# Foundations of Abstract Interpretation

**David Schmidt**

Kansas State University

www.cis.ksu.edu/~schmidt



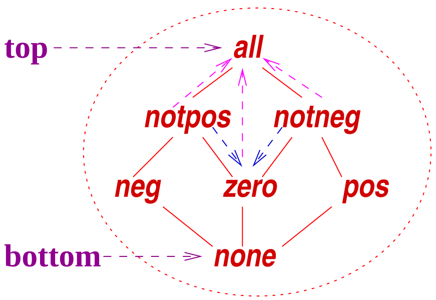
## Outline

1. Lattices and continuous functions
2. Galois connections, closures, and Moore families
3. Soundness and completeness of operations on abstract data
4. Soundness and completeness of execution trace computation



## Data sets are *complete lattices*

A complete lattice is a partially ordered set, with unique minimal and maximal elements, and with greatest-lower-bound and least-upper-bound operations:




$\sqcap\{\mathit{notpos}, \mathit{notneg}\} = \mathit{zero}$

$\sqcup\{\mathit{zero}, \mathit{notpos}, \mathit{notneg}\} = \mathit{all}$

$\sqcap\{\,\} = \mathit{all}$

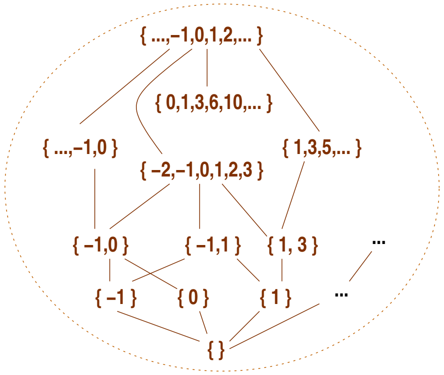
$\sqcup\{\,\} = \mathit{none}$

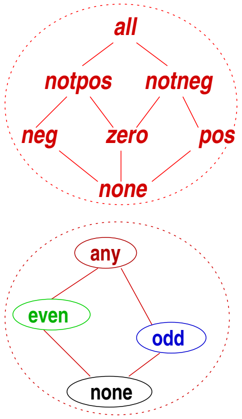


Here is a more precise definition: A *complete lattice*, $\mathcal{L} = \langle D, \sqsubseteq, \perp, \top, \sqcup, \sqcap \rangle$, consists of

- a set, $D$, and a partial ordering, $\sqsubseteq$, on $D$
- a smallest element, $\perp$ (such that $\perp \sqsubseteq d$, for all $d \in D$) and a greatest element, $\top$ (such that $d \sqsubseteq \top$, for all $d \in D$)
- a *least upper bound* operation, $\sqcup$, such that, for all $S \subseteq D$, $d \sqsubseteq \sqcup S$, for all $d \in S$, and for all other upper bounds, $c \in D$, such that $d \sqsubseteq c$, for all $d \in S$, we have that $\sqcup S \sqsubseteq c$
- a *greatest lower bound* operation, $\sqcap$, defined dually to the above: $\sqcap S \sqsubseteq d$, for all $d \in S$, and when $c \sqsubseteq d$, for all $d \in S$, we have that $c \sqsubseteq \sqcap S$

The first example is the complete lattice, $\langle \wp(\mathrm{Int}), \subseteq, \{\,\}, \mathrm{Int}, \bigcup, \bigcap \rangle$; the next two are abstractions of it:






## Monotonic and chain-continuous functions

Given complete lattices, $\mathcal{A}$ and $\mathcal{B}$, we say that a function, $f : A \to B$, is *monotonic* iff

$$\text{for all } a, a' \in A, a \sqsubseteq_A a' \text{implies } f(a) \sqsubseteq_B f(a')$$

A monotonic function preserves the "precision of information" in its argument.

Say that we have an $\omega$-*chain*, $a_0 \sqsubseteq_A a_1 \sqsubseteq_A \ldots \sqsubseteq_A a_i \sqsubseteq_A a_{i+1} \sqsubseteq_A \ldots$.

A function, $f : A \to B$, is $\omega$-*continuous* iff

$$\bigsqcup_{i \geq 0} f(a_i) = f(\bigsqcup_{i \geq 0} a_i)$$

An $\omega$-continuous function preserves the "limit of information" in a chain. Conventional computation employs monotonic and $\omega$-continuous functions, so it is no restriction to use only them.

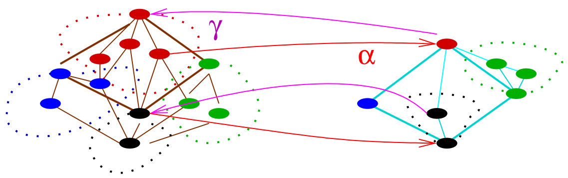


## Galois connections

Given a complete lattice of "concrete" (execution) data, $\mathcal{C}$, and a simpler complete lattice of "abstract" data, $\mathcal{A}$, we relate the two by $\alpha : C \to A$ that will act like a *homomorphism* when we study the operations on $C$.

