]\!]\sigma_0)\ \sigma_0) \ \sqcup \ \underline{\bot} \ = \ [[\![\texttt{x}]\!] \mapsto (\texttt{d0}, nil)] \end{array}$$

The example shows how $\mathcal{G}[\![\texttt{isNil x}]\!]$ passes σ_0 forwards because the guard holds true for the store, whereas $\mathcal{G}[\![\texttt{isNonNil x}]\!]$ passes $\underline{\bot}$.

4. Collecting domains

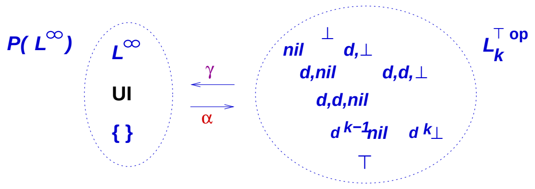
Reconsidering the L_k domains in Figure 6, we note that an element like $(d,\bot)$ denotes a list that has d as its head element and an unknown tail, that is, $(d,\bot)$ approximates the set, $\{(d,\ell) \mid \ell \in L^\infty\} \subseteq L^\infty$. In this sense, the elements of L_k name properties of L^∞, and L_k is a *structural approximating domain* of L^∞, like the ones used for abstract interpretation (cf. *Sign* in Figure 3).

We formalize this with a Galois connection. First, define the *collecting domain*, $\mathcal{P}(L^\infty)$, ordered by $\supseteq$. (We *ignore* the ordering on L^∞ [14].) Next, if we "crown" L^∞ with a $\top$ element, we have a Galois connection between the collecting domain and complete lattice, $L^{\infty\top}$; see Figure 8. Element

$\top \in L^{\infty\top}$ denotes contradictory (literally, no) information content and maps to the empty ("false") property in $\mathcal{P}(L^\infty)^{op}$. In contrast, $\bot \in L^{\infty\top}$ denotes all values in L^∞ ("true"). One might also restrict the collecting domain to be just the totally defined lists or just the finite, total lists.

The Figure shows how the Galois connection composes with the embedding-projection pair, $L_k^\top\langle\gamma_{k,\infty}, \alpha_{\infty,k}\rangle L^{\infty\top}$, where L_k is also crowned. The Galois connection that results, $L_k^\top\langle\gamma, \alpha\rangle\mathcal{P}(L^\infty)^{op}$, is significant: If we "rotate" it, we have a Galois connection suitable for abstract interpretation:

$\mathcal{P}(L^\infty)\langle\alpha, \gamma\rangle L_k^{\top op}$:



In this way, we have extracted a useful, structural abstract interpretation from a domain's inverse-limit construction.

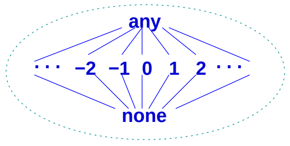
An element, $(d^n, \bot) \in L_k^{\top op}$, names the property of a list having at least n-many d-elements, and (d^n, nil) names the property of a list of exactly length n. The next section shows how to replace L^∞ by $L_k^{\top op}$ within the denotational semantics of Figure 7 and obtain an abstract interpretation.

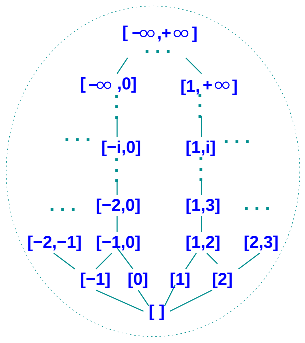
Other abstract domains can be synthesized by means of inverse limits and collecting domains. The *Sign* domain in Figure 3 is derived from these Scott-domain definitions:

$$N = \{1\}_\bot \oplus N, \quad \text{where } \oplus \text{ denotes disjoint sum with merged } \bot\text{s}$$
$$S = (N + \{0\} + N)_\bot$$

S denotes the integers partitioned into the negatives, zero, and the positives. The approximating domain, $S_1 = (N_0 + \{0\} + N_0)_\bot$, where $N_0 = \{\bot\}$, defines $\mathit{Sign} = S_1^{\top op}$ in Figure 3. The Galois connection in Figure 3 goes between the collecting domain of sets of *total values* of S^∞ and *Sign*. We can define better-precision signs-analyses by using domains S_k, $k > 1$, which would distinguish individual integers, e.g, $S_2^{\top op} = \{\top, \mathit{neg}, -1, \mathit{zero}, 1, \mathit{pos}, \bot\}$.

Many abstract domains are defined this way — they are "partitions" [15] of data sets, "crowned" by a $\top$, named by a finite domain from an inverse-limit sequence. But here are two that are not:





The *Const* domain, shown on the left, is used for constant-propagation analysis: a program's variables are analyzed to see if they are uninitialized (*none*), are assigned a single, constant value ($n \in Int$), or are assigned multiple values (*any*) [5]. Rather than an approximating domain, *Const* is $N^{\infty\top op}$, where N^{∞} is the inverse limit of $N = (\{0\} + N)_{\perp}$. In practice, the elements of *Const* are generated on-the-fly while the program is analyzed, such that only a finite number of them appear in the analysis.

On the right is the *Interval* domain, which is employed when an analysis determines the range of values that a variable is assigned [2]. The domain is infinite, its elements are generated on-the-fly while a program is analyzed, and its $\gamma : Interval \to \mathcal{P}(Int)$ is $\gamma([a, b]) = \{n \in Int \mid a \leq n \leq b\}$.

Domains like *Const* and *Interval* are "nonstructural" — not approximations of inverse limits. Standard relational domains from abstract interpretation are typically nonstructural, e.g., the *polyhedral domain* [6], whose values describe linear relationships between variables' values in the store. For example, this set of inequalities,

$$\{2\mathtt{x} + 1\mathtt{y} \leq 100,\ \ 4\mathtt{x} + 1\mathtt{y} - 3\mathtt{z} \leq 0,\ \ -1\mathtt{z} \leq 2\}$$

is an abstract value in the polyhedral domain that abstracts the store, $Var \to \Sigma$. Abstract polyhedra are conjunctive propositions of form, $\bigwedge_i((\sum_j(a_{ij} \cdot \mathtt{x}_{ij}) \leq b_i)$, and are implemented as tuples, matrices, or graphs. The values are generated on-the-fly while a program is analyzed. Similar to the polyhedral domain is the octagon domain [16] and the predicate-abstraction domains [7, 17].

Domains can be combined: There are the usual constructions for collecting domains for products, sums, and liftings. Figure 9 shows two such constructions, indexed product and lifting. The indexed product generates

Let D be a Scott-domain, A its approximant, and $\mathcal{P}(D)\langle\alpha,\gamma\rangle A$ the collecting Galois connection.

Set-indexed product: $I \to D$, for set I: $\mathcal{P}(I \to D)\langle\alpha_I,\gamma_I\rangle I \to A$

where $\gamma_I(a_i)_{i\in I} = \{(d_i)_{i\in I} \mid d_i \in \gamma(a_i)\}$
$\alpha_I(S) = (\alpha\{t_i \mid t \in S\})_{i\in I}$

Compressed lift: $D_{\underline{\bot}}$: $\mathcal{P}(D \cup \{\underline{\bot}\})\langle\alpha_{\bot},\gamma_{\underline{\bot}}\rangle A$ (that is, $\underline{\bot}$ is aliased to the existing $\bot \in A$)

where $\gamma_{\underline{\bot}}(a) = \gamma(a) \cup \{\underline{\bot}\}$
$\alpha_{\underline{\bot}}(S) = \alpha(S - \{\underline{\bot}\})$

Figure 9: Compound Galois connections for collecting domains

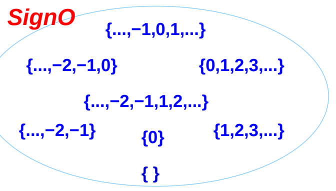
an *independent attribute analysis* [18], where a set of indexed tuples is abstracted to a single tuple that covers the set. The lifting construction *compresses* the $\underline{\bot}$ element with the existing $\bot$ in A and is used when an abstract interpretation ignores nontermination.

5. Open sets, disjunctive completion, and logic

Each abstract domain element names a property set; this suggests a topological connection. For approximating domain, L_k, and $\ell \in L_k$, each $\gamma(\ell)$ is a Scott-basic open set [19, 20] — a "computable property" [8]. Using the closure operator, $\rho = \gamma \circ \alpha : \mathcal{P}(L^\infty) \to \mathcal{P}(L^\infty)$, we have that the family of sets, $\rho[\mathcal{P}(L^\infty)]$, are all Scott-basic opens and the family is closed under (arbitrary) intersection.

It is natural to close $\rho[\mathcal{P}(L^\infty)]$ under arbitrary unions as well to generate a topology on L^∞, one that is coarser than the Scott topology — it defines the "topology of the abstract interpretation." This construction already exists in abstract-interpretation methodology — it is the *disjunctive completion* [14] of the abstract domain, and it adds elements to an abstract domain when more precision is needed for an analysis. For example, the *Sign* domain in Figure 3 can be disjunctively completed to a new domain, *SignO*, by closing $\gamma[\mathit{Sign}]$ under union:

$$\mathit{SignO} = \begin{array}{l}\{\mathit{none},\ \mathit{neg},\ \leq 0,\ \mathit{zero}, \\ \quad \neq 0,\ \geq 0,\ \mathit{pos},\ \mathit{any}\}\end{array} :$$



There is another reason why the disjunctive completion is useful. It reminds us that every abstract domain, $L_k^{\top op}$, defines a "logic," where $\top \in L_k$ denotes *False*, $\bot \in L_k$ denotes *True*, and $L_k^{\top op}$'s $\sqcap$ denotes conjunction and its $\sqsubseteq$ denotes entailment. The disjunctive completion employs $\sqcup$ as disjunction, making a frame [21].

In general terms, an abstract domain A's *logic* is defined as *(i)* primitive assertions, namely, $a \in A$; *(ii)* $f_0^\sharp(\phi)$, for ϕ in A's logic, $f_0^\sharp = \rho \circ f$, and f is forwards complete. (That is, f is a *logical operator*: for all $S \in \rho[\mathcal{P}(\Sigma)]$, $f[S] \in \rho[\mathcal{P}(\Sigma)]$; it maps property sets "on the nose.")

For example, *Sign*'s logic includes

$$\phi \ ::= \ a \mid \phi_1 \sqcap \phi_2 \mid \mathit{negate}^\sharp\ \phi \mid \mathit{filter}^\sharp_{<0}\ \phi, \quad \text{where } a \in \mathit{Sign}$$

because both $\cap$ and *negate* are logical operators (forwards complete). In constrast, union ($\cup$) is not a logical operator for *Sign* (although it is for *SignO*), nor is the successor operation, *succ*.

The logic of the approximating domain is critical to an abstract interpretation: *Only properties that belong to the abstract domain's logic may be soundly verified by the abstract interpretation.* This makes the forwards-completeness property critical to the design of an abstract interpretation.

The above development can be read as naive domain logic as presented by Abramsky [22], where a domain like L^∞ is generated from a set of atomic (finite) elements, which are the primitive propositions (observable properties) in the logic, closed under frame-like axioms.

6. Abstract denotational semantics

Recall from Section 2 that a Galois connection, $\mathcal{P}(\Sigma)\langle\alpha,\gamma\rangle A$, models subsets of Σ as elements of A. Computation by $f : \Sigma \rightharpoonup \Sigma$ is modelled by $f^\sharp : A \to A$ such that $f[\gamma(a)] \subseteq \gamma(f^\sharp(a))$, and the most precise such $f^\sharp$ is $f_0^\sharp = \alpha \circ f \circ \gamma$.

