Informal introduction to abstract interpretation

A little graphical language

- objects;
- operations on objects.
Objects

An object is a pair:
- an origin (a reference point ×);
- a finite set of black pixels (on a white background).

Example of an object: a flower

Operations on objects: constants
- constant objects;
  for example:
  petal = 🌸

Operations on objects: rotation
- rotation \( r[a](o) \) of objects \( o \) (of some angle \( a \) around the origin):
  
  ![Rotation Diagram]
Example 1 of rotation

\[ \text{petal} = \quad r[45](\text{petal}) = \]

Example 2 of rotation

\[ \text{flower} = \quad r[-45](\text{flower}) = \]

Operations on objects: union

- union \( o_1 \cup o_2 \) of objects \( o_1 \) and \( o_2 \) = superposition at the origin;

  for example:

  \[ \text{corolla} = \text{petal} \cup r[45](\text{petal}) \cup r[90](\text{petal}) \cup r[135](\text{petal}) \cup r[180](\text{petal}) \cup r[225](\text{petal}) \cup r[270](\text{petal}) \cup r[315](\text{petal}) \]

Operations on objects: add a stem

- \( \text{stem}(o) \) adds a stem to an object \( o \) (up to the origin, with new origin at the root);
Flower

\[ \text{flower} = \text{stem} (\text{corolla}) \]

Contraints

- A corolla is the \( \subseteq \)-least object \( X \) satisfying the two constraints:
  - A corolla contains a petal:
    \[ \text{petal} \subseteq X \]
  - and, a corolla contains its own rotation by 45 degrees:
    \[ r[45](X) \subseteq X \]
- Or, equivalently \( ^1 \):
  \[ F(X) \subseteq X, \quad \text{where} \quad F(X) = \text{petal} \cup r[45](X) \]

\( ^1 \) By Tarski’s fixpoint theorem, the least solution is \( \text{lfp} \subseteq F \).

Fixpoints

- corolla = \( \text{lfp} \subseteq F \)
  \[ F(X) = \text{petal} \cup r[45](X) \]

Iterates to fixpoints

- The iterates of \( F \) from the infimum 0 are:
  \[ X^0 = 0, \]
  \[ X^1 = F(X^0), \]
  \[ \ldots \ldots \ldots \]
  \[ X^{n+1} = F(X^n), \]
  \[ \ldots \ldots \ldots \]
  \[ \text{lfp} \subseteq F = X^\omega = \bigcup_{n \geq 0} X^n. \]
Iterates for the corolla

The bouquet

- bouquet = r[-45](flower) ∪ flower ∪ r[45](flower)
- The bouquet:

Upper-approximation

- An upper-approximation of an object is an object with:
  - same origin;
  - more pixels.

Examples of upper-approximations of flowers
Abstract objects

- an abstract object is a mathematical/computer representation of an approximation of a concrete object;

\[
\begin{array}{c}
\text{concrete object} \\
\text{abstract object} \\
\text{more abstract object}
\end{array}
\]

Abstraction

- an abstraction function \( \alpha \) maps a concrete object \( o \) to an approximation represented by an abstract object \( \alpha(o) \).

Abstract domain

- an abstract domain is a set of abstract objects plus abstract operations (approximating the concrete ones);

Example 1 of abstraction
Example 2 of abstraction

Comparing abstractions
- larger pen diameters: more abstract;
- different pen shapes: may be non comparable abstractions.

Concretization
- a concretization function $\gamma$ maps an abstract object $\overline{o}$ to the concrete object $\gamma(\overline{o})$ that is represents (that is to its concrete meaning/semantics).

Example of concretization
Galois connection 1/4

- \( \alpha \) is monotonic.

\[
\begin{array}{ccc}
\text{flower} & \subseteq & \text{abstract flower} \\
\end{array}
\]

implies

\[
\begin{array}{ccc}
\text{flower} & \subseteq & \text{abstract flower} \\
\end{array}
\]

Galois connection 2/4

- \( \gamma \) is monotonic.

\[
\begin{array}{ccc}
\text{flower} & \subseteq & \text{abstract flower} \\
\end{array}
\]

implies

\[
\begin{array}{ccc}
\text{flower} & \subseteq & \text{abstract flower} \\
\end{array}
\]

Galois connection 3/4

- for all concrete objects \( x \), \( \gamma \circ \alpha(x) \supseteq x \).

\[
\begin{array}{ccc}
\text{flower} & \alpha(\text{flower}) & \gamma(\alpha(\text{flower})) \\
\end{array}
\]

Galois connection 4/4

- for all abstract objects \( y \), \( \alpha \circ \gamma(y) \subseteq y \).

\[
\begin{array}{ccc}
\text{abstract flower} & \gamma(\text{abstract flower}) & \alpha(\gamma(\text{abstract flower})) \\
\end{array}
\]
Galois connections

\[ \langle \mathcal{D}, \subseteq \rangle \xrightarrow{\alpha} \langle \mathcal{D}, \subseteq \rangle \]

iff \[ \forall x, y \in \mathcal{D} : x \subseteq y \implies \alpha(x) \subseteq \alpha(y) \]
\[ \land \forall \bar{x}, \bar{y} \in \bar{\mathcal{D}} : \bar{x} \subseteq \bar{y} \implies \gamma(\bar{x}) \subseteq \gamma(\bar{y}) \]
\[ \land \forall x \in \mathcal{D} : x \subseteq \gamma(\alpha(x)) \]
\[ \land \forall \bar{y} \in \bar{\mathcal{D}} : \alpha(\gamma(\bar{y})) \subseteq \bar{y} \]

iff \[ \forall x \in \mathcal{D}, \bar{y} \in \bar{\mathcal{D}} : \alpha(x) \subseteq y \iff x \subseteq \gamma(y) \]

Abstract ordering

- \( x \subseteq y \) is defined as \( \gamma(x) \subseteq \gamma(y) \).