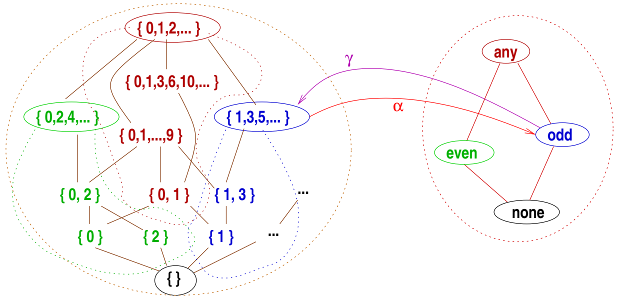
It will be useful that $\alpha$ have an "inverse,", $\gamma$:

**Definition:** For complete lattices $\mathcal{C}$ and $\mathcal{A}$, and monotonic functions, $\alpha : C \to A$, $\gamma : A \to C$, the pair, $\langle \alpha, \gamma \rangle$ form a *Galois connection*, written $\mathcal{C}\langle \alpha, \gamma \rangle \mathcal{A}$, iff $c \sqsubseteq_C \gamma \circ \alpha(c)$ and $\alpha \circ \gamma(a) \sqsubseteq_A a$.






The maps $\alpha$ and $\gamma$ are inverse maps on each other's image:



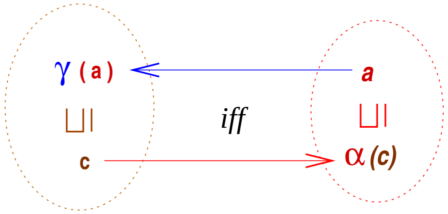

That is, for all $c \in \gamma[A], c = \gamma \circ \alpha(c)$; for all $a \in \alpha[C], a = \alpha \circ \gamma(a)$.

$\alpha$ is $\omega$-continuous (and even preserves $\sqcup$ for arbitrary sets in C); $\gamma$ preserves $\sqcap$ for arbitrary sets in $A$. Each map uniquely defines the other:

$$\gamma(a) = \sqcup\{c \mid \alpha(c) \sqsubseteq_A a\} \quad \text{and} \quad \alpha(c) = \sqcap\{a \mid c \sqsubseteq_C \gamma(a)\}$$

The previous fact suggests this alternative characterization of Galois connection:

**Proposition:** For complete lattices $\mathcal{C}$ and $\mathcal{A}$, the pair, $\langle \alpha : C \to A, \gamma : A \to C \rangle$, is a Galois connection when, for all $c \in C$ and $a \in A$, $\ c \sqsubseteq_C \gamma(a)$ iff $\ \alpha(c) \sqsubseteq_A a$.




From this definition, we can prove that both $\alpha$ and $\gamma$ are monotonic, that $c \sqsubseteq_C \gamma \circ \alpha(c)$, and that $\alpha \circ \gamma(a) \sqsubseteq_A a$.

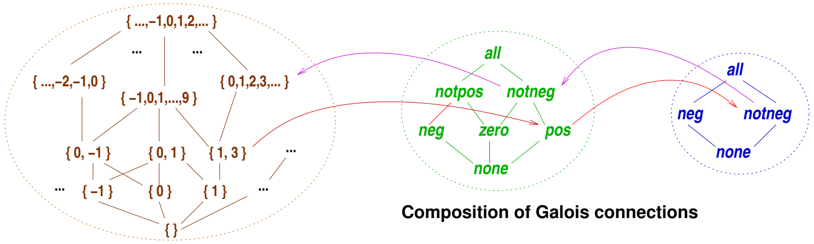


Galois connections are closed under composition, product, and so on:

**If** $\mathcal{C}\langle\alpha,\gamma\rangle\mathcal{D}$ and $\mathcal{D}\langle\alpha',\gamma'\rangle\mathcal{E}$ are Galois connections, **then** so is $\mathcal{C}\langle\alpha' \circ \alpha, \gamma \circ \gamma'\rangle\mathcal{E}$

**If** $\mathcal{C}_i\langle\alpha_i,\gamma_i\rangle\mathcal{D}_i$ is a Galois connection, for all $i \in I$, **then** so is $\Pi_{i \in I}\mathcal{C}_i\langle\Pi_{i \in I}\alpha_i, \Pi_{i \in I}\gamma_i\rangle\Pi_{i \in I}\mathcal{D}_i$.

**If** $\mathcal{C}\langle\alpha_C,\gamma_C\rangle\mathcal{C}'$ and $\mathcal{D}\langle\alpha_D,\gamma_D\rangle\mathcal{D}'$ are Galois connections, **then** so is $\mathcal{C} \to \mathcal{D}\langle(\lambda f.\alpha_D \circ f \circ \gamma_C), (\lambda f'.\gamma_D \circ f' \circ \alpha_C)\rangle\mathcal{C}' \to \mathcal{D}'$.




**Composition of Galois connections**



Why do we require the elaborate structure of a Galois connection?

1. If we are certain about the precise definition of $\gamma : A \to C$, we can mechanically synthesize the its adjoint, $\alpha(c) = \sqcap\{a | c \sqsubseteq_C \gamma(a)\}$.