A Galois connection induces an abstract interpretation of a language's denotational semantics: Replace Σ by A and replace functions, $f : \Sigma \to \Sigma_\bot$

Abstract store domain: $\sigma \in \Sigma^\sharp = \mathit{Var} \rightarrow L_k^{\top op}$
Collecting Galois connections for Scott-domains:

L^∞:	$\mathcal{P}(L^\infty)\langle\alpha,\gamma\rangle L_k^{\top op}$
$\Sigma = \mathit{Var} \rightarrow L^\infty$:	$\mathcal{P}(\Sigma)\langle\alpha_{\mathit{Var}},\gamma_{\mathit{Var}}\rangle\Sigma^\sharp$, defined in Figures 7, 8, and 9.
$\Sigma_{\underline{\perp}}$:	$\mathcal{P}(\Sigma_{\underline{\perp}})\langle\alpha_{\underline{\perp}},\gamma_{\underline{\perp}}\rangle\Sigma^\sharp$

$\mathcal{G}^\sharp : \mathit{Guard} \rightarrow \Sigma^\sharp \rightarrow \Sigma^\sharp$
$\quad \mathcal{G}^\sharp[\![\mathtt{G}]\!] = \alpha_{\underline{\perp}} \circ \mathcal{G}[\![\mathtt{G}]\!] \circ \gamma_{\mathit{Var}}$

$\mathcal{E}^\sharp : \mathit{Expression} \rightarrow \Sigma^\sharp \rightarrow L_k^{\top op}$
$\quad \mathcal{E}^\sharp[\![\mathtt{x}]\!]\sigma = \mathit{lookup}^\sharp\ [\![\mathtt{x}]\!]\ \sigma$
$\qquad$ where $\mathit{lookup}^\sharp\ v = \alpha \circ \mathit{lookup}\ v \circ \gamma_{\mathit{Var}}$, that is, $\mathit{lookup}^\sharp\ v\ \sigma = \sigma(v)$
$\quad \mathcal{E}^\sharp[\![\mathtt{tl\ E}]\!]\sigma = \mathit{tail}^\sharp(\mathcal{E}^\sharp[\![\mathtt{E}]\!]\sigma)$
$\qquad$ where $\mathit{tail}^\sharp = \alpha \circ \mathit{tail} \circ \gamma$,
$\qquad$ that is, $\mathit{tail}^\sharp(a,\ell) = \ell$; $\mathit{tail}^\sharp(\mathit{nil}) = \perp = \mathit{tail}^\sharp(\perp)$
$\quad \mathcal{E}^\sharp[\![\mathtt{cons\ a\ E}]\!]\sigma = \mathit{cons}^\sharp\ \mathtt{a}\ (\mathcal{E}^\sharp[\![\mathtt{E}]\!]\sigma)$
$\qquad$ where $\mathit{cons}^\sharp(a,v) = \alpha \circ \mathit{cons}\ a \circ \gamma$, that is, $\mathit{cons}^\sharp\ a\ \ell = (a,\ell)$

$\mathcal{C}^\sharp : \mathit{Command} \rightarrow \Sigma^\sharp \rightarrow \Sigma^\sharp$
$\quad \mathcal{C}^\sharp[\![\mathtt{x = E}]\!]\sigma = \mathit{update}^\sharp\ [\![\mathtt{x}]\!]\ (\mathcal{E}^\sharp[\![\mathtt{E}]\!]\sigma)\ \sigma$
$\qquad$ where $\mathit{update}^\sharp[\![\mathtt{x}]\!] = \alpha_{\underline{\perp}} \circ \mathit{update}[\![\mathtt{x}]\!] \circ (\gamma \times \gamma_{\mathit{Var}})$,
$\qquad$ that is, $\mathit{update}^\sharp\ v\ \ell\ \sigma = \sigma + [v \mapsto \ell]$
$\quad \mathcal{C}^\sharp[\![\mathtt{C_1;C_2}]\!] = \mathcal{C}^\sharp[\![\mathtt{C_2}]\!] \circ \mathcal{C}^\sharp[\![\mathtt{C_1}]\!]$
$\quad \mathcal{C}^\sharp[\![\mathtt{if}\ (\mathtt{G}_i : \mathtt{C}_i)_I \mathtt{fi}]\!] = \bigsqcup_{i\in I} \mathcal{C}^\sharp[\![\mathtt{C}_i]\!] \circ \mathcal{G}^\sharp[\![\mathtt{G}_i]\!]$
$\quad \mathcal{C}^\sharp[\![\mathtt{while\ B\ do\ C}]\!] = \mathit{lfp}\ \lambda f.\, \mathcal{G}^\sharp[\![\neg\mathtt{G}]\!] \sqcup (f \circ \mathcal{C}^\sharp[\![\mathtt{C}]\!] \circ \mathcal{G}^\sharp[\![\mathtt{G}]\!])$

Figure 10: Abstract interpretation derived from $\mathcal{P}(L^\infty)\langle\alpha,\gamma\rangle L_k^{\top op}$

by some $f^\sharp : A \rightarrow A$, say, $f_0^\sharp$. An induction proof shows that the resulting valuation, $\mathcal{C}^\sharp[\![\mathtt{C}]\!]$, is sound for $\mathcal{C}[\![\mathtt{C}]\!]$, for all phrases, C, in the language, because soundness is preserved by function composition and joins. Figure 10 shows the abstract denotational semantics that results from the Galois connection, $\mathcal{P}(L^\infty)\langle\alpha,\gamma\rangle L_k^{\top op}$, and the two constructions from Figure 9. This style of abstract interpretation was first proposed by Donzeau-Gouge [23] and Neilson [24, 25, 26].

Here is an example abstract denotation: Let $\sigma_0 = [[\![\mathtt{x}]\!] \mapsto \perp] \in \Sigma^\sharp$, that is, x might be any L^∞-value at all (because $\gamma(\perp) = L^\infty$):

$$\mathcal{C}^\sharp[\![\mathtt{if}\ (\mathtt{isNil\ x:\ x = cons\ d0\ x}),\ (\mathtt{isNonNil\ x:\ x = x})\ \mathtt{fi}]\!]\sigma_0$$
$$= (\mathcal{C}^\sharp[\![\mathtt{x = cons\ d0\ x}]\!] \circ \mathcal{G}^\sharp[\![\mathtt{isNil\ x}]\!])\sigma_0 \sqcup (\mathcal{C}^\sharp[\![\mathtt{x = x}]\!] \circ \mathcal{G}^\sharp[\![\mathtt{isNonNil\ x}]\!])\sigma_0$$

Now,

$$\begin{aligned}&\mathcal{G}^\sharp[\![\texttt{isNil x}]\!]\sigma_0 = (\alpha_{\underline{\perp}} \circ \mathcal{G}[\![\texttt{isNil x}]\!] \circ \gamma_{Var})\sigma_0\\ &= (\alpha_{\underline{\perp}} \circ \mathcal{G}[\![\texttt{isNil x}]\!])\{[[\![\texttt{x}]\!] \mapsto \ell] \mid \ell \in L^\infty\}\\ &= \alpha_{\underline{\perp}}\{[[\![\texttt{x}]\!] \mapsto nil],\ \underline{\perp}\} = [[\![\texttt{x}]\!] \mapsto nil]\end{aligned}$$

The abstracted guard calculates the abstract store that covers all stores that satisfy `isNil x`. A similar calculation demonstrates that $\mathcal{G}^\sharp[\![\texttt{isNonNil x}]\!]\sigma_0 = \alpha_{\underline{\perp}}(\{[[\![\texttt{x}]\!] \mapsto (d,\ell)] \mid \ell \in L^\infty\} \cup \{\underline{\perp}\}) = [[\![\texttt{x}]\!] \mapsto (d,\perp)]$. We complete the derivation:

$$\begin{aligned}&\mathcal{C}^\sharp[\![\texttt{x = cons d0 x}]\!][[\![\texttt{x}]\!] \mapsto nil]\ \sqcup\ \mathcal{C}^\sharp[\![\texttt{x = x}]\!][[\![\texttt{x}]\!] \mapsto (d,\perp)]\\ &=\ (update^\sharp\ [\![\texttt{x}]\!]\ (\mathcal{E}^\sharp[\![\texttt{cons d0 x}]\!][[\![\texttt{x}]\!] \mapsto nil])\ [[\![\texttt{x}]\!] \mapsto nil])\ \sqcup\ [[\![\texttt{x}]\!] \mapsto (d,\perp)]\\ &=\ [[\![\texttt{x}]\!] \mapsto (\texttt{d0}, nil)] \sqcup [[\![\texttt{x}]\!] \mapsto (d,\perp)]\\ &=\ [[\![\texttt{x}]\!] \mapsto (\texttt{d0} \sqcup d,\ nil \sqcup_{L_k^{\top op}} \perp)]\ =\ [[\![\texttt{x}]\!] \mapsto (\texttt{d0} \sqcup d,\ \perp)]\end{aligned}$$

The outcomes are joined, precision is lost, and the result is an abstract store that maps `x` to a non-nil list whose head is $\texttt{d0} \sqcup d$ and whose tail is unknown (i.e., might be any L^∞-value at all).

The example demonstrates how an abstract intepretation is used: an input property is supplied and its output is calculated by derivation. To calculate the output, $f(\sigma_0)$, from a program denotation, $f = \lambda\sigma.F_{f\sigma'}$, we must ensure finite unfolding of the calls, $f\sigma'$, and detectable termination of the unfoldings. To bound the unfolding, we employ "minimal function graph" semantics [27]: Starting from $f\sigma_0$, we generate the subsequent unfoldings, $f\sigma_i$, generating a family of k first-order equations,

$$\begin{aligned}&f\sigma_0 = F_{f\sigma_1}\\ &f\sigma_1 = F_{f\sigma_2}\\ &\cdots\\ &f\sigma_k = F_{f\sigma_j},\ \text{for some } j \leq k\end{aligned}$$

which we solve iteratively. The equation set is guaranteed finite if the abstract domain from which σ ranges is finite (e.g., *Sign* or $L_k^{\top op}$).

If the abstract domain is infinite but has finite height (e.g., *Const*), we force k to be finite by making the argument sequence, $\sigma_0, \sigma_1, \cdots, \sigma_k$, into a chain so that the domain's *finite-height* ensures a finite equation set: when $f(\sigma_i)$ generates the call, $f(\sigma')$, we replace the latter by $f(\sigma_i \sqcup \sigma')$, which can be safely used in place of the former. The abstract domain's finite height bounds the quantity of the generated equation set.

An abstract domain like *Interval* has infinitely ascending chains. In this situation, $\sqcup$ is replaced by a monotonic, extensive *widening function* that generates chains of finite height [1]. For the *Interval* domain, its widening function is defined $widen(\sigma_i, \sigma')$, where σ_i is the ith element in the chain under construction, and σ' is newly appearing in a call, $f(\sigma')$:

$$
\begin{aligned}
&\mathit{widen}([], [c,d]) = [c,d]\\
&\mathit{widen}([a,b],[c,d]) = [a,b],\ \text{if } a \le c \text{ and } d \le b\\
&\mathit{widen}([a,b],[c,d]) = [-\infty,b],\ \text{if } c < a \text{ and } d \le b\\
&\mathit{widen}([a,b],[c,d]) = [a,+\infty],\ \text{if } a \le c \text{ and } b < d\\
&\mathit{widen}([a,b],[c,d]) = [-\infty,+\infty],\ \text{if } c < a \text{ and } b < d
\end{aligned}
$$

Widening operations are also required for polyhedral domains.

Here is an example from Figure 10: For $\mathcal{C}^\sharp[\![\texttt{while NonNil x : x = tl x}]\!] = f$, where $f(\sigma) = \mathcal{G}^\sharp[\![\texttt{Nil x}]\!]\sigma \sqcup f(\mathcal{C}^\sharp[\![\texttt{x = tl x}]\!](\mathcal{G}^\sharp[\![\texttt{NonNil x}]\!]\sigma))$, we calculate from an input property σ_{db}: Let $\sigma_{db} = [\texttt{x} \mapsto (d, \bot)]$ and $\sigma_b = [\texttt{x} \mapsto \bot]$. (Recall, in $L_k^{\top op}$, that $\bot \in L_k^\top$ means "all lists," and $\top \in L_k^\top$ means "no lists.") Now, $\mathcal{C}^\sharp[\![\texttt{while NonNil x : x = tl x}]\!]\sigma_{db} = f\sigma_{db}$, where

$$
\begin{aligned}
f\sigma_{db} &= \mathcal{G}^\sharp[\![\texttt{Nil x}]\!]\sigma_{db} \sqcup f(\mathcal{C}^\sharp[\![\texttt{x = tl x}]\!](\mathcal{G}^\sharp[\![\texttt{NonNil x}]\!]\sigma_{db})\\
&= [\texttt{x} \mapsto \top] \sqcup f(\mathcal{C}^\sharp[\![\texttt{x = tl x}]\!]\sigma_{db})\\
&= [\texttt{x} \mapsto \top] \sqcup f\sigma_b\\
&= f\sigma_b\\
f\sigma_b &= \mathcal{G}^\sharp[\![\texttt{Nil x}]\!]\sigma_b \sqcup f(\mathcal{C}^\sharp[\![\texttt{x = tl x}]\!](\mathcal{G}^\sharp[\![\texttt{NonNil x}]\!]\sigma_b)\\
&= [\texttt{x} \mapsto \mathit{nil}] \sqcup f(\mathcal{C}^\sharp[\![\texttt{x = tl x}]\!]\sigma_{db})\\
&= [\texttt{x} \mapsto \mathit{nil}] \sqcup f\sigma_b
\end{aligned}
$$

We solve these two first-order equations.

The inductive definition format ensures soundness: For $\mathcal{E}[\![\texttt{op}(\texttt{E}_\texttt{i})]\!] = f(\mathcal{E}[\![\texttt{E}_\texttt{i}]\!])$, we define the abstract semantics inductively as $\mathcal{E}^\sharp[\![\texttt{op}(\texttt{E}_\texttt{i})]\!] = f_0^\sharp(\mathcal{E}^\sharp[\![\texttt{E}_\texttt{i}]\!])$, where $f_0^\sharp = \alpha \circ f \circ \gamma$. It is immediate that $\mathcal{E}^\sharp$ is sound for $\mathcal{E}$: $\mathcal{E}[\![\texttt{E}]\!] \circ \gamma = \gamma \circ \mathcal{E}^\sharp[\![\texttt{E}]\!]$ (equivalently stated as $\alpha \circ \mathcal{E}[\![\texttt{E}]\!] = \mathcal{E}^\sharp[\![\texttt{E}]\!] \circ \alpha$).