Abstract petal

\[ \alpha(\text{petal}) = \text{petal} \]
Abstract rotations

- $r[a](y) = \alpha(r[a](\gamma(y)))$

A commutation theorem on abstract rotations

- $\alpha(r[a](x))$
  $= \alpha(\gamma(\alpha(r[a](x))))$
  $= \alpha(\gamma(r[a](\alpha(x))))$
  $= \alpha(r[a](\gamma(\alpha(x))))$
  $= r[a](\alpha(x))$

Abstract stems

- $\text{stem}(y) = \alpha(\text{stem}(\gamma(y)))$

abstract corolla
$\gamma(\text{abstract corolla})$
$\text{stem}(\gamma(\text{abstract corolla}))$
$\alpha(\text{stem}(\gamma(\text{abstract corolla})))$
Abstract union

- $x \sqcup y = \alpha(\gamma(x) \cup \gamma(y))$

Abstract bouquet: (cont'd)

= $\alpha(\bigcup)$

Abstract bouquet: (end)

= $\alpha(\bigcup)$
A theorem on the abstract bouquet

abstract flower = α(concrete flower)

abstract bouquet = r[-45](abstract flower) △ abstract flower △ r[-45](abstract flower)
= r[-45](α(concrete flower)) △ α(concrete flower) △ r[-45](α(concrete flower))
= α(r[-45](concrete flower)) △ α(concrete flower) △ α(r[-45](concrete flower))
= α(r[-45](concrete flower) △ concrete flower △ r[-45](concrete flower))
= α(concrete bouquet)

Abstract transformer \( \overline{F} \)

- \( \alpha(F(X)) \)
  = \( \alpha(\text{petal} \cup r[45](X)) \)
  = \( \alpha(\text{petal}) \cup \alpha(r[45](X)) \)
  = \( \alpha(\text{petal}) \cup r[45](\alpha(X)) \)
  = \( \text{abstract petal} \cup r[45](\alpha(X)) \)
  = \( \overline{F}(\alpha(X)) \)
by defining
\[
F(X) = \text{abstract petal} \cup r[45](X)
\]
and so:
- abstract corolla = \( \alpha(\text{concrete corolla}) = \alpha(\text{lfp} \subseteq F) = \text{lfp} \subseteq F \)

Abstract fixpoint

- abstract corolla = \( \alpha(\text{concrete corolla}) = \alpha(\text{lfp} \subseteq F) \)
  where \( F(X) = \text{petal} \cup r[45](X) \))

Iterates for the abstract corolla
Abstract interpretation of the (graphic) language

- Similar, but by **syntactic induction** on the structure of programs of the language;

On abstracting properties of graphic objects

- No, because we implicitly used the following implicit initial abstraction:

\[
\langle \varphi(\varphi(\mathcal{P})), \subseteq \rangle & \overset{\gamma_0}{\iff} \langle \varphi(\mathcal{P}), \subseteq \rangle \\
\]

where:

\[\mathcal{P}\text{ is a set of pixels (e.g. pairs of coordinates)}\]

\[\alpha_0(X) = \bigcup X\]

\[\gamma_0(Y) = \{G \in \mathcal{P} \mid G \subseteq Y\}\]

On abstracting properties of graphic objects

- A **graphic object** is a set of (black) pixels (ignoring the origin for simplicity);

- So a **property of graphic objects** is a set of graphic objects that is a set of sets of (black) pixels (always ignoring the set of origins for simplicity);

- Was there something **wrong**?

Is it for fun (only)?

- Yes, but see image processing by **morphological filtering**:


  It can be entirely formalized by **abstract interpretation**.
Objects

When analyzing/proving programs we have to consider “objects” that represent some part of the computation state, such as:

- **Values**: booleans, integers, … \( \mathcal{V} \)
- **Variable names**: \( X \)
- **Environments**: \( X \mapsto \mathcal{V} \)
- **Stacks**: assigning values to variables in the context of block-structured languages: \( \bigcup_{n \geq 0} ([1, n] \mapsto (X \mapsto \mathcal{V})) \)

- **Heaps**: dynamic allocation;
- **Control points**: procedure names, labels, …;
- **States**: control & memory states;

- **Finite prefix traces**;
- **Maximal finite or infinite traces** (for deterministic programs);
- **Sets of maximal finite or infinite traces** (for nondeterministic programs);
- …
Properties

Properties are “sets of objects” (which have that property).
Examples:
- odd naturals: \{1, 3, 5, \ldots, 2n + 1, \ldots\}
- even integers: \{2z \mid z \in \mathbb{Z}\}
- values of integer variables: \{z \in \mathbb{Z} \mid \text{minint} \leq z \leq \text{maxint}\}
- values of maybe uninitialized integer variables: \{z \in \mathbb{Z} \mid \text{minint} \leq z \leq \text{maxint}\} \cup \{\Omega_m \mid m \in \mathcal{M}\} \text{ where } \mathcal{M} \text{ is a set of error messages}

The complete lattice of concrete properties

The set of properties \( \varphi(\Sigma) \) of objects in \( \Sigma \) is a complete boolean lattice:

\[
(\varphi(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg)
\]

where
- A property \( P \in \varphi(\Sigma) \) is the set of objects which have the property \( P \)
- \( \subseteq \) is logical implication since \( P \subseteq Q \) means that all objects with property \( P \) have property \( Q \) (\( o \in P \implies o \in Q \))
- equality of two variables \( x \) and \( y \): \( \{\rho \in \mathcal{X} \mapsto \mathcal{V} \mid x, y \in \text{dom}(\rho) \land \rho(x) = \rho(y)\} \)
- invariance property (of a program with states in \( \Sigma \)): \( I \in \varphi(\Sigma) \)
- trace property: \( T \in \varphi(\Sigma^{\infty}) \)
- trace semantics property: \( P \in \varphi(\varphi(\Sigma^{\infty})) \)
- \( \emptyset \) is false (\( \text{ff} \))
- \( \Sigma \) is true (\( \text{tt} \))
- \( \cup \) is disjunction (objects which have either property \( P \) and/or have property \( Q \) belong to \( P \cup Q \))
- \( \cap \) is conjunction (object which have property \( P \) and have property \( Q \) belong to \( P \cap Q \))
- \( \neg \) is negation (objects not having property \( P \) are those in \( \Sigma \setminus P \))
Abstraction, informal introduction

- Abstraction replaces something “concrete” by a schematic description that account for some, and in general not all properties, either known or inferred i.e. an “abstract” model or concept.
- In practice, such an abstract model of a concrete object σ:
  - *can describe some* of the properties of the concrete object
  - *cannot describe all* properties of this concrete object.