   (Or, dually, if we are certain about $\alpha$, we can synthesize $\gamma$ as $\gamma(a) = \sqcup\{c | \alpha(c) \sqsubseteq_A a\}$. )

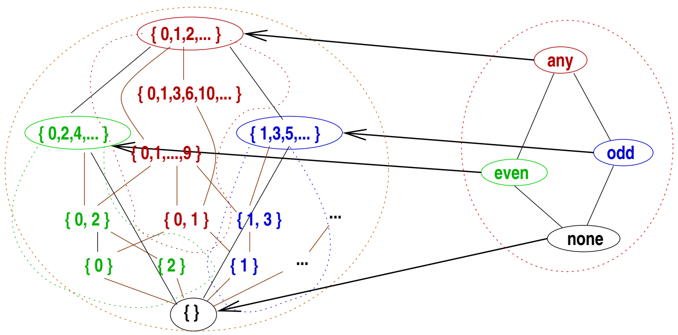
2. We obtain many mathematical properties about $\alpha$, expressed in terms of its adjoint, $\gamma$ (and vice versa).

3. Since we intend to use $\alpha : C \to A$ as a "homomorphism" from $C$ to $A$, we can use $\alpha$ and its adjoint $\gamma$ to synthesize abstract operations: For each $f : C \to C$, we can synthesize $f^\# : A \to A$, such that $\alpha$ is a "homomorphism" with respect to $f$ and $f^\#$. (We will see that $f^\# = \alpha \circ f \circ \gamma$.)



# Closure maps

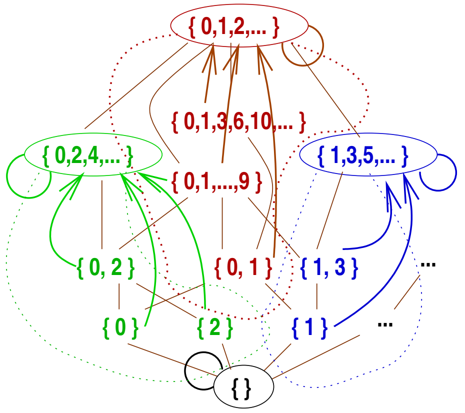
For $\mathcal{C}\langle\alpha,\gamma\rangle\mathcal{A}$, it is common that $\alpha$ is onto. This means $\mathcal{A}$ embeds into $\mathcal{C}$ as a sublattice:




$A$'s elements are mere "tokens" that name distinguished sets in $C$.

**Definition:** $\rho : \mathcal{C} \to \mathcal{C}$ is a *closure map* if it is **(i)***monotonic*; **(ii)***extensive:* $c \sqsubseteq_C \rho(c)$, for all $c \in C$; **(iii)***idempotent:* $\rho \circ \rho = \rho$.

*A closure map defines the embedding:*




$\rho\{0,2\} = \{0,2,4,...\}$

$\rho\{0,2,4,...\} = \{0,2,4,...\}$

$\rho\{0,1,...,9\} = \{0,1,2,...\}$

Every Galois connection, $\mathcal{C}\langle\alpha,\gamma\rangle\mathcal{A}$, defines a closure map, $\gamma \circ \alpha$.

Every closure map, $\rho : C \to C$, defines the Galois connection, $\mathcal{C}\langle\rho, \mathrm{id}\rangle\rho[C]$.



## Moore families

Given $\mathcal{C}$, can we define a closure map on it by choosing some elements of $\mathcal{C}$? The answer is *yes*, if the elements of $C$ we select are closed under greatest-lower-bounds:

**Definition:** $M \subseteq C$ is a *Moore family* iff for all $S \subseteq M$, $(\sqcap S) \in M$.

We can define a closure map as $\rho(c) = \sqcap\{c' \in M \mid c \sqsubseteq_C c'\}$.

For a closure map, $\rho : C \to C$, its image, $\rho[C]$, is a Moore family.

Given $C$, we can define an abstract interpretation by selecting some $M \subseteq C$ that is a Moore family!



## Closed binary relations

Often a Galois connection uses a powerset for its concrete domain, that is, $\wp(\mathcal{D})\langle\alpha,\gamma\rangle\mathcal{A}$. This format yields a simple characterization:

Given unordered set $D$ and complete lattice $A$, it is natural to relate the elements in $D$ to those in $A$ by a binary relation, $\mathcal{R} \subseteq D \times A$, such that $(d, a) \in \mathcal{R}$ means "$d$ *has property* $a$." We write this as $d\ \mathcal{R}\ a$ or as $d \models_{\mathcal{R}} a$.

**Example:** $D = \mathrm{Int}$, and $A = \{\mathrm{none}, \mathrm{neg}, \mathrm{pos}, \mathrm{zero}, \mathrm{nonneg}, \mathrm{nonpos}, \mathrm{any}\}$.

Then, $2\ \mathcal{R}\ \mathrm{nonneg}$, $2\ \mathcal{R}\ \mathrm{pos}$, and $2\ \mathcal{R}\ \mathrm{any}$. (Or we write, $2 \models_{\mathcal{R}} \mathrm{nonneg}$, $2 \models_{\mathcal{R}} \mathrm{pos}$, and $2 \models_{\mathcal{R}} \mathrm{any}$.)