Recall the two notions of completeness, applied to $\mathcal{E}$:

forwards completeness: For all $\texttt{E}$, $\mathcal{E}[\![\texttt{E}]\!] \circ \gamma = \gamma \circ \mathcal{E}^\sharp[\![\texttt{E}]\!]$
backwards completeness: For all $\texttt{E}$, $\alpha \circ \mathcal{E}[\![\texttt{E}]\!] = \mathcal{E}^\sharp[\![\texttt{E}]\!] \circ \alpha$

As proved by Cousot and Cousot [2], both forms of completeness are preserved by least- and greatest-fixed-point constructions, as well as by function composition and by inductive definition on syntax: If for every equation, $\mathcal{E}[\![\texttt{op}(\texttt{E}_\texttt{i})]\!] = f(\mathcal{E}[\![\texttt{E}_\texttt{i}]\!])$, $f_0^\sharp$ is forwards (resp. backwards) complete for f, then $\mathcal{E}^\sharp$ is forwards (resp. backwards) complete for $\mathcal{E}$. When there is not completeness, the inductive definition of $\mathcal{E}^\sharp$ is sound but may be weaker than the strongest abstract interpretation: $\mathcal{E}^\sharp[\![\texttt{E}]\!] \sqsupseteq \alpha \circ \mathcal{E}[\![\texttt{E}]\!] \circ \gamma$.

As noted earier, the two completeness forms both define strongest-postcondition semantics yet they are inequivalent. To clarify the situation, we study the topology induced by the underlying Galois connection.

7. Topological characterization of completeness

Topology plays a key role in denotational semantics. To solve the domain equation, $D = D \to D$, Scott needed to limit the cardinality of functions on D. Topological continuity was the appropriate criterion: For complete lattice L, Scott defined L's open sets to be those subsets of L that are (i) upwards closed and (ii) closed under tails of chains.[4] The functions that are topologically continuous for the Scott-topology of L are exactly the chain-continuous functions on L. Continuity limited the cardinality of $D \to D$ so that the recursive domain equation had a solution.

Consider the Scott-topology on an algebraic bcpo: D is algebraic iff there is a subset, $F_D \subseteq D$, of finite elements[5] such that for every $d \in D$, $d = \sqcup\{e \in F_D \mid e \sqsubseteq d\}$. Each $e \in F_D$ defines the property of "having e-information level," and the basic open sets for D's Scott-topology are $\{\uparrow e \mid e \in F_D\}$.[6]

Given that topology is the study of computing on properties, one would believe that it would be central to the theory of abstract interpretation [1], which studies exactly this topic. There are indeed some precedents.

In [28], Cousot and Cousot employed topology to establish soundness of convergence: They proposed a T0-topology, the $\sqcup$-*topology*, for complete lattices, where the basic open sets are up-closed and closed under finite meets. As with the Scott topology, a function is chain continuous iff it is $\sqcup$-topologically continuous. (The two topologies coincide for algebraic lattices.) The $\sqcup$-topology explains how computation on an abstract interpretation preserves properties: When lattice L's abstract interpretation is defined by an upper closure operation, $\rho : L \to L$, the $\sqcup$-topology on $\rho[L]$ is exactly the relative topology on L: every open $U' \subseteq \rho[L]$ equals $U \cap \rho[L]$, for some open $U \subseteq L$.

One application where topology has been employed is backwards strictness analysis. A characterization of a strictness-analysis domain as open-set properties was made by Hunt [29], who observed that Clack and Peyton Jones's backwards strictness analysis employed abstract values called *frontiers*, which were finite subsets of a finite lattice, D, that represented up-

[4]That is, for every chain, $C = \{c_0, c_1, \cdots c_i, \cdots\} \subseteq L$, when $\sqcup C \in U$, for open set $U \subseteq L$, then there exists some $c_k \in C$ such that $c_k \in U$ also. This means C's tail, from c_k onwards, is in U.

[5]$e \in D$ is *finite* iff for all chains $C \subseteq D$, $e \sqsubseteq \sqcup C$ implies $e \sqsubseteq c$ for some $c \in C$.

[6]where $\uparrow e = \{d \in D \mid e \sqsubseteq d\}$

closed subsets of D. Since up-closed subsets of a finite lattice are Scott-open, all monotone functions $f : D \to D$ are Scott-continuous, implying f^{-1} maps frontiers to frontiers, ensuring that the analysis preserved strictness properties "on the nose." Dybjer formalized this property for denotational semantics definitions and domain equations, axiomatizing the Scott topology of the latter as well as the law that the inverse of a Scott-continuous function maps open sets to open sets. He then showed strictness analysis is an instance of his axiomatization [30].

The most striking application of topology to abstract domains came from Jensen [31], who utilized Abramsky's domain theory in logical form [22]. Recall that Abramsky applied Stone duality [21] to domain theory, generating a Scott domain from a set of atomic elements that act as primitive propositions in a domain logic, closing them under a set of frame axioms. Jensen observed that one can use a finite subset of the atomic elements with the frame axioms to generate an abstract domain that approximates the domain generated from all the atomic elements. Jensen called his methodology *abstract interpretation in logical form* and applied it to strictness analysis, as did Benton, who proposed his own "strictness logic" [32].

How do these efforts relate to the development in this paper? For abstract domain, ${L_k^\top}^{op}$, its elements name properties that are used in an abstract interpretation: each $\ell \in L_k^\top$ names the set, $\uparrow\ell \subseteq L^\infty$, a Scott-basic open set in L^∞. The collection, $\gamma[L_k^\top]$, is a family that is closed under intersection but not necessarily under union. If we close under union, we have a topology on L^∞, coarser than the Scott-topology. But this analogy fails for relational abstract domains. To resolve the issue, we will assume that the elements in *any* abstract domain define "open sets" like the ones in ${L_k^\top}^{op}$ and develop the consequences.

One defines a topology so to ask, "what are the continuous functions?" In the case of the "topology" defined by an abstract domain, we ask "what are the open, closed, and continuous maps?" We will see that the elements of an overapproximating abstract domain define closed sets and the elements of an underapproximating abstract domain define open sets; we also see that those functions that preserve members of an abstract domain (the closed/open maps) are the forwards-complete functions of abstract-interpretation theory and those functions that reflect members of an abstract domain (the continuous maps) are the backwards-complete functions.

8. Basic definitions

We review core concepts from topology [33]: For a set, Σ, a *topology*, $\mathcal{O}_\Sigma \subseteq \mathcal{P}(\Sigma)$, is a family of property sets, called the *open sets*, that are closed under union (for all $S \subseteq \mathcal{O}_\Sigma$, $\bigcup S \in \mathcal{O}_\Sigma$) and binary intersection ($U_1 \cap U_2 \in \mathcal{O}_\Sigma$ when $U_1, U_2 \in \mathcal{O}_\Sigma$) and include Σ ($\bigcup \mathcal{O}_\Sigma = \Sigma$). The complement, $\sim U = \Sigma - U$, of an open set U is a *closed set*; define $\mathcal{C}_\Sigma = \{\sim U \mid U \in \mathcal{O}_\Sigma\}$. For topology $\mathcal{O}_\Sigma$, a *base* is a subset, $\mathcal{B}_\Sigma \subseteq \mathcal{O}_\Sigma$, such that every $U \in \mathcal{O}_\Sigma$ is the union of some members of the base (for all $U \in \mathcal{O}_\Sigma$, there exists $S \subseteq \mathcal{B}_\Sigma$ such that $\cup S = U$). The members of the base are called *basic-open sets*.

For $S \subseteq \Sigma$, its *interior*, $\iota(S)$, is the largest open set within S; $\iota(S) = \bigcup\{U \in \mathcal{O}_\Sigma \mid U \subseteq S\}$. The smallest closed set enclosing S is its *closure*, $\rho(S) = \bigcap\{K \mid S \subseteq K,\ K \in \mathcal{C}_\Sigma\}$.

A function, $f : \Sigma \to \Sigma$, is *(topologically) continuous* iff for all $s \in \Sigma$ and $V \in \mathcal{O}_\Sigma$, if $f(s) \in V$, then there exists some $U \in \mathcal{O}_\Sigma$ such that $s \in U$ and $f[U] \subseteq V$ (where $f : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ is $f[U] = \{f(x) \mid x \in U\}$). See Figure 13. A crucial result is that f is continuous iff for all $U \in \mathcal{O}_\Sigma$, $f^{-1}(U) \in \mathcal{O}_\Sigma$ also, where $f^{-1}(U) = \{x \in \Sigma \mid f(x) \in U\}$. (As a corollary, f is continuous iff for all $K \in \mathcal{C}_\Sigma$, $f^{-1}(K) \in \mathcal{C}_\Sigma$ also.) Function f is an *open map* iff for all $U \in \mathcal{O}_\Sigma$, $f[U] \in \mathcal{O}_\Sigma$ and it is a *closed map* iff for all $K \in \mathcal{C}_\Sigma$, $f[K] \in \mathcal{C}_\Sigma$.

9. Property families, function preservation and reflection

We now adapt topological concepts to abstract interpretation. For a concrete state set, Σ, choose some $\mathcal{F}_\Sigma \subseteq \mathcal{P}(\Sigma)$ as a family of properties. In Figure 3, the family $\mathit{Sign}_{\mathit{Int}}$ is $\{\emptyset, \{i \mid i < 0\}, \{0\}, \{i \mid i > 0\}, \mathit{Int}\}$.

For each $U \in \mathcal{F}_\Sigma$, its complement is $\sim U = \Sigma - U$; for $\mathcal{F}_\Sigma$, its *complement family*, $\sim \mathcal{F}_\Sigma$, is $\{\sim U \mid U \in \mathcal{F}_\Sigma\}$. E.g., $\sim \mathit{Sign}_{\mathit{Int}}$ is $\{\mathit{Int}, \{i \mid i \geq 0\}, \{i \mid i \neq 0\}, \{i \mid i \leq 0\}, \emptyset\}$. When property family $\mathcal{O}_\Sigma \subseteq \mathcal{P}(\Sigma)$ is closed under unions, then $\mathcal{O}_\Sigma$ is an *open family* and has the interior operator, $\iota : \mathcal{P}(\Sigma) \to \mathcal{O}_\Sigma$. Dually, if a property family $\mathcal{C}_\Sigma$ is closed under intersections, it is a *closed family* (*Moore family* [2]) and has a closure operator, $\rho : \mathcal{P}(\Sigma) \to \mathcal{C}_\Sigma$. $\mathit{Sign}_{\mathit{Int}}$ in Figure 3 is a closed (but not open) family, whose closure operation is the ρ stated in the Figure. If $\mathcal{O}_\Sigma$ is an open family, then its complement is a closed family (and vice versa), where $\bigcap_{i \in I} K_i = \sim \bigcup_{i \in I} \sim K_i$ (and where $\bigcup_{i \in I} U_i = \sim \bigcap_{i \in I} \sim U_i$).

Let $f : \Sigma \rightarrow \Delta$ be a *total* function;[7] define $f : \mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Delta)$ as $f[S] = \{f(s) \in \Delta \mid s \in S\}$. Next, define function inverse, $f^{-1} : \mathcal{P}(\Delta) \rightarrow \mathcal{P}(\Sigma)$, as $f^{-1}(T) = \{s \in \Sigma \mid f(s) \in T\}$.

For property families, $\mathcal{F}_\Sigma$ and $\mathcal{F}_\Delta$, $f : \Sigma \rightarrow \Delta$ is $\mathcal{F}_\Sigma\mathcal{F}_\Delta$*-preserving* iff for all $U \in \mathcal{F}_\Sigma$, $f[U] \in \mathcal{F}_\Delta$. In such a case, $f : \mathcal{F}_\Sigma \rightarrow \mathcal{F}_\Delta$ is well defined. To reduce notation, we use functions, $f : \Sigma \rightarrow \Sigma$, with the same domain and codomain (and we say, "f is $\mathcal{F}_\Sigma$-preserving"), but all results that follow hold for functions with distinct codomains and domains, too.

Definition 3. *For $s \in \Sigma$ and $S \subseteq \Sigma$, let U_s (respectively, U_S) denote a member of $\mathcal{F}_\Sigma$ such that $s \in U_s$ (respectively, $S \subseteq U_S$).*

(i) *For $s \in \Sigma$, $f : \Sigma \rightarrow \Sigma$ is* continuous at s *iff for all $V_{f(s)} \in \mathcal{F}_\Sigma$, there exists some $U_s \in \mathcal{F}_\Sigma$ such that $f[U_s] \subseteq V_{f(s)}$.*

(ii) *For $S \subseteq \Sigma$, f is* continuous at S *iff for all $V_{f[S]} \in \mathcal{F}_\Sigma$, there exists some $U_S \in \mathcal{F}_\Sigma$ such that $f[U_S] \subseteq V_{f[S]}$.*

(iii) *f is* $\mathcal{F}_\Sigma$-reflecting *iff for all $V \in \mathcal{F}_\Sigma$, $f^{-1}(V) \in \mathcal{F}_\Sigma$, that is, f^{-1} is $\mathcal{F}_\Sigma$-preserving.*

Proposition 4. *(i) f is $\mathcal{F}_\Sigma$-reflecting iff f is continuous at S, for all $S \subseteq \Sigma$. (ii) If $\mathcal{F}_\Sigma$ is an open family, then f is $\mathcal{F}_\Sigma$-reflecting iff f is continuous at s, for all $s \in \Sigma$.*

PROOF. We prove (i); (ii) is a standard result [33]. If: for $V \in \mathcal{F}_\Sigma$, consider $f^{-1}(V)$. Because f is continuous at all $S \subseteq \Sigma$, there is some $U_{f^{-1}(V)} \in \mathcal{F}_\Sigma$ such that $f[U_{f^{-1}(V)}] \subseteq V$. But $U_{f^{-1}(V)}$ must equal $f^{-1}(V)$ for this to hold.