Intuitive example 1 of abstraction

Cars \(\xrightarrow{\alpha}\) Trademarks
- A concrete property of cars is a set of cars.
- It can be abstracted by the set of their trademarks.
- A trademark is a set of cars.
- An abstract property of cars is a set of cars which, whenever it contains one car of some trademark, also contains all cars of that trademark.

Intuitive example 2 of abstraction

Scientific papers \(\xrightarrow{\alpha}\) set of keywords
- A concrete property of scientific papers is a set of scientific papers.
- Each scientific paper is abstracted by a list of keywords.
- A property of scientific papers can be abstracted by the list of keywords appearing in all papers with that property.
- An abstract property of scientific papers is therefore a set of papers which have all keywords belonging to the list.
Abstraction, definition of concrete and abstract properties

Abstraction in a reasoning/computation such that:
– Only some properties $A \subseteq \varphi(\Sigma)$ of the objects in $\Sigma$ can be used;
– The properties $P \in A$ that can be used are called abstract;
– The properties $P \in \varphi(\Sigma)$ are called concrete;

Direction of abstraction

– When approximating a concrete property $P \in \varphi(\Sigma)$, by an abstract property $\overline{P} \in A$, with $\overline{P} \neq P$, a relation must be established between the concrete $P$ and abstract property $\overline{P}$ to establish that

“$\overline{P} \in A$ is an approximation/abstraction of $P \in \varphi(\Sigma)$”

so as to ensure the soundness of the reasoning in the abstract with respect to the concrete, exact one.

– We consider essentially two cases:
  - Approximation from above: $P \subseteq \overline{P}$
  - Approximation from below: $P \supseteq \overline{P}$
– Other relations can be considered (e.g. probabilistic properties)
– The two notions are dual so formally only one need to be studied formally (approximation from above)
– In practice, useful approximation from below are much harder to discover.

– Abstract reasonings/computations involve sound approximations, in that:
  - The concrete properties that are also abstract can be used in the abstract reasoning/computation “as is”, without any loss of information;
  - The concrete properties $P \in \varphi(\Sigma) \setminus A$ which are not abstract cannot be used in the reasoning/computation and therefore must be approximated by some other abstract property $\overline{P} \in A$, which, since $P \neq \overline{P}$, involves some form of approximation.
Abstraction from below

\( \times \): points which have the concrete property \( P \)
\( \circ \): points which have the abstract property \( \bar{P} \)

- To answer the question \( \langle x, y \rangle \in P \)\? using only \( \bar{P} \) (such that \( P \subseteq \bar{P} \)):
  - If \( \langle x, y \rangle \notin \bar{P} \) then “I don’t know”
  - If \( \langle x, y \rangle \in \bar{P} \) then “Yes”

Why can an abstraction from above be “simpler” than the original concrete property?

- The concrete property is a set of objects
  - The objects are complex
  - The set can be infinite
  - In general their exists no suitable computer representation of the concrete property
- The abstract property is a larger set of objects
  - larger structures are in general even more expensive to store in the computer memory/compute with than smaller ones

but, well-chosen larger structures can have simpler encodings which can ve exploited for memorization and computation

- Example:
What to do in absence of (upper) abstraction?

- Assume a mechanized reasoning about a computer systems with objects/states $\Sigma$, we use an abstraction $A \subset \wp(\Sigma)$
- Assume concrete properties $P \in \wp(\Sigma)$ which cannot be expressed in the abstract, must be approximated from above by $\overline{P} \in A : P \supseteq \overline{P}$
- How should the mechanized reasoning proceed when some property $P$ has no abstraction $\overline{P} \in A$ from above ($\forall \overline{P} \in A : P \not\supseteq \overline{P}$)?
  - loop?
  - block?
  - ask for help?
  - fail?
  - answer something sensible!
- The only way to be always able to say something sensible for all $P \in \wp(\Sigma)$ is to assume that $\Sigma \in A$:
  Any concrete property should be approximable by “I don’t know” (i.e. $\Sigma \in A$, $\Sigma$ meaning “true”)

Minimal abstractions

- Assume concrete properties $P \in \wp(\Sigma)$ must be approximated from above by $\overline{P} \in A \subset \wp(\Sigma)$ such that $P \subseteq \overline{P}$
- The smaller the abstract property $\overline{P}$ is, the most precise the approximation will be
- Obviously, there might be no minimal abstract property at all in $A$

- If a concrete property $P \in \wp(\Sigma)$ has minimal upper approximations $\overline{P} \in A$:
  - $P \subseteq \overline{P}$
  - $\beta P' : P \subseteq P' \subseteq \overline{P}$
  then such minimal approximations are more precise than the non-minimal ones
- So minimal abstract upper approximations, if any, should be preferred
- In particular, an abstract property $\overline{P} \in A$ is best approximated by itself
In absence of minimal abstraction

- A classical example of absence of minimal abstract upper-approximations is that of a disk with no minimal convex polyhedral approximation [1]
  - $\Sigma = \mathbb{R} \times \mathbb{R}$
  - $A = \text{convex polyhedra}$
- Absence of minimal approximation is shown by Euclidean's construction:

- In absence of minimal approximations, the approximation $P \subseteq P_1$ can always be approximated by a better one $P \subseteq P_2 \subseteq P_1$!
- Some arbitrary choice has to be performed. This case will be studied later (see [2]). So, in the following, we assume the existence of minimal approximations

Example of minimal abstractions in absence of a best approximation

- $x$ can be approximated by $y = \gamma(\overline{y})$ and $z = \gamma(\overline{z})$ but $y$ and $z$ are not comparable.