We immediately define the function, $\gamma : A \to \wp(D)$, as

$$\gamma(a) = \{d \in D \mid d\ \mathcal{R}\ a\}$$

For example, $\gamma(\mathrm{nonneg}) = \{0, 1, 2, ...\}$.



We can check if $\gamma$ is the upper adjoint of a Galois connection, say, by showing that $\gamma[A]$ defines a Moore family. But we can check for this directly upon $\mathcal{R}$:

**Proposition:** $\mathcal{R} \subseteq D \times A$ defines a Galois connection between $\wp(D)$ and $A$ iff **(i)** $\mathcal{R}$ is *U-closed*: $c\ \mathcal{R}\ a$ and $a \sqsubseteq_A a'$ imply $c\ \mathcal{R}\ a'$; **(ii)** $\mathcal{R}$ is *G-closed*: $c\ \mathcal{R}\ \sqcap\{a \mid c\ \mathcal{R}\ a\}$.

If $\mathcal{R}$ defines a Galois connection, then we have this crucial property:

- for all $a \in A$ and $C \in \wp(D)$, $C \subseteq \gamma(a)$ iff $\alpha(C) \sqsubseteq_A a$ iff ($c\ \mathcal{R}\ a$, for all $c \in C$).

This is of course the definition of a Galois connection, and in this sense, $\mathcal{R}$ "is" a Galois connection.

## A recipe for abstract-domain building

Given an unordered set, $D$, of concrete data values, we might ask, *"What are the properties about $D$ that I wish to calculate? Can I relate these properties, $a \in A$, to elements $d \in D$ via a UG-closed binary relation, $\mathcal{R}_D \subseteq D \times A$?"* Given a set, $A$, and relation, $\mathcal{R}_D \subseteq D \times A$,

1. Define $\gamma : A \to \wp(D)$ as $\gamma(a) = \{d \mid d\ \mathcal{R}_D\ a\}$.
2. Define this partial ordering on $A$: $a \sqsubseteq a'$ iff $\gamma(a) \subseteq \gamma(a')$. (If there are distinct $a, a' \in A$ such that $\gamma(a) = \gamma(a')$, then merge them.) This forces U-closure.
3. Ensure that $\gamma[A]$ is a Moore family by adding greatest-lower-bound elements to $A$ as needed. This forces G-closure.
4. Use the existing machinery to define the Galois connection between $\wp(D)$ and $A$.



## Example: Abstracting the Program State

The concrete storage vector is a product,

$$\mathrm{Store} = \Pi_{i \in \mathrm{Identifier}} \mathrm{Data}$$

and the concrete program state is a $\mathrm{ProgramPoint} \times \mathrm{Store}$ pair.

**Example:** $p_1, \langle x : 3, y : 4 \rangle$ is a program state.

Say that we have the relation, $\mathcal{R}_{\mathrm{Data}} \subseteq \mathrm{Data} \times \mathrm{AbsData}$, and we have the induced Galois connection, $\wp(\mathrm{Data})\langle \alpha_{\mathrm{Data}}, \gamma_{\mathrm{Data}} \rangle \mathrm{AbsData}$. Now, we can build Galois connections that abstract the store and the state.

A concrete store is related to an abstract store:

$$\langle x_i : v_i \rangle_{i \in \mathrm{Id}}\ \mathcal{R}_{\mathrm{Store}}\ \langle x_i : a_i \rangle_{i \in \mathrm{Id}}, \text{ iff, for all } i \in \mathrm{Id}, v_i\ \mathcal{R}_{\mathrm{Data}}\ a_i$$

**Example:** $\langle x : 3, y : 4 \rangle\ \mathcal{R}_{\mathrm{Store}}\ \langle x : \mathrm{any}, y : \mathrm{even} \rangle$.

This produces a Galois connection, $\wp(\mathrm{Store})\langle \alpha_{\mathrm{Store}}, \gamma_{\mathrm{Store}} \rangle \mathrm{AbsStore}$,



where $\mathrm{AbsStore} = \Pi_{i \in \mathrm{Identifier}} \mathrm{AbsData}$ and

$$\gamma \langle x_i : a_i \rangle_{i \in \mathrm{Id}} = \{ \langle x_i : v_i \rangle_{i \in \mathrm{Id}} \mid v_i \in \gamma_{\mathrm{Data}}(a_i), \text{ for all } i \in \mathrm{Id} \}$$

$$\alpha_{\mathrm{Store}}(S) = \langle \bigsqcup_{s \in S} \alpha(s(i)) \rangle_{i \in \mathrm{Id}}$$

For example,
$\gamma_{\mathrm{Store}} \langle x : \mathrm{even}, y : \mathrm{odd} \rangle = \{ \langle x : 0, y : 1 \rangle, \langle x : 0, y : 3 \rangle, \langle x : 2, y : 1 \rangle, \ldots \}$

A program point is abstracted to itself: $p\ \mathcal{R}_{PP}\ p$, suggesting that the abstract domain of program points might be merely $\mathrm{AbsPP} = \mathrm{ProgramPoint} \cup \{\bot, \top\}$. ($\top$ and $\bot$ are needed to make $\mathrm{AbsPP}$ a complete lattice.)