Only if: for $S \subseteq \Sigma$, say that $V_S \in \mathcal{F}_\Sigma$. Since f is reflecting, $f^{-1}(V_S) \in \mathcal{F}_\Sigma$. Thus, $f[f^{-1}(V_S)] \subseteq V_S$.$\square$

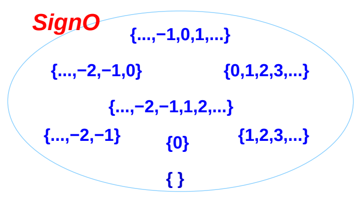
We retain these critical dualities for all f and $\mathcal{F}_\Sigma$:

Proposition 5. *$f : \Sigma \rightarrow \Sigma$ is $\sim\mathcal{F}_\Sigma$-reflecting iff f is $\mathcal{F}_\Sigma$-reflecting.*

f is $\mathcal{F}_\Sigma$-preserving iff $\widetilde{f} = \sim \circ f \circ \sim$ is $\sim\mathcal{F}_\Sigma$-preserving.

In Figure 3, *negate* and *square* are $\mathit{Sign}_{\mathit{Int}}$-reflecting (but *succ* is not). Both functions are $\sim \mathit{Sign}_{\mathit{Int}}$ reflecting, where $\sim \mathit{Sign}_{\mathit{Int}} = \{\mathit{Int}, \{i \mid i \geq 0\}, \{i \mid i \neq 0\}, \{i \mid i \leq 0\}, \emptyset\}$. Since *negate* is $\mathit{Sign}_{\mathit{Int}}$-preserving, $\widetilde{\mathit{negate}}$ is $\sim\mathit{Sign}_{\mathit{Int}}$-preserving, e.g., $\widetilde{\mathit{negate}}\{i \mid i \geq 0\} = \{i \mid i \leq 0\}$. We exploit such dualities in the next section.

[7]The results are best understood with total functions. Partial functions are addressed in a later section.



$\rho_\cup(S) = \cup\{\rho\{s\} \mid s \in S\}$

$f^\sharp = \rho_\cup \circ f$

$f^{\sharp -1}(U) = \cup\{V \in \mathit{Sign}\mathcal{O}_{\mathit{Int}} \mid f^\sharp(V) \subseteq U\}$

For $\mathit{succ}(i) = i + 1$,

$\mathit{succ}^\sharp\{i \mid i < 0\} = \{i \mid i \leq 0\}$	$\mathit{succ}^{\sharp -1}\{i \mid i > 0\} = \{i \mid i \geq 0\}$
$\mathit{succ}^\sharp\{0\} = \{i \mid i > 0\}$	$\mathit{succ}^{\sharp -1}\{0\} = \emptyset$
$\mathit{succ}^\sharp\{i \mid i > 0\} = \{i \mid i > 0\}$	$\mathit{succ}^{\sharp -1}\{i \mid i < 0\} = \emptyset$
$\mathit{succ}^\sharp\{i \mid i \leq 0\} = \mathit{Int}$	$\mathit{succ}^{\sharp -1}\{i \mid i \leq 0\} = \{i \mid i < 0\}$, etc.
$\mathit{succ}^\sharp\{i \mid i \neq 0\} = \mathit{Int}$, etc.	

Figure 11: Using $\mathit{Sign}_{\mathit{Int}} = \{\emptyset, \{i \mid i < 0\}, \{0\}, \{i \mid i > 0\}, \mathit{Int}\}$ as a base for a topology.

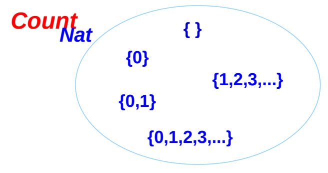
10. Postcondition and precondition analyses

A property family lists the properties that can be computed by an abstract interpretation. Function $f^\sharp : \mathcal{F}_\Sigma \rightarrow \mathcal{F}_\Sigma$ soundly approximates $f : \Sigma \rightarrow \Sigma$ iff for all $V \in \mathcal{F}_\Sigma, f[V] \subseteq f^\sharp(V)$. When $\mathcal{C}_\Sigma$ is a closed family, we use its closure operator, ρ, to define from f its sound, strongest-postcondition approximation, $f^\sharp = \rho \circ f$. A forwards abstract interpretation calculates overapproximating postconditions, and one uses a closed family to generate a postcondition analysis; the literature abounds with examples [4, 1].

What if we desire preconditions from a closed family? We might define $f^\sharp$'s inverse, $f^{\sharp -}_{\mathcal{C}_\Sigma} : \mathcal{C}_\Sigma \rightarrow \mathcal{P}(\mathcal{C}_\Sigma)$, as

$$(\star) \qquad f^{\sharp -}_{\mathcal{C}_\Sigma}(U) = \{V \in \mathcal{C}_\Sigma \mid f^\sharp(V) \subseteq U\}$$

Although this definition is sound, in the sense that $\cup f^{\sharp -}_{\mathcal{C}_\Sigma}(U) \subseteq f^{-1}(U)$, the value $\cup f^{\sharp -}_{\mathcal{C}_\Sigma}(U)$ is not necessarily expressible in the closed family, $\mathcal{C}_\Sigma$. To repair the flaw, we close $\mathcal{C}_\Sigma$ under unions, that is, *we use it as a base for a topology* on Σ, namely, $\mathcal{CO}_\Sigma = \{\cup T \mid T \subseteq \mathcal{C}_\Sigma\}$, which is both an open *and* a closed family. (The closure map $\rho_\cup : \mathcal{CO}_\Sigma \rightarrow \mathcal{CO}_\Sigma$ equals $\rho_\cup(S) = \cup\{\rho\{s\} \mid s \in S\}$.) Now, we approximate with $\mathcal{CO}_\Sigma$: for $f : \Sigma \rightarrow \Sigma$, we define $f^\sharp : \mathcal{CO}_\Sigma \rightarrow \mathcal{CO}_\Sigma$ as $f^\sharp = \rho_\cup \circ f$; we define $f^{\sharp -}_{\mathcal{CO}_\Sigma} : \mathcal{CO}_\Sigma \rightarrow \mathcal{P}(\mathcal{CO}_\Sigma)$ as $f^{\sharp -}_{\mathcal{CO}_\Sigma}(U) = \{V \in \mathcal{CO}_\Sigma \mid f^\sharp(V) \subseteq U\}$, like before; and this makes $f^\sharp$'s weakest



For $Nat = \{0, 1, 2, \cdots\}$,
$Count_{Nat} = \{\emptyset, \{0\}, \{0,1\}, \{1,2,3,\cdots\}, Nat\}$
$\iota(S) = \cup\{U \mid U \subseteq S\}$, e.g.,
$\iota\{0,1,2\} = \{0,1\}$
$\iota\{2,4,6,8,\cdots\} = \emptyset$

For $succ(n) = n + 1$,
$succ^{-o} = \iota \circ succ^{-1}$, e.g.,
$succ^{-o}\{0,1\} = \{0\}$
$succ^{-o}\{0\} = \emptyset = succ^{-o}(\emptyset)$
$succ^{-o}\{1,2,3,\cdots\} = Nat = succ^{-o}(Nat)$

Figure 12: Open family for counting analysis

precondition, $f^{\sharp-1} : \mathcal{CO}_\Sigma \rightarrow \mathcal{CO}_\Sigma$, well defined: $f^{\sharp-1}(U) = \cup f^{\sharp-}_{\mathcal{CO}_\Sigma}(U)$.[8]

$\mathcal{CO}_\Sigma$ is the disjunctive completion construction, seen earlier. Figure 11 shows the disjunctive completion of $Sign_{Int}$ to $Sign\mathcal{O}_{Int}$ and the precondition function for $succ^\sharp$. Now, we have preconditions, but the extra sets generated by the disjunctive completion may make the abstract domain *too large* for a practical static analysis.

If we are primarily interested in preconditions, we should start with an *open* family of properties (one closed under unions), $\mathcal{O}_\Sigma \subseteq \mathcal{P}(\Sigma)$, so that we have straightaway an interior operator, $\iota : \Sigma \rightarrow \mathcal{O}_\Sigma$. We underapproximate the inverses of transition functions: For $f : \Sigma \rightarrow \Sigma$, define $f^{-o} : \mathcal{O}_\Sigma \rightarrow \mathcal{O}_\Sigma$ as $f^{-o} = \iota \circ f^{-1}$. $f^{-o}(\psi)$ calculates the *weakest precondition of f and ψ expressible in $\mathcal{O}_\Sigma$:* for $\phi, \psi \in \mathcal{O}_\Sigma$, if $f[\phi] \subseteq \psi$, then $f[f^{-o}(\psi)] \subseteq \psi$ and $\phi \subseteq f^{-o}(\psi)$.

Disjunctive completions of closed families — topologies — are the standard examples of open families, but Figure 12 defines an open but not closed family, $Count_{Nat}$, for a backwards counting analysis. The successor operation, $succ : Nat \rightarrow Nat$, is $Count_{Nat}$-reflecting, so $succ^{-o} = succ^{-1}$. (See the Figure.) Predecessor ($pred(n + 1) = n$, $pred(0) = 0$) is not reflecting, and $pred^{-o} = \iota \circ pred^{-1}$ yields $pred^{-o}\{0,1\} = \iota\{0,1,2\} = \{0,1\}$, etc. As indicated by research on backwards strictness analysis [32, 30, 29, 31], one should use an open family of properties to generate a precondition analysis.

[8]More precisely stated, it is the *weakest liberal precondition*, as explained in Section 15. Also, since $\mathcal{CO}_\Sigma$ possesses an interior operator, ι, we can define the precondition function as $\iota \circ f^{-1}$ and prove that $f^{\sharp-1} = \iota \circ f^{-1}$ [11].

Because the complement of a closed family is open (and vice versa), we can move from a postcondition analysis to its dual, precondition analysis: Say that $\mathcal{C}_\Sigma$ is closed so that $\mathcal{O}_\Sigma = {\sim}\mathcal{C}_\Sigma$ is open. First, every $\mathcal{C}_\Sigma$-reflecting f is $\mathcal{O}_\Sigma$-reflecting, and for every $\mathcal{C}_\Sigma$-preserving $f : \Sigma \to \Sigma$, $\widetilde{f}$ is $\mathcal{O}_\Sigma$-preserving, by Proposition 5.

Lemma 6. *For all $f : \Sigma \to \Sigma$ and $V \in \mathcal{F}_\Sigma$, ${\sim} f^{-1}(V) = f^{-1}({\sim}V)$.*

For all $V \in \mathcal{F}_\Sigma$, $\widetilde{f^{-1}}(V) = f^{-1}(V)$.

For closed family $\mathcal{C}_\Sigma$ and its complement, $\mathcal{O}_\Sigma = {\sim}\mathcal{C}_\Sigma$, ${\sim} \circ \rho = \iota \circ {\sim}$.

These results yield

Proposition 7. *$\widetilde{(f^{-1})^\sharp}(U) = f^{-o}(U)$, for all $U \in \mathcal{O}_\Sigma$. (Note: $\widetilde{(f^{-1})^\sharp} = {\sim} \circ (f^{-1})^\sharp \circ {\sim}$.)*

Proof. $\widetilde{(f^{-1})^\sharp}(U) = {\sim} \circ \rho \circ f^{-1} \circ {\sim} ({\sim} K)$, where $U = {\sim} K$. This equals ${\sim}\rho(f^{-1}(K)) = \iota({\sim} f^{-1}(K))$, by the previous lemma, which equals $\iota(f^{-1}({\sim} K))$, by the lemma, which equals $f^{-o}(U)$. $\square$

The Proposition says, by using $\mathcal{C}_\Sigma$'s closure operator to define the over-approximating $(f^{-1})^\sharp$, we can compute an underapproximating, weakest-precondition analysis on $\mathcal{O}_\Sigma = {\sim}\mathcal{C}_\Sigma$ defined as $\widetilde{(f^{-1})^\sharp}$.