- The other possible upper approximations would be less precise (than both $y$ and $z$ in that particular example)
- Notice that $\gamma$ cannot be the upper adjoint of a Galois connection since it is not a complete meet morphism: $\gamma(\overline{y}) \land \gamma(\overline{z}) \neq \gamma(\overline{y} \land \overline{z})$
- Abstraction in absence of best approximation is studied in [2]

References

Which minimal abstraction to choose?

- If there are several minimal possible abstract approximations $P_1, P_2, \ldots$ of a concrete property $P$, the most useful choice (ultimately providing the most precise unformation) may depend upon the circumstances and on later reasonings/computations.

- Example: rule of signs.
  - In "1+0", it is better to choose '+', because of the rule '+ +' = '+', while '+ +' yields no information (I don't know).
  - In "-1+0", it is better to choose '-', because of the rules (- '+') = '-', and '-' + '-' = '-', while '-' + '+' yields no information (I don't know).
  - Both cases have to be tried (backtracking).

There can be infinitely many ones.

Best abstraction

- A very handy choice of the abstract properties $A \subseteq \varphi(\Sigma)$ is when every concrete property $P$ has a best approximation $P \in P$:
  - $P \subseteq \overline{P}$
  - $\forall P' \in A : (P \subseteq P') \implies (P \subseteq P')$

- It follows that $P$ is the glb of the over-approximations of $P$ in $A$:

$$P = \bigcap\{P' \in A \mid P \subseteq P'\} \in A$$

Proof. We have $\forall P \in \{P' \in A \mid P \subseteq P\} : P \subseteq \overline{P}$, so $P \subseteq \bigcap\{P' \in A \mid P \subseteq P\}$ by def. glb $\cap$.

- Moreover $\forall P' \in A : (P \subseteq P') \implies (\bigcap\{P' \in A \mid P \subseteq P\} \subseteq P')$

- It follows that $P = \bigcap\{P' \in A \mid P \subseteq P\}$

- So $\exists P : (P \subseteq P) \land (\forall P' \in A : (P \subseteq P') \implies (P \subseteq P')) \iff P = \bigcap\{P' \in A \mid P \subseteq P\} \in A$

$\square$
The abstract domain is a Moore family

**Theorem.** The hypothesis that any concrete property \( P \in \wp(\Sigma) \) has a best abstraction \( \bar{P} \in A \), implies that the abstract domain \( A \) is a Moore family.

**Proof.** Let \( X \subseteq A \) be a set of abstract properties. Its intersection \( \bigcap X \) has a best approximation \( \bar{P} \in A \). We have therefore
\[
\bar{P} = \bigcap \{ \bar{P} \in A \mid \bigcap X \subseteq \bar{P} \}
\]
But \( \forall \bar{P} \in X : \bigcap X \subseteq \bar{P} \) and \( X \subseteq A \) so \( X \subseteq \{ \bar{P} \in A \mid \bigcap X \subseteq \bar{P} \} \) and therefore \( \bigcap \{ \bar{P} \in A \mid \bigcap X \subseteq \bar{P} \} \subseteq \bigcap X \) by def. of glb. By antisymmetry, \( \bigcap X = \bigcap \{ \bar{P} \in A \mid \bigcap X \subseteq \bar{P} \} = \bar{P} \in A \), proving \( A \) to be a Moore family. \( \square \)

In particular \( \bigcap \emptyset = \Sigma \in A \), which is consistent with our hypothesis that \( A \) should contain \( \Sigma \) to have the ability to express “I don’t know”.

---

Example and counter-example of Moore family based abstraction

- **Example:** rule of signs with best approximation of 0

- **Counter-example:** rule of signs without best approximation of 0
Closure operator based abstraction

Assume that the abstract domain $A$ is a Moore family of the concrete domain $\langle \rho(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, - \rangle$. Then the abstraction map is

$$\rho \in \rho(\Sigma) \mapsto A$$

$$\rho(P) \overset{\text{def}}{=} \bigcap\{ P \in A \mid P \subseteq P \}$$

Then $\rho$ is an upper closure operator on $\rho(\Sigma)$.

**Proof.** $\rho$ is the closure operator induced by the Moore family, a result simply depending on the fact that $\langle \rho(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap \rangle$ is a complete lattice. 

Example of abstraction map
Equivalent specification of an abstraction by a Moore family and a closure operator

In case of existence of a best abstraction, it is equivalent to specify the abstraction domain $A$

1. as a Moore family $\mathcal{M}$
2. as a closure operator $\rho$

**Proof.**
- Given $\mathcal{M}$ defined $\rho(P) = \bigcap \{\mathcal{P} \in \mathcal{M} | P \subseteq \mathcal{P} \}$ so that $A = \mathcal{M} = \rho(\Sigma)$
- Conversely, given a closure operator $\rho$, define $A = \rho(\rho(\Sigma)) = \{\rho(P) | P \in \rho(\Sigma)\}$ which is therefore the set of fixpoints of $\rho$ whence a Moore family since $\rho$ operates on a complete lattice

Examples of specifications of an abstraction by a Moore family and a closure operator

- The *most imprecise abstraction* is “I don’t know”
  - $\mathcal{M} = \{\Sigma\}$
  - $\rho = \lambda P. \Sigma$
- The *most precise abstraction* is “identity”
  - $\mathcal{M} = \rho(\Sigma)$
  - $\rho = \lambda P. P$

Generalizing to complete lattices

- The reasoning on abstractions of concrete properties $\langle \rho(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$ to an abstract domain which, in case of best abstraction is a Moore family, whence a complete lattice, can be generalized to an arbitrary concrete complete lattice $\langle L, \subseteq, \bot, \top, \cup, \cap \rangle$
- This allow a compositional approach where $\langle L, \subseteq, \bot, \top, \cup, \cap \rangle$ is abstracted to $\langle A_1, \subseteq_1, \bot_1, \top_1, \cup_1, \cap_1 \rangle$ which itself can be further abstracted to $\langle A_2, \subseteq_2, \bot_2, \top_2, \cup_2, \cap_2 \rangle$, ...