Finally, we can relate a concrete state to an abstract one:

$$p, s\ \mathcal{R}_{\mathrm{State}}\ p', \sigma \text{ iff } p\ \mathcal{R}_{PP}\ p' \text{ and } s\ \mathcal{R}_{\mathrm{Store}}\ \sigma$$

Hence, $\gamma_{\mathrm{State}}(p_i, \sigma) = \{ p_i, s \mid s \in \gamma_{\mathrm{Store}}(\sigma) \}$.



## Concrete and abstract operations

Now that we know how to model $c \in C$ by $\alpha(c) \in A$, we must model concrete computation steps, $f : C \to C$, by abstract computation steps, $f^{\#} : A \to A$.

**Example:** We have concrete domain, $\mathrm{Nat}$, and concrete operation, $\mathrm{succ} : \mathrm{Nat} \to \mathrm{Nat}$, defined as $\mathrm{succ}(n) = n + 1$.

We have abstract domain, $\mathrm{Parity}$, and abstract operation, $\mathrm{succ}^{\#} : \mathrm{Parity} \to \mathrm{Parity}$, defined as

$$\mathrm{succ}^{\#}(\mathrm{even}) = \mathrm{odd}, \quad \mathrm{succ}^{\#}(\mathrm{odd}) = \mathrm{even}$$
$$\mathrm{succ}^{\#}(\mathrm{any}) = \mathrm{any}, \quad \mathrm{succ}^{\#}(\mathrm{none}) = \mathrm{none}$$

$\mathrm{succ}^{\#}$ must be consistent (sound) with respect to $\mathrm{succ}$:
if $n\ \mathcal{R}_{\mathrm{Nat}}\ a$, then $\mathrm{succ}(n)\ \mathcal{R}_{\mathrm{Nat}}\ \mathrm{succ}^{\#}(a)$

where $\mathcal{R} \subseteq \mathrm{Nat} \times \mathrm{Parity}$ relates numbers to their parities (e.g., $2\ \mathcal{R}_{\mathrm{Nat}}\ \mathrm{even}$, $5\ \mathcal{R}_{\mathrm{Nat}}\ \mathrm{odd}$, etc.).

We want *soundness*: $n\ \mathcal{R}_{\mathrm{Nat}}\ a$ implies $\mathrm{succ}(n)\ \mathcal{R}_{\mathrm{Nat}}\ \mathrm{succ}^{\#}(a)$, for all $n \in \mathrm{Nat}$ and $a \in \mathrm{Parity}$.

Since we have the Galois connection, $\wp(\mathrm{Nat})\langle\alpha,\gamma\rangle\mathrm{Parity}$, we know that $\gamma(a) = \{n \mid n\ \mathcal{R}_{\mathrm{Nat}}\ a\}$.

So, soundness is stated equivalently as

$$\text{for all } a \in A, \text{ for all } n \in \gamma(a),\ \mathrm{succ}(n) \in \gamma(\mathrm{succ}^{\#}(a))$$

and this is equivalent to saying,

$$\text{for all } a \in A,\ \mathrm{succ}^{*}(\gamma(a)) \subseteq_{\mathrm{Nat}} \gamma(\mathrm{succ}^{\#}(a))$$
$$\text{that is,}$$
$$\text{for all } a \in A,\ (\mathrm{succ}^{*} \circ \gamma)(a) \subseteq_{\mathrm{Nat}} (\gamma \circ \mathrm{succ}^{\#})(a)$$

where $\mathrm{succ}^{*} : \wp(\mathrm{Nat}) \to \wp(\mathrm{Nat})$ is $\mathrm{succ}^{*}(S) = \{\mathrm{succ}(n) \mid n \in S\}$.

This is interesting, because it states a commutative, "semi-homomorphism" property....



**Definition:** For Galois connection, $\mathcal{C}\langle\alpha,\gamma\rangle\mathcal{A}$, and functions $f : C \to C$, $f^{\#} : A \to A$, $f^{\#}$ is a *sound approximation* of $f$ iff

$$(\alpha \circ f)(c) \sqsubseteq_A (f^{\#} \circ \alpha)(c),\ \text{for all } c \in C$$
$$\text{iff}$$
$$(f \circ \gamma)(a) \sqsubseteq_C (\gamma \circ f^{\#})(a),\ \text{for all } a \in A$$

This slightly abstract presentation exposes that $\alpha$ is a "semi-homomorphism" with respect to $f$ and $f^{\#}$:

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



**Example 1:** $n\ \mathcal{R}_{\mathrm{Nat}}\ a$ implies $\mathrm{succ}(n)\ \mathcal{R}_{\mathrm{Nat}}\ \mathrm{succ}^{\#}(a)$