As an example, consider ${\sim} \mathit{Sign}_{\mathit{Int}} = \{\mathit{Int}, \{i \mid i \geq 0\}, \{i \mid i \neq 0\}, \{i \mid i \leq 0\}, \emptyset\}$, based on Figure 3. This open family's logic includes

$$\psi ::= {\sim} U \mid \psi_1 \cup \psi_2 \mid \mathit{negate}^{-1}\psi \mid \mathit{sq}^{-1}\psi, \quad \text{for } U \in \mathit{Sign}_{\mathit{Int}}$$

Because succ is not $\mathit{Sign}_{\mathit{Int}}$-reflecting, we underapproximate it by $\mathit{succ}^{-o} = \widetilde{(\mathit{succ}^{-1})^\sharp}$. We have $\mathit{succ}^{-o}\{i \mid i \neq 0\} = \{i \mid i \geq 0\}$; $\mathit{succ}^{-o}\mathit{Int} = \mathit{Int}$; and $\mathit{succ}^{-o}(U) = \emptyset$, otherwise. In this fashion, a postcondition analysis based on $\mathcal{C}_\Sigma$ defines a precondition analysis on ${\sim}\mathcal{C}_\Sigma$.

Finally, every $\mathcal{F}_\Sigma$ possesses both a logic for validation (viz., $\mathcal{F}_\Sigma$'s sets and its logical operators) as well as a dual, *refutation logic*: ${\sim}\mathcal{F}_\Sigma$'s logic. We say that S *has property* $\neg\phi$ if $S \subseteq {\sim}\phi$, for ${\sim}\phi \in {\sim}\mathcal{F}_\Sigma$. This is the foundation for three-valued static analyses [34], where one uses a single abstract domain to compute validation, refutation, and "don't know" judgements.

11. From continuity to completeness

There is a correspondence between functions that preserve and reflect property sets and abstract-interpretation-complete functions: Recall that $f : \Sigma \to \Sigma$ is $\mathcal{F}_\Sigma$-preserving iff for all $S \in \mathcal{F}_\Sigma$, $f[S] \in \mathcal{F}_\Sigma$. But this is *exactly the definition of abstract-interpretation forwards completeness* when $\mathcal{F}_\Sigma$ is a closed family. We say that f is $\mathcal{F}_\Sigma$*-forwards complete*. In topological terms, f is a closed map. The forwards-completeness notion also applies when $\mathcal{F}_\Sigma$ is an open family and f is an open map.

We now develop the equivalence of $\mathcal{F}_\Sigma$-reflection to backwards completeness. For $S, S' \subseteq \Sigma$, write $S \leq_{\mathcal{F}_\Sigma} S'$ iff for all $K \in \mathcal{F}_\Sigma, S \subseteq K$ implies $S' \subseteq K$. This is the *specialization ordering* in topology. Write $S \equiv_{\mathcal{F}_\Sigma} S'$ iff $S \leq_{\mathcal{F}_\Sigma} S'$ and $S' \leq_{\mathcal{F}_\Sigma} S$. Note that $S \supseteq S'$ implies $S \leq_{\mathcal{F}_\Sigma} S'$, but the converse need not hold. Say that $f : \Sigma \to \Sigma$ is $\mathcal{F}_\Sigma$*-monotone* if for all $S, S' \in \mathcal{P}(\Sigma)$, $S \leq_{\mathcal{F}_\Sigma} S'$ implies $f[S] \leq_{\mathcal{F}_\Sigma} f[S']$. The following definition is the usual one for abstract-interpretation backwards completeness:

Definition 8. *For property family, $\mathcal{F}_\Sigma$, $f : \Sigma \to \Sigma$ is $\mathcal{F}_\Sigma$-*backwards-complete *iff for all $S, S' \subseteq \Sigma$, $S \equiv_{\mathcal{F}_\Sigma} S'$ implies $f[S] \equiv_{\mathcal{F}_\Sigma} f[S']$.*

Clearly, if f is $\mathcal{C}_\Sigma$-monotone, it is $\mathcal{C}_\Sigma$-backwards-complete, but the converse also holds for a closed family:

Proposition 9. *If $\mathcal{C}_\Sigma$ is a closed family and f is $\mathcal{C}_\Sigma$-backwards-complete, then f is $\mathcal{C}_\Sigma$-monotone.*

Proof. Assume $S \leq S'$ and $f[S] \subseteq K$, for $K \in \mathcal{C}_\Sigma$. Say that ρ is the closure operator for $\mathcal{C}_\Sigma$; then, $\rho \circ f[S] \subseteq K$, because K is closed, implying $\rho \circ f \circ \rho[S] \subseteq K$, by $\mathcal{C}_\Sigma$-backwards-completeness, that is, $(\rho \circ f)(\rho(S)) \subseteq K$. This implies $\rho \circ f[S'] \subseteq K$, because $S \subseteq \rho(S)$ and $S \leq S'$. This gives $\circ f[S'] \subseteq K$.

Proposition 10. *If f is $\mathcal{F}_\Sigma$-reflecting, then it is $\mathcal{F}_\Sigma$-backwards-complete.*

Proof. Assume $S \leq_\Sigma S'$ and show $f[S] \leq_\Sigma f[S']$: Say that $f[S] \subseteq K \in \mathcal{F}_\Sigma$; since f is reflecting, $f^{-1}(K) \in \mathcal{F}_\Sigma$, too, and $S \subseteq f^{-1}(K)$. Because $S \leq_\Sigma S'$, $S' \subseteq f^{-1}(K)$, implying $f[S'] \subseteq K$. □

If $\mathcal{C}_\Sigma$ is a closed family, we use its ρ to prove the converse. Here are the key technical properties:

Lemma 11. *For all $S \subseteq \Sigma$, $S \equiv_{\mathcal{C}_\Sigma} \rho(S)$.*
For all $S, S' \subseteq \Sigma$, $S \equiv_{\mathcal{C}_\Sigma} S'$ iff $\rho(S) = \rho(S')$.

Lemma 12. *The following are equivalent for closed family, $\mathcal{C}_\Sigma$:*
(i) f is $\mathcal{C}_\Sigma$-backwards-complete;
(ii) for all $S \subseteq \Sigma$, $f[S] \equiv_{\mathcal{C}_\Sigma} f[\rho(S)]$;
(iii) $\rho \circ f = \rho \circ f \circ \rho$.

Proof. *(i) implies (ii)*: From Lemma 11, $S \equiv_{\mathcal{C}_\Sigma} \rho(S)$; apply (i).
(ii) implies (iii): From (ii), $f[S]$ and $f[\rho(S)]$ are contained in exactly the same closed sets, hence their closures are equal.
(iii) implies (i): Let $S \equiv_{\mathcal{C}_\Sigma} S'$ and $f[S] \subseteq K$ for arbitrary $K \in \mathcal{C}_\Sigma$. Then, $\rho \circ f[S] \subseteq K$ and then $\rho \circ f[\rho(S)] \subseteq K$, by (iii). By Lemma 11, $\rho \circ f[\rho(S')] \subseteq K$, implying $f[\rho(S')] \subseteq K$.

For a closed family, reflection (topological continuity) is backwards completeness:

Theorem 13. *For $\mathcal{C}_\Sigma$, $f : \Sigma \to \Sigma$ is $\mathcal{C}_\Sigma$-backwards-complete iff f is $\mathcal{C}_\Sigma$-reflecting.*

Proof. The if-part is already proved. For the only-if part, assume $f[S] \subseteq K \in \mathcal{C}_\Sigma$ and show there is some $L_S \in \mathcal{C}_\Sigma$ such that $f[L_S] \subseteq K$. Let $\rho(S)$ be the L_S: we have $f[\rho(S)] \equiv_{\mathcal{C}_\Sigma} f[S]$ by the previous Lemma, which implies $f[\rho(S)] \subseteq K$. □

Corollary 14. *(i) if f is $\mathcal{C}_\Sigma$-backwards-complete, then f^{-1} is both $\mathcal{C}_\Sigma$- and $\sim\mathcal{C}_\Sigma$-forwards complete.*
(ii) f is $\mathcal{C}_\Sigma$-forwards complete iff $\tilde{f}$ is $\sim\mathcal{C}_\Sigma$-forwards complete.

Proof. By Proposition 5 and the previous Theorem.

12. Relation to partial-order backwards completeness

The crucial characterization of backwards completeness by Giacobazzi, et al. [9] was made in a "frame-theory" presentation [21], where $(\mathcal{P}(\Sigma), \subseteq)$ is abstracted to a complete lattice, $(D, \sqsubseteq)$, and $\mathcal{C}_\Sigma$ is abstracted to $\rho[D] \subseteq D$, namely, the fixed points of an upper closure map, $\rho : D \to D$. We can rephrase their work in terms of our development:

First, define $f^- : D \to \mathcal{P}(D)$ as $f^-(d) = \{e \in D \mid f(e) \sqsubseteq d\}$. When f^- is chain-continuous, then $f^-(d)$ has a set of maximal points, denoted by $max(f^-(d))$. When f is an *additive* function, that is, $f(\sqcup S) = \sqcup_{d \in S} f(d)$, for all $S \subseteq D$, then $max(f^-(d))$ is a singleton set. *This is the case for the point-set topology used in the previous section.*

Let $\rho[D]$ define D's closed family of "properties" and let $f : D \to D$ be chain-continuous. First, (i) f is *continuous at* $d \in D$ iff for all $e \in \rho[D]$, if $f(d) \sqsubseteq e$, then there exists $d' \in \rho[D]$ such that $d \sqsubseteq d'$ and $f(d') \sqsubseteq e$. Next, (ii) f is *ρ-reflecting* iff for all $e \in \rho[D]$, $max((f^-(d)) \subseteq \rho[D]$ (that is, the maximum elements of $f^-(d)$ are in $\rho[D]$). It is easy to prove that (i) and (ii) are equivalent.

We define $d \equiv_{\rho[D]} d'$ iff for all $e \in \rho[D]$, $d \sqsubseteq e$ iff $d' \sqsubseteq e$, that is, iff $\rho(d) = \rho(d')$. This yields the definition of backwards completeness: f is ρ-backwards complete if $d \equiv_{\rho[D]} d'$ implies $f(d) \equiv_{\rho[D]} f(d')$ for all $d, d' \in D$, that is, $\rho \circ f = \rho \circ f \circ \rho$. We have immediately the main result of Giacobazzi, et al. [9] in the "frame theory": $f : D \to D$ is ρ-backwards complete iff it is ρ-reflecting.

The characterizations of forwards completeness as property preservation and backwards completeness as property reflection (continuity) link the shell constructions of Giacobazzi, et al. [10, 9], to refinements of topologies and the characterization of function continuity to convergence of nets [33].

13. Application to structural approximating domains

For domain L^∞ and its finite approximants, L_k, consider the relationship between the Scott-continuous functions, $f : L^\infty \to L^\infty$, and the backwards-complete functions for each $\mathcal{P}(L^\infty)\langle \alpha^k, \gamma^k \rangle L_k^{\top op}$, $k \geq 0$. First, all functions f are trivially L_0-backwards complete (that is, backwards complete for $\mathcal{P}(L^\infty)\langle \alpha^0, \gamma^0 \rangle L_0^{\top op}$). Since the collection of property sets defined by $\gamma^k[L_k]$ is a subset of those for $\gamma^{k+1}[L_{k+1}]$, any L_k-backwards complete f is L_j-backwards complete for $j < k$.

Consider the domain defined in Figure 8:

- There is a Scott-continuous function, $f : L^\infty \to L^\infty$, that is not L_k-backwards complete for all $k > 0$. Define f as follows: $f(d^k, nil) = nil$, for all $k \geq 0$, and $f(\ell) = \bot$, otherwise. This function is Scott-continuous. Consider $f^{-1}\{nil\}$; it is exactly all the total, finite lists in L^∞, and for no finite element $e \in L^\infty$ does this set equal $\uparrow e$. (Nor does the union of the upclosed sets of finite elements in any L_k equal $f^{-1}(nil)$ — the union of the basic opens of *all* finite lists in L^∞ are required.)

- For each $k > 0$, there is a monotone, L_k-backwards complete function that is not Scott-continuous. For k, define $f_k : L^\infty \to L^\infty$ as follows: $f(\bot) = \bot$; for $j < k$, $f_k(d^j, nil) = (d^j, nil)$ and $f_k(d^j, \bot) = (d^j, \bot)$. For $j \geq k$, $f_k(d^j, nil) = (d^k, \bot)$; $f_k(d^j, \bot) = (d^k, \bot)$. Finally, define $f_k(d^\infty) = d^\infty$. This makes f_k monotone and backwards complete but Scott-discontinuous. The result does not change when the sets defined by L_k are closed under union.

These results are not surprising, because the property family for each L_k-domain is coarser than the Scott topology for the corresponding domain. They *are* frustrating, however, because they show how difficult it is to establish a homomorphism property from a concrete to an abstract denotational semantics.