Correspondance between concrete and abstract properties
Given a closure operator $\rho$ on a poset $\langle L, \sqsubseteq \rangle$ (typically $L$ is $\rho(\Sigma)$), Morgado’s theorem states that for all $P, P' \in L$

$$\rho(P) \sqsubseteq \rho(P') \iff P \sqsubseteq \rho(P')$$

that is, by definition of Galois connections ($1_L \overset{\text{def}}{=} \lambda x \in L : x$):

$$\langle L, \sqsubseteq \rangle \overset{1_L}{\longrightarrow} \langle \rho(L), \sqsubseteq \rangle$$

**Proof.** We must prove $\forall z \in L : \forall y \in \rho(L) : (\rho(z) \sqsubseteq y) \iff (z \sqsubseteq 1_L(y))$.

We have $y \in \rho(L)$ iff $\exists z \in L : \rho(z) = y$ so that this condition is equivalent to $\forall z, y \in L : (\rho(z) \sqsubseteq \rho(y)) \iff (z \sqsubseteq \rho(y))$ which directly follows from Morgado’s theorem. Moreover, $\rho$ is surjective on $\rho(L)$. □

Correspondance between concrete and representations of abstract properties

- Let $\langle A, \leq \rangle$ be an order-isomorphic representation of the abstract domain $\langle \rho(L), \sqsubseteq \rangle$. We have

$$\langle \rho(L), \sqsubseteq \rangle \overset{\epsilon}{\longrightarrow} \langle A, \leq \rangle$$

where $\epsilon^{-1}$ is the inverse of the bijection $\epsilon \in \rho(L) \mapsto A$ and $\epsilon \in \rho(L) \mapsto A$.

By composition, we get:

$$\langle L, \sqsubseteq \rangle \overset{1_L \circ \epsilon^{-1}}{\longrightarrow} \langle A, \leq \rangle$$

Specification of an abstract domain by a Galois surjection

- Inversely, we can consider a Galois surjection

$$\langle A, \leq \rangle \overset{\gamma}{\longrightarrow} \langle \rho(L), \sqsubseteq \rangle$$

- Then $\rho = \gamma \circ \alpha$ is a closure operator and $\langle A, \leq \rangle$ is order-isomorphic to $\langle \rho(L), \sqsubseteq \rangle$.

- We have an order-isomorphic representation of the abstract domain $\langle \rho(L), \sqsubseteq \rangle$, which is a Moore family.
Specification of an abstract domain by a Galois surjection, example

Because $\alpha$ is surjective, $\gamma$ is injective and order is preserved, each element in the Moore family $\{\bot, 0, \bot, +, \top\}$ has a unique isomorphic representation $\{\bot, 0, -1, +1, \top\}$. This would not be the case when $\alpha$ is not surjective.

A graphical illustration of the specification of an abstraction by a Galois surjection

- Abstraction of a set of point in $\mathbb{R}^2$ by an interval:

- Concretization:

Galois Connection $\langle L, \sqsubseteq \rangle \xleftarrow{\alpha} \langle \overline{L}, \sqsubseteq \rangle$

- $\blacklozenge$: Moore family of best approximations;
- $\blacklozenge$: concrete values with the same abstraction.

- The abstraction $\alpha$ is monotone:
- The concretization $\gamma$ is monotone:

![Diagram showing monotonicity of $\gamma$]

- The composition $\alpha \circ \gamma$ is:
  - The identity for Galois surjections
  - Reductive (indeed a lower-closure operator) for Galois connections

![Diagram showing composition]

- The intuition of $\subseteq$ is that $P \subseteq P'$ implies $\gamma(P) \subseteq \gamma(P')$ so that $P$ is more precise than $P'$ when expressed in the concrete

- So $\alpha \circ \gamma(P) \subseteq P$ means that concretization can lose no information, since if the concrete property $P$ is overapproximated by $P$ then

$$P \subseteq \gamma(P) \iff P \subseteq \gamma(\alpha \circ \gamma(P))$$

so that using $P$ or $\alpha \circ \gamma(P)$ is exactly the same in the concrete, as far as precision is concerned.
Why are abstract domains complete lattices in the presence of best abstractions?

- The abstractions start from the complete lattice of concrete properties \((\rho(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg)\) where objects in \(\Sigma\) represent program computations and the elements of \(\rho(\Sigma)\) represent properties of these program computations.
- We have defined abstract domains with best approximations in three equivalent different ways (more are considered in [3]):
  - As a Moore family;
  - As a closure operator (which fixpoints form the abstract domain);
  - As the image of the concrete domain by a Galois surjection.

In all cases, it follows that the abstract domain is a complete lattice, since we have seen that:

- A Moore family of a complete lattice is a complete lattice;
- The image of a complete lattice by an upper closure operator is a complete lattice (Ward);
- The image of a complete lattice by the surjective abstraction of a Galois connection is a complete lattice.

In general this property does not hold in absence of best abstraction or if arbitrary points are added to the abstract domain as shown next.