Galois connection: $\wp(\mathrm{Nat})\langle\alpha,\gamma\rangle\mathrm{Parity}$

$\mathrm{succ}^{*} : \wp(\mathrm{Nat}) \to \wp(\mathrm{Nat})$

$\mathrm{succ}^{*}(S) = \{\mathrm{succ}(n) \mid n \in S\}$

where $\mathrm{succ}(n) = n + 1$

$\mathrm{succ}^{\#} : \mathrm{Parity} \to \mathrm{Parity}$

$\mathrm{succ}^{\#}(\mathrm{even}) = \mathrm{odd}, \quad \mathrm{succ}^{\#}(\mathrm{odd}) = \mathrm{even}$

$\mathrm{succ}^{\#}(\mathrm{any}) = \mathrm{any}, \quad \mathrm{succ}^{\#}(\mathrm{none}) = \mathrm{none}$

We have that $\alpha \circ \mathrm{succ}^{*} = \mathrm{succ}^{\#} \circ \alpha$:

$$\begin{tikzcd}
\{2,6\} \arrow[r, "\alpha"] \arrow[d, "\mathrm{succ}^{*}"'] & \mathrm{even} \arrow[d, "\mathrm{succ}^{\#}"] \\
\{3,7\} \arrow[r, "\alpha"] & \mathrm{odd} = \mathrm{odd}
\end{tikzcd}$$



**Example 2:** $n\ \mathcal{R}_{\mathrm{Nat}}\ a$ implies $\mathrm{div2}(n)\ \mathcal{R}_{\mathrm{Nat}}\ \mathrm{div2}^{\#}(a)$

Galois connection: $\wp(\mathrm{Nat})\langle\alpha,\gamma\rangle\mathrm{Parity}$

$\mathrm{div2}^{*} : \wp(\mathrm{Nat}) \to \wp(\mathrm{Nat})$

$\mathrm{div2}^{*}(S) = \{\mathrm{div2}(n) \mid n \in S\}$

where $\mathrm{div2}(2n+1) = \mathrm{div2}(2n) = n$

$\mathrm{div2}^{\#} : \mathrm{Parity} \to \mathrm{Parity}$

$\mathrm{div2}^{\#}(\mathrm{even}) = \mathrm{div2}^{\#}(\mathrm{odd}) = \mathrm{any}$

$\mathrm{div2}^{\#}(\mathrm{any}) = \mathrm{any}, \quad \mathrm{div2}^{\#}(\mathrm{none}) = \mathrm{none}$

We have that $\alpha \circ \mathrm{div2}^{*} \sqsubseteq_{\mathrm{Parity}} \mathrm{div2}^{\#} \circ \alpha$:

$$\begin{tikzcd}
\{6\} \arrow[r, "\alpha"] \arrow[d, "\mathrm{div2}^{*}"'] & \mathrm{even} \arrow[d, "\mathrm{div2}^{\#}"] \\
\{3\} \arrow[r, "\alpha"] & \mathrm{odd} \sqsubseteq \mathrm{any}
\end{tikzcd}$$

## Synthesizing $f^{\#}$ from $f$

The previous slides show how $\alpha$ acts as a "semi-homomorphism" between $f$ and $f^{\#}$.

Given the Galois connection, $\mathcal{C}\langle\alpha,\gamma\rangle\mathcal{A}$, and operation, $f : C \to C$, the most precise $f^{\#}_{best} : A \to A$ that is sound with respect to $f$ is

$$f^{\#}_{best} = \alpha \circ f \circ \gamma$$

**Proposition:** $f^{\#}$ is sound with respect to $f$ iff $f^{\#}_{best} \sqsubseteq_{A\to A} f^{\#}$.

(Note: $f \sqsubseteq_{A\to A} g$ iff for all $a \in A$, $f(a) \sqsubseteq_A g(a)$.)

Of course, $f^{\#}_{best}$ has a *mathematical* definition — not an algorithmic one — $f^{\#}_{best}$ might not be finitely computable !

**Parity example continued:**

$$\begin{aligned} succ^{\#}_{best}(even) &= \alpha \circ succ^{*}(\gamma\ even) \\ &= \alpha(succ^{*}\{2n \mid n \geq 0\}) \\ &= \alpha\{2n+1 \mid n \geq 0\} = odd \end{aligned}$$



**One more example:**

Given $\wp(Nat)\langle\alpha,\gamma\rangle Parity$ and $div2 : Nat \to Nat$, we have

$$\begin{aligned} &div2^{*} : \wp(Nat) \to \wp(Nat) \\ &div2^{*}(S) = div2[S] = \{div2(n) \mid n \in S\} \end{aligned}$$

Hence, $div2^{\#}_{best} = \alpha \circ div2^{*} \circ \gamma$. The operation loses precision: $\alpha(div2^{*}\{4\}) = \alpha\{2\} = even$, but

$$\begin{aligned} div2^{\#}_{best}(even) &= \alpha(div2^{*}(\gamma(even))) \\ &= \alpha(div2^{*}\{0,2,4,\ldots\}) \\ &= \alpha\{1,2,3,\ldots\} = any \end{aligned}$$

Nonetheless, this is the best we can do, given the crude structure of the abstract domain, **Parity**.