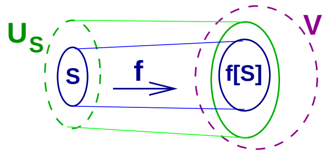
14. Completeness for open families

How do the definitions of forwards- and backwards-completeness relate to open families? Let $\mathcal{O}_\Sigma$ be open (closed under unions) and $\iota : \mathcal{P}(\Sigma) \to \mathcal{O}_\Sigma$ be its interior map. Recall that open families are used for precondition analyses, so for $f : \Sigma \to \Sigma$, we focus upon $f^{-1} : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$, defined as $f^{-1}(S) = \{s \in \Sigma \mid f(s) \in S\}$. The weakest precondition transformer for f in $\mathcal{O}_\Sigma$ is $\iota \circ f^{-1} : \mathcal{O}_\Sigma \to \mathcal{O}_\Sigma$.

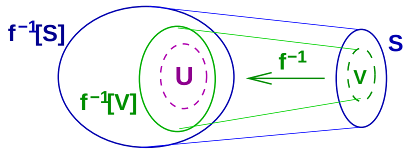
$\mathcal{O}_\Sigma$-forwards completeness for f^{-1} is defined $f^{-1} \circ \iota = \iota \circ f^{-1} \circ \iota$, that is, f^{-1} maps open sets to open sets, that is, f is topologically continuous. Stated completely, f^{-1} is $\mathcal{O}_\Sigma$-forwards complete iff f^{-1} is $\mathcal{O}_\Sigma$-preserving iff f is $\mathcal{O}_\Sigma$-reflecting iff f is $\sim\mathcal{O}_\Sigma$-reflecting. This is the classic pre- post-condition duality of predicate transformers [35].

We can define $\mathcal{O}_\Sigma$-backwards completeness for f^{-1} as $\iota \circ f^{-1} = \iota \circ f^{-1} \circ \iota$. But backwards completeness in $\mathcal{O}_\Sigma$ is *not* a statement of f's continuity — the definition of the specialization ordering in Section 11 is suitable for closed sets, not opens.

But we can dualize it: For $\mathcal{O}_\Sigma$ and $S, S' \subseteq \Sigma$, say that $S \leq_{\mathcal{O}_\Sigma} S'$ iff for every $U \in \mathcal{O}_\Sigma$, when $U \subseteq S$, then $U \subseteq S'$ as well. That is, $S \leq_{\mathcal{O}_\Sigma} S'$ when Ss interior falls within S's. Then, $S \equiv_{\mathcal{O}_\Sigma} S'$ iff $S \leq_{\mathcal{O}_\Sigma} S'$ and $S' \leq_{\mathcal{O}_\Sigma} S$, that is, S and S' have the same interior: $\iota(S) = \iota(S')$. It is easy to prove



(i) For $S \subseteq \Sigma$, f is *continuous at* S iff whenever $f[S] \subseteq V \in \mathcal{F}_\Sigma$, there exists some $U \in \mathcal{F}_\Sigma$ such that $S \subseteq U$ and $f[U] \subseteq V$.



(ii) For $S \subseteq \Sigma$, $f^{-1} : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ is *dual continuous at* S iff whenever $f^{-1}[S] \supseteq U \in \mathcal{F}_\Sigma$, then there exists $V \in \mathcal{F}_\Sigma$, $V \subseteq S$, such that $f^{-1}[V] \supseteq U$.

Figure 13: Continuity and dual continuity at a set

that f^{-1} is backwards-$\mathcal{O}_\Sigma$ complete iff for all $S, S' \subseteq \Sigma$, $S \equiv_{\mathcal{O}_\Sigma} S'$ implies $f^{-1}(S) \equiv_{\mathcal{O}_\Sigma} f^{-1}(S')$.

Backwards completeness for an open family and f^{-1} is a "dual continuity" property. Say that $f^{-1} : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ is *dual continuous* at $S \subseteq \Sigma$ iff for all $U \in \mathcal{O}_\Sigma$, if $f^{-1}[S] \supseteq U$ then there exists $V \in \mathcal{O}_\Sigma$, $V \subseteq S$, such that $f^{-1}[V] \supseteq U$. Figure 13 depicts dual continuity at a set.

Theorem 15. *For open family $\mathcal{O}_\Sigma$ and $f : \Sigma \to \Sigma$, f^{-1} is dual continuous for all $S \subseteq \Sigma$ iff f^{-1} is $\mathcal{O}_\Sigma$-backwards complete, that is, $\iota \circ f^{-1} = \iota \circ f^{-1} \circ \iota$.*

15. Partial functions

The examples in Sections 1 and 2 used partial functions of arity $\Sigma \rightharpoonup \Sigma$. The completeness results proved in the previous sections used total functions, of arity $\Sigma \to \Sigma$. We now reconcile this discrepency and expose the two forms of precondition analysis.

The examples based on partial functions, $f : \Sigma \rightharpoonup \Sigma$, used this definition of function image: $f[S] = \{f(\sigma) \in \Sigma \mid \sigma \in S\}$, which ignores those $\sigma_0 \in S$ such that $f(\sigma_0) = \underline{\perp}$. When $f(\sigma_0) = \underline{\perp}$ then $\{f(\sigma_0)\} \subseteq U$ for every $U \in \mathcal{F}_\Sigma$. As a consequence, the definition of inverse image cannot be merely $f^{-1}(S) = \{\sigma \in \Sigma \mid f(\sigma) \in S\}$, because this omits those σ_0 such that $f(\sigma_0) = \underline{\perp}$. One repair is to use the definition, $f^{-1}(S) = \cup\{S' \subseteq \Sigma \mid f[S'] \subseteq S\}$, but there is the unpleasant consequence that when $f(\sigma_0) = \underline{\perp}$, then both $\sigma_0 \in f^{-1}(U)$ as well as $\sigma_0 \in f^{-1}(\sim U)$.

It is better to model $f : \Sigma \rightharpoonup \Sigma$ as the total function, $f : \Sigma \to \Sigma_{\underline{\perp}}$, as one does in denotational semantics. The examples in Sections 1 and 2 tacitly use closed families on the space, $\Sigma \cup \{\underline{\perp}\}$, such that for every $V \in \mathcal{C}_{\Sigma\cup\{\underline{\perp}\}}$, $\underline{\perp} \in V$.

When a property family, $\mathcal{F}_\Sigma$, extends to the space $\Sigma_{\underline{\perp}} = \Sigma \cup \{\underline{\perp}\}$ such that $\mathcal{F}_{\Sigma\cup\{\underline{\perp}\}} = \{V \cup \{\underline{\perp}\} \mid V \in \mathcal{F}_\Sigma\}$, we say that $\mathcal{F}_\Sigma$ *is* $\underline{\perp}$*-inclusive.* In practice, property families used for calculating postconditions of partial functions are $\underline{\perp}$-inclusive, because termination is undecidable. The result is a *partial correctness* postcondition analysis. Now we can use the classical definitions of function image and preimage from Section 8 and retain the crucial property that $f^{-1}(V)$ and $f^{-1}({\sim}V)$ form a partition of Σ, for every $f : \Sigma \to \Sigma_{\underline{\perp}}$.

Section 10 defined a precondition semantics for closed families. It is worth reviewing. Consider this partial integer-square-root function, $sqrt : Int \rightharpoonup Int$:

$$\begin{array}{l} sqrt(0) = 0 \\ sqrt(i) = j, \ \ \text{if } i > 0, \ \text{such that } j > 0, \ j * j \le i, \ \text{and } (j+1)*(j+1) > i \\ sqrt(i) = \underline{\perp}, \ \ \text{if } i < 0 \end{array}$$

We have $sqrt\{-2,-1,0\} = \{\underline{\perp}, 0\}$, $sqrt\{4, 8, 10\} = \{2, 3\}$, etc.

We employ the $\underline{\perp}$-inclusive property family, $Sign_{Int} = \{none, neg, zero, pos, any\}$, from Figure 3. *Without ambiguity, we use the same property names for* $Sign_{Int\cup\{\underline{\perp}\}} = \{none, neg, zero, pos, any\}$, with the assumption that $\underline{\perp}$ belongs to each named set.

Then, $sqrt^{-1}[zero] = zero \cup neg$, because $sqrt[neg] = \{\underline{\perp}\}$ and $\underline{\perp} \in zero \in Sign_{Int\cup\{\underline{\perp}\}}$. For that matter, $sqrt^{-1}[neg] = neg$. This indicates that a $\underline{\perp}$-inclusive property family computes *weakest liberal preconditions*, where termination is not a necessary condition for membership.

Since $Sign_{Int}$ is a closed family, so is $Sign_{Int\cup\{\underline{\perp}\}}$; the latter's closure operator is defined $\rho_{\underline{\perp}}(S) = \rho(S) \cup \{\underline{\perp}\}$. We define $sqrt$'s approximation as $sqrt^\sharp = \rho_{\underline{\perp}} \circ sqrt$ (e.g., $sqrt^\sharp(zero) = zero$, $sqrt^\sharp(pos) = pos$, $sqrt^\sharp(neg) = none$, with the assumption that $\underline{\perp}$ belongs to each named answer set).

Section 10 showed that that one defines $sqrt$'s precondition for a closed family as follows: For $U \in \mathcal{C}_{\Sigma\cup\{\underline{\perp}\}}$,

$$sqrt^{\sharp^{-1}}(U) = \cup\{V \in Sign_{Int} \mid sqrt^\sharp(V) \subseteq U\}$$

For the example, we close $Sign_{Int}$ under unions, producing $SignO_{Int}$ (see Figure 11), which we decree is $\underline{\perp}$-inclusive. This makes $sqrt^{\sharp^{-1}}$ soundly underapproximate $sqrt^{-1}$.

But say we want precondition analysis for $sqrt$ that demands termination as necessary for membership. When property family $\mathcal{F}_\Sigma$ extends to $\Sigma \cup \{\underline{\perp}\}$ such that $\mathcal{F}_{\Sigma\cup\{\underline{\perp}\}} = \mathcal{F}_\Sigma$, that is, for every $U \in \mathcal{F}_{\Sigma\cup\{\underline{\perp}\}}$, $\underline{\perp} \notin U$, we say that $\mathcal{F}_\Sigma$

is $\underline{\perp}$-*exclusive.* In practice, open families that calculate weakest preconditions are $\underline{\perp}$-exclusive. In the case of $sqrt$, we return to the property family $SignO_{Int}$, which possesses the interior operator, $\iota(S) = \cup\{V \in Sign_{Int} \mid V \subseteq S\}$. So, $sqrt^{\flat-1}$ is $\iota \circ sqrt^{-1}$ such that $sqrt^{\flat-1}(zero) = zero$, $sqrt^{\flat-1}(neg) = none$, etc.

The development in this section is expressible within powerdomain theory of denotational semantics, where partial functions are defined with arity, $\Sigma \rightarrow \Sigma_{\underline{\perp}}$, and weakest-liberal-preconditions are defined with arity $\mathcal{P}_L(\Sigma) \rightarrow \mathcal{P}(\Sigma)$, where $\mathcal{P}_L(\Sigma)$ is the lower powerdomain [36, 37, 13], whose sets are downwards closed in $\Sigma_{\underline{\perp}}$. Weakest preconditions are defined $\mathcal{P}_U(\Sigma) \rightarrow \mathcal{P}(\Sigma)$, where $\mathcal{P}_U(\Sigma)$ is the upper powerdomain [37, 38], whose sets are upwards closed in $\Sigma_{\underline{\perp}}$.

16. Nondeterminism and semicontinuity

Nondeterministic systems use transition *relations* on $\Sigma \times \Sigma$, which we treat as functions of arity, $f : \Sigma \rightarrow \mathcal{P}(\Sigma)$. The property family for $\mathcal{P}(\Sigma)$ is different from Σ's and depends on how we define f's preimage, a map, $\mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)$. We have two choices: for $S \subseteq \Sigma$,

$$\begin{aligned} pre_f(S) &= \{c \in \Sigma \mid f(c) \cap S \neq \emptyset\} \\ \widetilde{pre}_f(S) &= \{c \in \Sigma \mid f(c) \subseteq S\} \end{aligned}$$

The following definitions come from Vietoris via Smyth [8]:

Definition 16. *For property family, $\mathcal{F}_\Sigma \subseteq \Sigma$,*
$f : \Sigma \rightarrow \mathcal{P}(\Sigma)$ *is* lower semicontinuous for $\mathcal{F}_\Sigma$ *iff pre_f is $\mathcal{F}_\Sigma$-preserving.*
$f : \Sigma \rightarrow \mathcal{P}(\Sigma)$ *is* upper semicontinuous for $\mathcal{F}_\Sigma$ *iff $\widetilde{pre}_f$ is $\mathcal{F}_\Sigma$-preserving.*

Say we want pre_f in the logic for $\mathcal{F}_\Sigma$; what property family for $\mathcal{P}(\Sigma)$ is appropriate? The answer was found by Smyth [8]: define $\mathcal{O}^L_{\mathcal{F}_\Sigma} \subseteq \mathcal{P}(\mathcal{P}(\Sigma))$ to be the open family generated by taking all unions of the base, $\mathcal{B}^L_{\mathcal{F}_\Sigma} = \{\exists U \mid U \in \mathcal{F}_\Sigma\}$, where $\exists U = \{S \subseteq \Sigma \mid S \cap U \neq \emptyset\}$. (Read $\exists U$ as "all the sets that meet property U"). Indeed, for all $U \in \mathcal{F}_\Sigma$, $f^{-1}(\exists U) = pre_f(U)$. $\mathcal{O}^L_{\mathcal{F}_\Sigma}$ is called the *lower topology based on* $\mathcal{F}_\Sigma$. When $\mathcal{F}_\Sigma$ is open, we apply this result, due to Smyth [8]:

Proposition 17. *If $\mathcal{O}_\Sigma \subseteq \Sigma$ is an open family for Σ, then $f : \Sigma \rightarrow \mathcal{P}(\Sigma)$ is lower semicontinuous for $\mathcal{O}_\Sigma$ iff f is $\mathcal{O}_\Sigma\mathcal{O}^L_{\mathcal{O}_\Sigma}$-reflecting.*

That is, pre_f lies in the logic for $\mathcal{O}_\Sigma$ iff f is $\mathcal{O}_\Sigma\mathcal{O}^L_{\mathcal{O}_\Sigma}$-reflecting. Smyth used this result to explain the lower-powerdomain construction of denotational semantics in topological terms.