Relaxing the condition on the uniqueness of the representation of abstract properties: Galois connections

- Assume the correspondence between concrete and abstract properties is a non-surjective Galois connection:
  \[\langle L, \subseteq \rangle \xrightarrow{\alpha} \langle A, \leq \rangle\]

\(\gamma\) is not surjective, which means that at least two different abstract properties \(P_1\) and \(P_2\) have exactly the same concretization:

\[P_1 \neq P_2 \land \gamma(P_1) = \gamma(P_2)\]

Example of non-surjective Galois connection based abstraction

Here “1” and “+1” are two different encodings of the same concrete property (i.e; positive or zero).
Reduction

- With non-surjective Galois connections \((\mathcal{L}, \sqsubseteq) \xrightarrow{\gamma} (A, \leq)\), there are at least two different representations in the abstract of at least one concrete property.
- This may happen when abstract computer representations of the same concrete property are not unique (e.g. sets represented by ordered trees).
- Reduction is always mathematically possible, by considering \((\mathcal{L}, \sqsubseteq) \xrightarrow{\alpha} (A_{\equiv}, \leq_{\equiv})\) where \(\mathcal{P} \equiv \mathcal{P}' \iff \gamma(\mathcal{P}) = \gamma(\mathcal{P}')\), \(\alpha_{\equiv}(\mathcal{P}) = [\alpha(\mathcal{P})]_{\equiv}\), \(\gamma([\mathcal{P}]_{\equiv}) = \gamma(\mathcal{P})\) and \([\mathcal{P}]_{\equiv} \leq_{\equiv} [\mathcal{P}']_{\equiv} \iff \mathcal{P} \leq \mathcal{P}'\).

Example:
- Abstract properties are intervals \([a, b]\) meaning \(\gamma([a, b]) \overset{\text{def}}{=} \{x \mid \minint \leq a \leq x \leq b \leq \maxint\}\).
- The empty set is represented by any \([a, b]\) with \(b < a\). This can be left as is or normalized as e.g. \([\maxint, \minint]\).
- The supremum is represented by any \([a, b]\) with \(a \leq \minint\) and \(\maxint \leq b\). This can be left as is or better normalized as e.g. \([\minint, \maxint]\).
- Sometimes it is better to have a “normal form”, but this reduction may also be sometimes algorithmically very expensive.

Standard examples of abstractions formalized by Galois connections

Subset restriction abstraction

If
- \(C\) is a set, \(A \subset C\) is a strict subset
- \(\alpha_A(X) \overset{\text{def}}{=} X \cap A\)
- \(\gamma_A(Y) \overset{\text{def}}{=} (Y \cup \lnot A)\) where \(\lnot A \overset{\text{def}}{=} C \setminus A\)

then
\[
(\rho(C), \subseteq) \overset{\gamma_A}{\leftrightarrow} (\rho(A), \subseteq)
\]

**Proof.** \(\alpha_A(X) \subseteq Y \iff (X \cap A) \subseteq Y \iff X \subseteq (Y \cup \lnot A) \iff X \subseteq \gamma_A(Y)\).
If \(Y \in \rho(A)\) then \(\alpha(A) = A\) proving \(\alpha\) to be onto. \(\Box\)
– The intuition is that we approximate a set by remembering a few members
– Example: tests
  - \( C \) is the set of all program execution traces
  - \( A \) is the subset of traces explored by tests
  - \( \gamma(P) = P \cup \neg A \) i.e. nothing is known about execution traces which have not been tested!
– Example: keywords of a scientific paper

Subset inclusion abstraction

If
- \( C \) is a set, \( S \subset C \) is a strict subset
- \( \alpha_S(X) \overset{\text{def}}{=} (X \subseteq S) \)
- \( \gamma_S(b) \overset{\text{def}}{=} \{ b \ ? S \supsetneq C \} \)
then
\[
\langle \varphi(C), \subseteq \rangle \xleftarrow{\alpha_S} \langle B, \subseteq \rangle
\]

Proof. \( \alpha_S(X) \iff b \iff b \Rightarrow (X \subseteq S) \iff \{ b \ ? (X \subseteq S) : \parallel \} \iff \{ b \ ? (X \subseteq S) : (X \subseteq C) \} \iff X \subseteq \gamma_S(b). \alpha_S \) is onto since \( \alpha_S(S) = \parallel \) and \( \alpha_S(C) = (C \subseteq S) = \parallel \) since \( S \subset C \). \( \square \)

Elementwise/homomorphic abstraction

If
- \( @ : C \mapsto A \) elementwise abstraction
- \( \alpha_{@}(P) \overset{\text{def}}{=} \{ @p \mid p \in P \} \) set abstraction
- \( \gamma_{@}(Q) \overset{\text{def}}{=} \{ p \mid @p \in Q \} \) concretization
then
\[
\langle \varphi(C), \subseteq \rangle \xleftarrow{\alpha_{@}} \langle \varphi(A), \subseteq \rangle
\]

Proof. Given \( X \subseteq C \) and \( Y \subseteq A \), we have
- \( \alpha_{@}(X) \subseteq Y \)
- \( \{ @x \mid x \in X \} \subseteq Y \) \( \overset{\text{def}. \alpha_{@}}{=} \)
- \( \forall x \in X : @x \in Y \) \( \overset{\text{def}. / \subseteq}{=} \)
\[ X \subseteq \{ x \in C \mid \Theta(x) \in Y \} \quad \{ \text{def.} / \subseteq \} \]
\[ X \subseteq \gamma_{\Theta}(Q) \quad \{ \text{def.} / \gamma_{\Theta} \} \]

– The intuition is to remember only one characteristics \( \Theta(x) \) of the objects \( x \) which satisfy the concrete property.