## Completeness

Given $\mathcal{C}\langle\alpha,\gamma\rangle\mathcal{A}$, we state soundness of $f^{\#}$ with respect to $f$ as $\alpha \circ f \sqsubseteq_{A\to A} f^{\#} \circ \alpha$ iff $f \circ \gamma \sqsubseteq_{C\to C} \gamma \circ f^{\#}$

**Definition:** $f^{\#}$ is *forwards ($\gamma$) complete* with respect to $f$ iff $f \circ \gamma =_{C\to C} \gamma \circ f^{\#}$

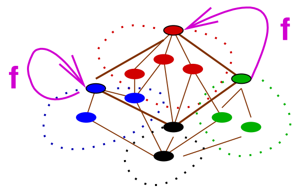
**Definition:** $f^{\#}$ is *backwards ($\alpha$) complete* with respect to $f$ iff $\alpha \circ f =_{A\to A} f^{\#} \circ \alpha$

The two completeness notions are not equivalent!

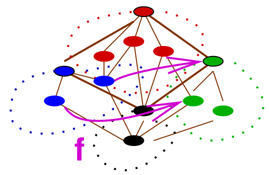
For an $f^{\#}$ to be (forwards or backwards) complete, it must equal $f^{\#}_{best} = \alpha \circ f \circ \gamma$. Indeed, the structure of the Galois connection and $f : C \to C$ determines whether $f^{\#}_{best}$ is complete.



**Forwards ($\gamma$) completeness:** $f^{\#}_{best}$ is forwards-complete iff $f$ maps image points of $\gamma$ to image points of $\gamma$ — $f(\gamma[A]) \subseteq \gamma[A]$.




**Backwards ($\alpha$) completeness:** $f^{\#}_{best}$ is backwards-complete iff $f$ maps all points in the same $\alpha$-equivalence class to points in the same $\alpha$-equivalence class — $\alpha(c) = \alpha(c')$ implies $\alpha(f(c)) = \alpha(f(c'))$.



## *Transfer functions* generate computation steps

Each program transition from program point $p_i$ to $p_j$ has an associated *transfer function*, $f_{ij} : C \to C$ (or $f^{\#}_{ij} : A \to A$), which describes the associated computation.

This defines a computation step of the form, $p_i, s \to p_j, f_{ij}(s)$.

**Example:** Assignment $p_0 : \mathtt{x} = \mathtt{x} + 1;\ p_1 : \cdots$ has the transfer function, $f_{01}\langle ...x : n...\rangle = \langle ...x : n + 1...\rangle$. For example, $p_0, \langle x : 3\rangle \to p_1, f_{01}\langle x : 3\rangle = p_1, \langle x : 4\rangle$.

For modelling multiple transitions in conditional/nondeterministic choice, we attach a transfer function to each possible transition.

**Example:** For

```
p0 : cases
    x ≤ y:   p1 : y = y - x;
    y ≤ x:   p2 : x = x - y;
end
```



For

```
p0 : cases
    x ≤ y:   p1 : y = y - x;
    y ≤ x:   p2 : x = x - y;
end
```

we have these functions:

$$f_{01}(s) = \begin{cases} s \text{ if } s.x \le s.y \\ \perp \text{ otherwise} \end{cases}$$

$$f_{02}(s) = \begin{cases} s \text{ if } s.y \le s.x \\ \perp \text{ otherwise} \end{cases}$$

For example, $p_0, \langle x : 5, y : 3\rangle \to p_1, \perp$, because $x \not\le y$, but $p_0, \langle x : 5, y : 3\rangle \to p_2, \langle x : 5, y : 3\rangle$, because $y \le x$. The transfer functions "filter" the data that arrives at a program point.

We ignore computation steps, $p, s \to p', \perp$, that produce "no data" ($\perp$).

An *execution trace* is a (possibly infinite) sequence, $p_0, s_0 \to p_1, s_1 \to \cdots \to p_j, s_j \to \cdots$, such that, for all $i \ge 0$:

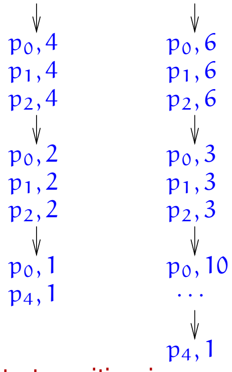
- $p_i, s_i \to p_{succ(i)}, f_{i,succ(i)}(s_i)$
- no $s_i$ equals $\perp$.



## Using the $f^{\#}$s to build sound, abstract trace trees

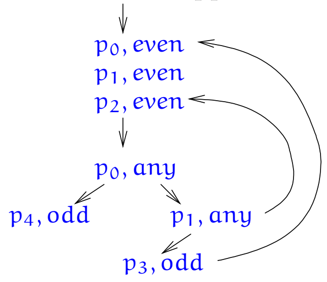
```
p0 : while (x != 1) {
  p1 : if Even(x)
    p2 : then  x = x div2;
    p3 : else  x = 3*x + 1;
  }
p4 : exit
```

Two concrete traces:




**Note:** $p_i, v$ abbreviates $p_i, \langle x : v\rangle$

Abstract overapproximating trace:




Each concrete transition is generated by an $f_{ij}$; each abstract transition is generated by the corresponding $f^{\#}_{ij}$.