For abstract interpretation $\mathcal{O}_\Sigma$, for $f : \Sigma \to \mathcal{P}(\Sigma)$, we must compute f's preimage in $\mathcal{O}_\Sigma$'s logic, that is, as a function of arity, $\mathcal{O}_\Sigma \to \mathcal{O}_\Sigma$. If f is lower semicontinuous, we use pre_f itself, thanks to the above proposition. But if f is not lower semicontinuous, we use $\mathcal{O}_\Sigma$'s interior operator, ι, to (under)approximate pre_f by $(\iota \circ pre_f) : \mathcal{O}_\Sigma \to \mathcal{O}_\Sigma$, like in Section 10.

We can dualize the previous development and discover a well-known technique for approximating $\widetilde{pre}_f$ within a closed family: As usual, define $\mathcal{C}_\Sigma = {\sim}\mathcal{O}_\Sigma$; we can calculate that ${\sim}\mathcal{O}^L_{\mathcal{O}_\Sigma}$ is the closed family whose members are all the intersections of sets taken from the (co)base, $\mathcal{B}^U_{\mathcal{C}_\Sigma} = \{\forall K \mid K \in \mathcal{C}_\Sigma\}$, where $\forall K = \{S \subseteq \Sigma \mid S \subseteq K\}$. (Read $\forall K$ as "all the sets covered by property K.") Indeed, for all $K \in \mathcal{C}_\Sigma$, $f^{-1}(\forall K) = \widetilde{pre}_f(K)$. We name this closed family: $\mathcal{C}^U_{\mathcal{C}_\Sigma} = {\sim}\mathcal{O}^L_{\mathcal{O}_\Sigma}$.

Corollary 18. *Let $\mathcal{C}_\Sigma$ be a closed family and define $\mathcal{O}_\Sigma = {\sim}\mathcal{C}_\Sigma$.*

pre_f is $\mathcal{O}_\Sigma$-preserving iff $\widetilde{pre}_f$ is $\mathcal{C}_\Sigma$-preserving.

f is $\mathcal{O}_\Sigma\mathcal{O}^L_{\mathcal{O}_\Sigma}$-reflecting iff it is $\mathcal{C}_\Sigma\mathcal{C}^U_{\mathcal{C}_\Sigma}$-reflecting.

Hence, $\widetilde{pre}_f$ is $\mathcal{C}_\Sigma$-preserving iff f is $\mathcal{C}_\Sigma\mathcal{C}^U_{\mathcal{C}_\Sigma}$-reflecting iff f is upper semicontinuous for $\mathcal{C}_\Sigma$.

Proof. By Propositions 5 and 17. □

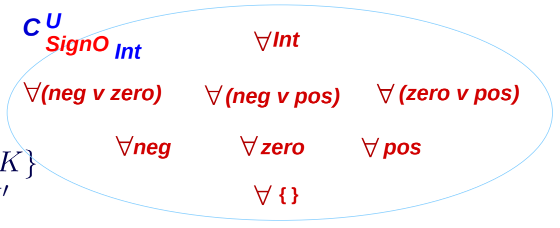
The corollary tells us $\widetilde{pre}_f$ lies in $\mathcal{C}_\Sigma$'s logic when $f : \Sigma \to \mathcal{P}(\Sigma)$ is upper semicontinuous. But what if f is not? Then we must underapproximate $\widetilde{pre}_f$ by some function of arity, $\mathcal{C}_\Sigma \to \mathcal{C}_\Sigma$. But we have no interior map to aid us, only a closure map.

The classic approach is to overapproximate f by some $f^\sharp : \mathcal{C}_\Sigma \to \mathcal{C}^U_{\mathcal{C}_\Sigma}$, from which we define a $\mathcal{C}_\Sigma$-preserving $\widetilde{pre}_{f^\sharp}$. To do this, we need some insight about $f^\sharp$'s codomain: Each $M \in \mathcal{C}^U_{\mathcal{C}_\Sigma}$ is a set of sets formed as $M = \bigcap_{i \in I}\{\forall K_i \mid K_i \in \mathcal{C}_\Sigma\}$. Read property M as "$\forall K_1 \wedge \forall K_2 \wedge \cdots \wedge \forall K_i \wedge \cdots$" — M's members are sets covered by property K_1 and covered by property K_2 and ... covered by property K_i and so on. This forces $f^\sharp$ to have this format, for all arguments $K_0 \in \mathcal{C}_\Sigma$:

$$f^\sharp(K_0) = \forall K_1 \wedge \forall K_2 \wedge \cdots \wedge \forall K_i \wedge \cdots$$

By pointwise reasoning, the M defined above equals $\forall \bigcap\{K_i \mid K_i \in \mathcal{C}_\Sigma\}$, read as "$\forall(K_1 \wedge K_2 \wedge \cdots \wedge K_i \wedge \cdots)$." But $\bigcap\{K_i \mid K_i \in \mathcal{C}_\Sigma\} \in \mathcal{C}_\Sigma$, meaning that $f^\sharp$ reverts to this more benign format:

Let $zero = \{0\}$
$neg = \{i \mid i < 0\}$
$pos = \{i \mid i > 0\}$
Let $\forall K = \{S \subseteq Int \mid S \subseteq K\}$
Let $K \vee K'$ denote $K \cup K'$



$sqrt : Int \to \mathcal{P}(Int)$
$sqrt(0) = \{0\}$
$sqrt(1) = \{-1, 1\} = sqrt(2) = sqrt(3)$
$sqrt(4) = \{-2, 2\} = sqrt(5)$, etc.
$sqrt(-1) = \emptyset$, etc.

$\widetilde{pre}_{sqrt} : \mathcal{P}(Int) \to \mathcal{P}(Int)$
$\widetilde{pre}_{sqrt}\{0, 1\} = \{0\}$
$\widetilde{pre}_{sqrt}\{-1, 0, 1\} = \{0, 1, 2, 3\}$
$\widetilde{pre}_{sqrt}\{i \mid i \neq 0\} = \{i \mid i \neq 0\}$, etc.

$sqrt^\sharp : \mathcal{C}^U_{Sign\mathcal{O}_{Int}} \to \mathcal{C}^U_{Sign\mathcal{O}_{Int}}$
$sqrt^\sharp(zero) = \forall zero$
$sqrt^\sharp(pos) = \forall(neg \vee pos)$
$sqrt^\sharp(neg) = \forall\emptyset$
$sqrt^\sharp(Int) = \forall Int$
$sqrt^\sharp(neg \vee pos) = \forall(neg \vee pos)$
$sqrt^\sharp(zero \vee neg) = \forall zero$
$sqrt^\sharp(zero \vee pos) = \forall Int$, etc.

$\widetilde{pre}_{sqrt^\sharp} : Sign\mathcal{O}_{Int} \to Sign\mathcal{O}_{Int}$
$\widetilde{pre}_{sqrt^\sharp}(neg \vee pos) = neg \vee pos$
$\widetilde{pre}_{sqrt^\sharp}(pos) = \emptyset$
$\widetilde{pre}_{sqrt^\sharp}(zero \vee neg) = zero$
$\widetilde{pre}_{sqrt^\sharp}(Int) = Int$, etc.

Figure 14: $sqrt$, upper topology on $Sign\mathcal{O}_{Int}$, and $sqrt^\sharp$

$$f^\sharp(K) = \forall K'$$

where $K, K' \in \mathcal{C}_\Sigma$. The quantifier reminds us that f's answer is a *set* of Σ-values, covered by K'. In temporal logic, the quantifier is written as $\Box$. That is, because $f^\sharp$ overapproximates f and $f^\sharp(K) = \forall K'$, we have that $K \models [f]K'$ is a sound assertion in temporal logic, that is, $K \subseteq \widetilde{pre}_f(K') = f^{-1}(\forall K)$. This connects the topology, $\mathcal{C}_\Sigma$, to the temporal logic.

Say we overapproximate $f : \Sigma \to \mathcal{P}(\Sigma)$ as expected by $f^\sharp(K) = \rho_U(f[K])$, where ρ_U is the closure operation for $\mathcal{C}^U_{\mathcal{C}_\Sigma}$: $\rho_U(T) = \bigcap\{\forall K \mid T \subseteq \forall K, K \in \mathcal{C}_\Sigma\}$. (That is, $\rho_U(T)$ computes the conjunction of all properties K that cover all the sets in T.) Next, we desire a sound $\widetilde{pre}_{f^\sharp}$ so that $\widetilde{pre}_{f^\sharp}(K) \subseteq \widetilde{pre}_f(K) = f^{-1}(\forall K)$, for all $K \in \mathcal{C}_\Sigma$. We work from Equation ($\star$) in Section 10; $f^\sharp$'s inverse image is

$$f^{\sharp -}_{\mathcal{C}_\Sigma}(K) = \{K' \in \mathcal{C}_\Sigma \mid f^\sharp(K') \subseteq \forall K\}$$

We want $\widetilde{pre}_{f^\sharp}(K) = \cup f^{\sharp -}(K)$, and if $\mathcal{C}_\Sigma$ is also closed under unions, we have what we want. If not, then we repeat the development in Section 10:

build the disjunctive completion of $\mathcal{C}_\Sigma$ (close it under unions), $\mathcal{CO}_\Sigma$; redefine $f^\sharp : \mathcal{CO}_\Sigma \to \mathcal{C}^U_{\mathcal{CO}_\Sigma}$; and define $\widetilde{pre}_{f^\sharp} : \mathcal{CO}_\Sigma \to \mathcal{CO}_\Sigma$ as $\widetilde{pre}_{f^\sharp}(K) = \cup {f^\sharp}^{-}_{\mathcal{CO}_\Sigma}(K)$.

Figure 14 displays an integer square-root function, $sqrt : Int \to \mathcal{P}(Int)$. The disjunctive completion of $Sign_{Int}$ produces the topology, $Sign\mathcal{O}_{Int}$, in Figure 11, from which we generate $\mathcal{C}^U_{Sign\mathcal{O}_{Int}}$, illustrated in Figure 14.

There is a useful, dual development of everything seen so far in this section. Starting again with Σ and its property family, $\mathcal{F}_\Sigma$, define the property family for $\mathcal{P}(\Sigma)$, namely, $\mathcal{O}^U_{\mathcal{F}_\Sigma} \subseteq \mathcal{P}(\mathcal{P}(\Sigma))$, as the open family generated by taking all unions of the base, $\mathcal{B}^U_{\mathcal{F}_\Sigma} = \{\forall U \mid U \in \mathcal{F}_\Sigma\}$, where $\forall U = \{S \subseteq \Sigma \mid S \subseteq U\}$. This is the *upper topology based on* $\mathcal{F}_\Sigma$, used by Smyth to characterize the upper powerdomain of denotational semantics. (Recall, for all $U \in \mathcal{F}_\Sigma$, that $f^{-1}(\forall U) = \widetilde{pre}_f(U)$.)

Proposition 19. *[8] Let $\mathcal{O}_\Sigma \subseteq \Sigma$ be an open family. $f : \Sigma \to \mathcal{P}(\Sigma)$ is upper semicontinuous for $\mathcal{O}_\Sigma$ iff f is $\mathcal{O}_\Sigma\mathcal{O}^U_{\mathcal{O}_\Sigma}$-reflecting.*

When f is not upper semicontinuous, we may use $\iota \circ \widetilde{pre}_f : \mathcal{O}_\Sigma \to \mathcal{O}_\Sigma$, where ι is $\mathcal{O}_\Sigma$'s interior operator, to underapproximate $\widetilde{pre}_f$ within the logic, $\mathcal{O}_\Sigma$. This is an elegant alternative to the tedious formulation of $f^\sharp$ and $\widetilde{pre}_{f^\sharp}$ presented in the preceding paragraphs.

The dual of Proposition 19 goes as follows: $\mathcal{C}^L_{\mathcal{C}_\Sigma} = {\sim}\mathcal{O}^U_{\mathcal{O}_\Sigma}$, whose members are all intersections of sets from the (co)base, $\mathcal{B}^L_{\mathcal{C}_\Sigma} = \{\exists K \mid K \in \mathcal{C}_\Sigma\}$, where $\exists K = \{S \subseteq \Sigma \mid S \cap K \neq \emptyset\}$. For all $K \in \mathcal{C}_\Sigma$, $f^{-1}(\exists K) = pre_f(K)$.