– Example 1: trademark of cars
  - \( C \): set of all cars
  - \( A \): set of all car trademarks
  - \( \Theta(c) \): trademark of car \( c \)
  - \( \Theta(X) \): trademarks of cars in set \( X \)
  - \( \gamma_{\Theta}(T) \): all cars which trademark is in \( T \)

– Example 2: signs
  - \( \Theta : \mathbb{Z} \mapsto \{ - , 0 , + \} \)
    - \( \Theta(z) \) \( \overset{\text{def}}{=} - \) if \( x < 0 \)
    - \( \Theta(z) \) \( \overset{\text{def}}{=} 0 \) if \( x = 0 \)
    - \( \Theta(z) \) \( \overset{\text{def}}{=} + \) if \( x > 0 \)

– Example 3: congruence
  - \( \Theta \in \mathbb{Z} \mapsto \mathbb{Z}/n, n = 3 \)
  - \( \Theta(x) \) \( \overset{\text{def}}{=} |x| \text{ mod } n \)

\[ \langle \mathcal{P}(C) , \subseteq \rangle \overset{\gamma_{\Theta}}{\leftarrow} \langle \mathcal{P}(A) , \subseteq \rangle \text{ when } \Theta : C \mapsto A \text{ is onto} \]
– Often rediscovered (e.g. abstract model-checking)
– Often misunderstood (e.g.: \( \langle \Theta , \gamma \rangle \) instead of \( \langle \alpha , \gamma \rangle \))

Functional abstraction

If
– \( D^\parallel \) and \( D^\parallel \) are sets
– \( \langle L , \subseteq , \bot , T , \sqcup , \sqcap \rangle \) is a complete lattice
– \( \Theta : D^\parallel \mapsto D^\parallel \)
– \( \alpha \in (D^\parallel \mapsto L) \mapsto (D^\parallel \mapsto L) \)
  \( \alpha(f) \overset{\text{def}}{=} \lambda y . \sqcup\{ f(x) \mid \Theta(x) = y \} \)
– \( \gamma \in (D^\parallel \mapsto L) \mapsto (D^\parallel \mapsto L) \)
  \( \gamma(g) \overset{\text{def}}{=} g \circ \Theta \)
then
\[ \langle D^\parallel \mapsto L , \bot \rangle \overset{\gamma}{\leftarrow} \langle D^\parallel \mapsto L , \bot \rangle \]
Proof. \( \alpha(f) \subseteq g \iff \forall y : \alpha(f)(y) \subseteq g(y) \iff \forall y : \bigcup \{ f(x) \mid \Theta(x) = y \} \subseteq g(y) \iff \forall y : \exists x : (\Theta(x) = y) \implies (f(x) \subseteq g(y)) \iff f \subseteq g \cdot \Theta \iff f \subseteq \gamma(g) \)

- Example: this is a generalization of the previous homomorphic abstraction with \( L = \{ \text{ff}, \text{tt} \} \) ordered with \( \text{ff} \subseteq \text{tt} \) using the representation of sets by their characteristic function.

Relational Abstraction

If
- \( \Theta \in D \times D \)
- \( \alpha \in \wp(D) \rightarrow \wp(D) \)
- \( \alpha(X) \overset{\text{def}}{=} \{ y \mid \exists x \in X : \langle x, y \rangle \in \Theta \} = \text{post}[\Theta]X \)
- \( \gamma \in \wp(D) \rightarrow \wp(D) \)
- \( \gamma(Y) \overset{\text{def}}{=} \{ x \mid \forall y : (\langle x, y \rangle \in \Theta) \implies (y \in Y) \} = \text{pre}[\Theta]Y \)

then
\[
\langle \wp(D), \subseteq \rangle \overset{\gamma}{\xrightarrow{\alpha}} \langle \wp(D), \subseteq \rangle
\]

Relational lattice abstraction

If
- \( \langle A, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle \) Abstract domain (complete lattice)
- \( \rho \in \wp(C \times A) \) Abstraction relation
- \( \alpha_{\rho}(P) \overset{\text{def}}{=} \bigcup \{ a \in A \mid \exists p \in P : \langle p, a \rangle \in \rho \} \) Set abstraction
- \( \gamma_{\rho}(a) \overset{\text{def}}{=} \{ p \mid \forall a' \in A : (\langle p, a' \rangle \in \rho) \implies (a' \sqsubseteq a) \} \) Concretization

then
\[
\langle \wp(C), \subseteq \rangle \overset{\gamma_{\rho}}{\xrightarrow{\alpha_{\rho}}} \langle A, \sqsubseteq \rangle
\]
Proof. \( \forall P \subseteq C, \forall Y \in A: \)
\[
\alpha_p(P) \subseteq Y \\
\mid \{ a \in A | \exists p \in P : (p, a) \in \rho \} \subseteq Y \\
\mid \forall a \in A : (\exists p \in P : (p, a) \in \rho) \implies (a \subseteq Y) \\
\mid \forall a \in A : \forall p \in P : ((p, a) \in \rho) \implies (a \subseteq Y) \\
\mid \forall p \in P : \forall a \in A : ((p, a) \in \rho) \implies (a \subseteq Y) \\
\mid P \subseteq \gamma(p(a)) \\
\mid \gamma(p(a)) \subseteq Y)
\]
\( \square \)

- Intuition: the weakest abstract property in relation with the concrete elements

**Example 2** (semi-dual of example 1):

If
- \( @ \in D^h \mapsto p(D^h) \)
- \( \alpha \in p(D^h) \mapsto p(D^h) \)
- \( \gamma \in p(D^h) \mapsto p(D^h) \)
- \( \alpha(X) \defeq \bigcup \{ @x | x \in X \} \)
- \( \gamma(Y) \defeq \{ y | @y \subseteq Y \} \)
then
\[
\langle p(D^h), \subseteq \rangle \xRightarrow[\alpha]{\subseteq} \langle p(D^h), \subseteq \rangle
\]

Proof. \( \alpha(X) \subseteq Y \iff \bigcup \{ @x | x \in X \} \subseteq Y \iff \forall x \in X : @x \subseteq Y \iff X \subseteq \{ x | @x \subseteq Y \} \iff X \subseteq \gamma(Y). \) \( \square \)

**Minimal Abstraction**

If
- \( \langle L, \leq, 0, 1, \land, \lor \rangle \) complete lattice
- \( \alpha_m \in p(L) \mapsto L \)
- \( \alpha_m(S) \defeq \land S \)
- \( \gamma_m \in L \mapsto p(L) \)
- \( \gamma_m(x) \defeq \{ x | \leq x \} \)
then
\[
\langle p(L), \subseteq \rangle \xRightarrow[\gamma_m]{\subseteq} \langle L, \geq \rangle
\]

Proof.
\[
\alpha_m(S) \geq \ell
\]
Given $S \in L$, $\alpha\{S\} = \wedge\{S\} = S$ proving $\alpha$ onto.