Each concrete transition, $p_i, s \to p_j, f_{ij}(s)$, is reproduced by a corresponding abstract transition, $p_i, a \to p_j, f^{\#}_{ij}(a)$, where $s \in \gamma(a)$.

**For example,** $p_2 : \mathtt{x} = \mathtt{x\ div2}$ is interpreted *concretely* by $f_{20}(2n) = n = f_{20}(2n + 1)$ and is interpreted *abstractly* by $f^{\#}_{20}(even) = any = f^{\#}_{20}(odd) = f^{\#}_{20}(any)$.

The traces embedded in the abstract trace tree "cover" (*simulate*) the concrete traces, e.g., this concrete trace,

$p_0, 4 \to p_1, 4 \to p_2, 4 \to p_0, 2 \to p_1, 2 \to p_2, 2 \to p_0, 1 \to p_4, 1$

is simulated by this abstract trace, which is extracted from the abstract computation tree:

$p_0, even \to p_1, even \to p_2, even \to p_0, any \to p_1, any \to p_2, even \to p_0, any \to p_4, odd$

and indeed, *all* concrete traces starting with $p_0, 2n$, $n >= 0$, are simulated by the abstract tree in this manner.

## Proof of soundness of trace construction

For $S \in C$ and $a \in A$, say that $S \mathcal{R} a$ iff $S \sqsubseteq_C \gamma(a)$ iff $\alpha(S) \sqsubseteq_A a$.

**Lemma:** $\alpha \circ f \sqsubseteq_{A \to A} f^\# \circ \alpha$ **iff** $f \circ \gamma \sqsubseteq_{C \to C} \gamma \circ f^\#$ **iff** $S \mathcal{R} a$ implies $f(S) \mathcal{R} f^\#(a)$.

**Theorem:** For every concrete trace, $(p_i, s_i)_{i \geq 0}$, there exists an abstract trace, $(p_i, a_i)_{i \geq 0}$, such that for all $i \geq 0$, $\{s_i\} \mathcal{R} a_i$.

**Proof:** We use the Lemma and induction to assemble this diagram:

$$\begin{array}{lllll}
p_0, s_0 \longrightarrow & p_1, f_0(s_0) = p_1, s_1 & \longrightarrow p_2, f_1(s_1) = p_2, s_2 \longrightarrow \cdots \longrightarrow & p_i, s_i & \longrightarrow \cdots \\
\mathcal{R} & \mathcal{R} & \mathcal{R} & \mathcal{R} & \\
p_0, a_0 \longrightarrow & p_1, f_0^\#(a_0) = p_1, a_1 & \longrightarrow p_2, f_1^\#(a_1) = p_2, a_2 \longrightarrow \cdots \longrightarrow & p_i, a_i & \longrightarrow \cdots
\end{array}$$

(Note: each $s_i$ in the diagram is more precisely stated as $\{s_i\}$, because $C = \wp(\mathrm{Store})$.) Due to imprecision of the $f^\#$s, the abstract trace tree may possess many traces that begin with $p_0, a_0$, but there is always one trace in the tree that simulates the concrete trace.



When all the operations, $f^\#_{ij}$, are complete with respect to the $f_{ij}$s, the previous result is strengthened:

Say that $S \mathcal{R} a$ iff $\alpha(S) = a$. (Similarly, say that $S \mathcal{R} a$ iff $S = \gamma(a)$.)

In both cases, the lemma holds:

**Lemma:** $\alpha \circ f =_{A \to A} f^\# \circ \alpha$ **iff** $S \mathcal{R} a$ implies $f(S) \mathcal{R} f^\#(a)$.
(Similarly, $f \circ \gamma =_{C \to C} \gamma \circ f^\#$ **iff** $S \mathcal{R} a$ implies $f(S) \mathcal{R} f^\#(a)$.)

**Theorem** ($\alpha$-completeness): When $S \mathcal{R} a$ iff $\alpha(S) = a$, then for every concrete trace, $(p_i, s_i)_{i \geq 0}$, there exists an abstract trace, $(p_i, a_i)_{i \geq 0}$, such that for all $i \geq 0$, $\{s_i\} \mathcal{R} a_i$.

**Theorem** ($\gamma$-completeness): When $S \mathcal{R} a$ iff $\gamma(a) = S$, $S \subseteq \mathrm{Store}$, then for every trace on *sets of stores*, $(p_i, S_i)_{i \geq 0}$, there exists an abstract trace, $(p_i, a_i)_{i \geq 0}$, such that for all $i \geq 0$, $S_i \mathcal{R} a_i$.



## References


- P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. ACM POPL 1977.
- P. Cousot and R. Cousot. Systematic design of program analysis frameworks. ACM POPL, 1979.
- P. Cousot. Slides for invited lecture at VMCAI 2003, New York City. `www.di.ens.fr/~cousot`
- R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. Journal ACM 47(2) 2000.
- R. Giacobazzi and E. Quinterelli. Incompleteness, counterexamples, and refinements in abstract model-checking. SAS 2001, Springer LNCS 2126.
- D. Schmidt. Structure-preserving binary relations for program abstraction. In The Essence of Computation: Complexity, Analysis, Transformation. Springer LNCS 2566, 2002.