Corollary 20. *$\widetilde{pre}_f$ is $\mathcal{O}_\Sigma$-preserving iff pre_f is $\mathcal{C}_\Sigma$-preserving.*

f is $\mathcal{O}_\Sigma\mathcal{O}^U_{\mathcal{O}_\Sigma}$-reflecting iff it is $\mathcal{C}_\Sigma\mathcal{C}^L_{\mathcal{C}_\Sigma}$-reflecting.

Hence, pre_f is $\mathcal{C}_\Sigma$-preserving iff f is $\mathcal{C}_\Sigma\mathcal{C}^L_{\mathcal{C}_\Sigma}$-reflecting iff f is lower semicontinuous for $\mathcal{C}_\Sigma$.

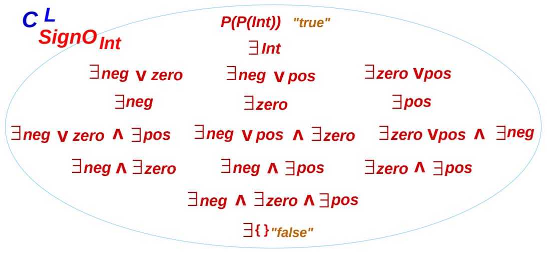
Say that $f : \Sigma \to \mathcal{P}(\Sigma)$ is not lower semicontinuous. When we approximate it by $f^\flat : \mathcal{C}_\Sigma \to \mathcal{C}^L_{\mathcal{C}_\Sigma}$, what is the result? What is $pre_{f^\flat}$? The answer characterizes significant research on underapproximation in abstract model checking [39, 40, 41].

Each $M \in \mathcal{C}^L_{\mathcal{C}_\Sigma}$ is a set of sets of form $M = \bigcap_{i \in I}\{\exists K_i \mid K_i \in \mathcal{C}_\Sigma\}$. Read M as "$\exists K_1 \wedge \exists K_2 \wedge \cdots \wedge \exists K_i \wedge \cdots$" — each of M's members is a set that meets (*witnesses*) K_1 and K_2 and ... K_i and so on. This forces $f^\flat$ to have this format, for all arguments $K_0 \in \mathcal{C}_\Sigma$:

$$f^\flat(K_0) = \exists K_1 \wedge \exists K_2 \wedge \cdots \wedge \exists K_i \wedge \cdots$$

Let $K \wedge K'$ denote $K \cap K'$

$pre_{sqrt} : \mathcal{P}(Int) \rightarrow \mathcal{P}(Int)$
$pre_{sqrt}\{0,1\} = \{0,1,2,3\}$
$pre_{sqrt} Int = \{i \mid i \geq 0\}$
$pre_{sqrt}\{i \mid i < 0\} = \{i \mid i > 0\}$
etc.



$sqrt^{\flat-1} : Sign_{Int} \rightarrow \mathcal{C}^L_{Sign\mathcal{O}_{Int}}$	$pre_{sqrt^{\flat-1}} : Sign\mathcal{O}_{Int} \rightarrow Sign\mathcal{O}_{Int}$
$sqrt^{\flat-1}(pos) = \exists neg \wedge \exists pos$	$pre_{sqrt^{\flat-1}}(pos) = pos$
$sqrt^{\flat-1}(zero) = \exists zero$	$pre_{sqrt^{\flat-1}}(neg) = pos$
$sqrt^{\flat-1}(Int) = true$	$pre_{sqrt^{\flat-1}}(zero) = zero$
$sqrt^{\flat-1}(neg) = true$	$pre_{sqrt^{\flat-1}}(Int) = zero \vee pos$
$sqrt^{\flat-1}(pos \vee zero) = \exists Int$	$pre_{sqrt^{\flat-1}}(\emptyset) = \emptyset$
$sqrt^{\flat-1}(pos \vee neg) = true$	$pre_{sqrt^{\flat-1}}(zero \vee neg) = zero \vee pos$, etc.

Figure 15: Lower topology on $Sign\mathcal{O}_{Int}$ and $sqrt^{\flat-1}$

The quantifiers remind us that f's answer is a set of Σ-values, witnessing (meeting) each of the K_i's. In temporal logic, the quantifier is written as $\Diamond$, and one may write $K_0 \models \langle f \rangle K_i$, for each such K_i.[9]

We approximate $f : \Sigma \rightarrow \mathcal{P}(\Sigma)$ by $f^\flat(K) = \rho_L(f[K])$, where ρ_L is the closure operation for $\mathcal{C}^L_{\mathcal{C}_\Sigma}$: $\rho_L(T) = \bigcap\{\exists K \mid T \subseteq \exists K,\ K \in \mathcal{C}_\Sigma\}$, that is, $\rho_L(T)$ collects all the properties, K, that are witnessed (met) by each of the sets in T. $f^\flat(K)$ is the strongest postcondition of $K \in \mathcal{C}_\Sigma$ in the logic associated with $\mathcal{C}^L_{\mathcal{C}_\Sigma}$, the "language of witnesses." Once again, we define $f^{\flat}{}^{-}_{\mathcal{C}_\Sigma}(K) = \{K' \in \mathcal{C}_\Sigma \mid f^\flat(K') \subseteq \exists K\}$ and $pre_{f^\flat}(K) = \cup f^{\flat}{}^{-}_{\mathcal{C}_\Sigma}(K)$. This is the definition used by Cleaveland et al. [39], Dams [40], and Schmidt [41] to prove that $pre_{f^\flat}$ computes weakest preconditions for f within the logics for $\mathcal{C}_\Sigma$ and $\mathcal{C}^L_{\mathcal{C}_\Sigma}$. When $pre_{f^\flat}$'s image does not fall within $\mathcal{C}_\Sigma$ — see $pre_{sqrt^{\flat-1}}(Int)$ in Figure 15, for example — disjunctive completion of $\mathcal{C}_\Sigma$ to a topology again saves the day.

[9]Larsen and Xinxin [42] and Shoham and Grumberg [43] have noted that the nonreducible structure of $\exists K_1 \wedge \exists K_2 \wedge \cdots \wedge \exists K_i \wedge \cdots$ is a source of precison loss in abstract-model checking and have proposed useful alternatives.

17. Conclusion

Abstract interpretation and denotational semantics share foundations and applications, and the interaction between the two areas is intricate. We have shown how the inverse-limit construction and its associated Scott-topology give new insights into the intricacies of abstract program analysis. In particular, the application of topology to abstract interpretation has a promising future.

Acknowledgement: Robert Tennent's depth of insight and clarity of presentation have been a continuing source of inspiration, and this paper is dedicated to him on the occasion of his 65th birthday. Flash Sheridan is thanked for his careful reading of an earlier draft.

References

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

[2] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Proc. 6th ACM Symp. on Principles of Programming Languages, ACM Press, 1979, pp. 269–282.

[3] N. Jones, F. Nielson, Abstract interpretation: a semantics-based tool for program analysis, in: S. Abramsky, D. Gabbay, T. Maibaum (Eds.), Handbook of Logic in Computer Science, Vol. 4, Oxford Univ. Press, 1995, pp. 527–636.

[4] P. Cousot, Semantic foundations of program analysis, in: S. Muchnick, N. Jones (Eds.), Program Flow Analysis, Prentice Hall, 1981, pp. 303–342.

[5] F. Nielson, H. Nielson, C. Hankin, Principles of Program Analysis, Springer Verlag, 1999.

[6] P. Cousot, N. Halbwachs, Automatic discovery of linear restraints among variables of a program, in: Proc. 5th ACM Symp. on Principles of Programming Languages, ACM Press, 1978, pp. 84–96.

[7] S. Graf, H. Saidi, Construction of abstract state graphs with pvs, in: Proc. Conf. Computer Aided Verification, Springer LNCS 1254, 1997, pp. 72–83.

[8] M. Smyth, Powerdomains and predicate transformers: a topological view, in: Proc. ICALP'83, LNCS 154, Springer, 1983, pp. 662–675.

[9] R. Giacobazzi, F. Ranzato, F. Scozzari, Making abstract interpretations complete, J. ACM 47 (2000) 361–416.

[10] R. Giacobazzi, E. Quintarelli, Incompleteness, counterexamples, and refinements in abstract model checking, in: Static Analysis Symposium, LNCS 2126, Springer Verlag, 2001, pp. 356–373.

[11] D. Schmidt, Comparing completeness properties of static analyses and their logics, in: Asian Symp. Prog. Lang. Systems (APLAS'06), LNCS 4279, Springer Verlag, 2006, pp. 183–199.

[12] C. Gunter, Semantics of Programming Languages, MIT Press, Cambridge, MA, 1992.

[13] G. Plotkin, Domains, lecture notes, Univ. Pisa/Edinburgh (1983).

[14] P. Cousot, R. Cousot, Higher-order abstract interpretation, in: Proceedings IEEE Int. Conf. Computer Lang., 1994.

[15] F. Ranzato, F. Tapparo, Strong preservation as completeness in abstract interpretation, in: Proc. European Symp. Programming, LNCS 2986, Springer Verlag, 2004, pp. 18–32.

[16] A. Miné, The octagon abstract domain, J. Higher-Order and Symbolic Computation 19 (2006) 31–100.

[17] T. Ball, A. Podeksi, S. Rajamani, Boolean and cartesian abstraction for model checking C programs, J. Software Tools for Technology Transfer 5 (2003) 49–58.

[18] N. Jones, S. Muchnick, Flow analysis and optimization of LISP-like structures, in: Proc. 6th. ACM Symp. Principles of Programming Languages, 1979, pp. 244–256.

[19] G. Gierz, K. Hofmann, K. Keimel, J. Lawson, M. Mislove, D. Scott, Continuous Lattices and Domains, Cambridge Univ. Press, 2003.

[20] J. Reynolds, Notes on a lattice-theoretic approach to the theory of computation, Technical report, Computer Science, Syracuse University (1972).

[21] P. Johnstone, Stone Spaces, Cambridge University Press, 1986.

[22] S. Abramsky, Domain theory in logical form, Ann.Pure Appl.Logic 51 (1991) 1–77.

[23] V. Donzeau-Gouge, Denotational definition of properties of program's computations, in: S. Muchnick, N. Jones (Eds.), Program Flow Analysis: Theory and Applications, Prentice-Hall, 1981.

[24] F. Nielson, A denotational framework for data flow analysis, Acta Informatica 18 (1982) 265–287.

[25] F. Nielson, Program transformations in a denotational setting, ACM Trans. Prog. Languages and Systems 7 (1985) 359–379.

[26] F. Nielson, H. R. Nielson, Two-Level Functional Languages, Cambridge University Press, 1992.

[27] N. Jones, A. Mycroft, Data flow analysis of applicative programs using minimal function graphs, in: Proc. 13th ACM Symp. on Principles of Prog. Languages, 1986, pp. 296–306.

[28] P. Cousot, R. Cousot, Static determination of dynamic properties of recursive procedures, in: E. Neuhold (Ed.), Formal Description of Programming Concepts, North-Holland, 1978, pp. 238–277.

[29] S. Hunt, Frontiers and open sets in abstract intepretation, in: Proc. ACM Symp. Functional Prog. and Comp. Architecture, 1989, pp. 194–216.

[30] P. Dybjer, Inverse image analysis generalises strictness analysis, Information and Computation 90 (1991) 194–216.

[31] T. Jensen, Abstract interpretation in logical form, Ph.D. thesis, Imperial College, London (1992).

[32] N. Benton, Strictness logic and polymorphic invariance, in: Proc. Logical Found. Comp. Sci, 1992, pp. 33–44.

[33] S. Willard, General Topology, Dover Publications, 2004.

[34] M. Sagiv, T. Reps, R. Wilhelm, Parametric shape analysis via 3-valued logic, ACM TOPLAS 24 (2002) 217–298.

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

[36] C. Gunter, D. Scott, Semantic domains, in: J. vanLeeuwen (Ed.), Handbook of Theoretical Computer Science: Volume B, Elsevier, 1990, pp. 633–674.

[37] R. Heckmann, Power domain constructions, Ph.D. thesis, Univ. Saarbrücken (1990).

[38] M. Smyth, Powerdomains, Journal of Computer and System Sciences 16 (1978) 23–36.

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

[40] D. Dams, R. Gerth, O. Grumberg, Abstract interpretation of reactive systems, ACM Trans. Prog. Lang. Systems 19 (1997) 253–291.

[41] D. Schmidt, Underapproximating predicate transformers, in: Proc. Symp. Static Analysis (SAS'06), LNCS 4134, Springer Verlag, 2006, pp. 127–143.

[42] K. Larsen, L. Xinxin, Equation solving using modal transition systems, in: Proc. Logic in Computer Science, IEEE Press, 1990, pp. 108–117.

[43] S. Shoham, O. Grumberg, Three-valued abstraction: More precision at less cost, Information and Computation 206 (1998) 1313–1333.