- Approximate a set by its glb
  - Example: $\langle \mathbb{Z} \cup \{-\infty, +\infty\}, \leq, -\infty, +\infty, \min, \max \rangle$

- Approximate a set by its lub
  - Example 1: $\langle \mathbb{Z} \cup \{-\infty, +\infty\}, \leq, -\infty, +\infty, \min, \max \rangle$

- Example 2: $\Sigma$: objects, $\wp(\Sigma)$: semantics, $\wp(\wp(\Sigma))$: semantic properties, $\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap \rangle$ complete lattice and $\langle \wp(\wp(\Sigma)), \subseteq \rangle \overset{\lambda S, \wp(S)}{\cong} \langle \wp(\Sigma), \subseteq \rangle$, as already seen with the flowers example on page 51.

Maximal abstraction

If
- $\langle L, \leq, 0, 1, \wedge, \vee \rangle$ complete lattice
- $\alpha_M \in \wp(L) \mapsto L$
  $\alpha_M(S) \overset{\text{def}}{=} \bigvee S$
- $\gamma_M \in L \mapsto \wp(L)$
  $\gamma_M(\ell) \overset{\text{def}}{=} \{ x \in L \mid \ell \geq x \}$
then

$$\langle \wp(L), \subseteq \rangle \overset{\gamma_M}{\bigwedge} \overset{\alpha_M}{\bigvee} \langle L, \leq \rangle$$

Proof. Duality on $L$.

Interval abstraction

If
- $\langle L, \leq, 0, 1, \wedge, \vee \rangle$ complete lattice
- $\alpha_i \in \wp(L) \mapsto L \times L$
  $\alpha_i(S) \overset{\text{def}}{=} \bigwedge S \bigvee S$
- $\gamma_i \in L \times L \mapsto \wp(L)$
  $\gamma_i([a, b]) \overset{\text{def}}{=} \{ x \in L \mid a \leq x \leq b \}$
then

$$\langle \wp(L), \subseteq \rangle \overset{\gamma_i}{\bigwedge} \overset{\alpha_i}{\bigvee} \langle L \times L, \subseteq \rangle$$

where $[a, b] \subseteq [a', b'] \overset{\text{def}}{=} (a' \leq a \wedge b \leq b')$
Proof.

\[ \alpha_m(S) \geq [a, b] \]
\[ \iff [\wedge S, \vee S] \subseteq [a, b] \]
\[ \iff a \leq \wedge S \wedge \vee S \leq b \]
\[ \iff \forall x \in S : a \leq x \wedge \forall y \in S : y \leq b \]
\[ \iff \forall x \in S : a \leq x \leq b \]
\[ \iff S \subseteq \{ x \in L \mid a \leq x \leq b \} \]
\[ \iff S \subseteq \gamma_s([a, b]) \]

This is also the reduced product of the minimal and maximal abstractions (see later).

Abstraction of a function at a point

If
- \( S \) is a set with element \( s \in S \)
- \( \langle L, \subseteq, \bot, T, \cup, \cap \rangle \) is a complete lattice
- \( \alpha_s(f) \) is the infimum
- \( \gamma_s(v) \) is the supremum

then
\[ \langle S \mapsto L, \subseteq \rangle \xrightarrow{\gamma_s} \langle L, \subseteq \rangle \]

Proof. \( \alpha_s(f) \subseteq v \iff f(s) \subseteq v \iff \forall x \in S : f(x) \subseteq \{ x = s \? v : T \} \iff \forall x \in S : f(x) \subseteq \gamma_s(v)(x) \iff f \subseteq \gamma_s(v) \]

Abstraction of a function at a set of points

If
- \( S \) is a set with subset \( T \in S \)
- \( \langle L, \subseteq, \bot, T, \cup, \cap \rangle \) is a complete lattice
- \( \alpha_T(f) \) is the infimum
- \( \gamma_T(\varphi) \) is the supremum

then
\[ \langle S \mapsto L, \subseteq \rangle \xrightarrow{\gamma_T} \langle L, \subseteq \rangle \]

Proof. \( \alpha_T(f) \subseteq \varphi \iff \forall y \in T : \alpha_T(f)(y) \subseteq \varphi(y) \iff \forall y \in T : f(y) \subseteq \varphi(y) \iff \forall x \in S : f(x) \subseteq \{ x \in T \? \varphi(x) : T \} \iff f \subseteq \lambda x \in S : \{ x \in T \? \varphi(x) : T \} \iff \forall x \in S : f(x) \subseteq \gamma_T(\varphi) \iff f \subseteq \gamma_s(\varphi) \).
Example: approximate an invariance specification attaching an invariant assertion to each program point of a program/procedure by a precondition at the program/procedure entry and a postcondition at the program/procedure exit:

- $S$: set of program points
- $T = \{\text{entry point, exit point}\}$
- $L = \varphi(X \mapsto V)$, assertions
- $I \in S \mapsto L$, invariants attached to each program point
- The top is $X \mapsto V$ (“I don’t know”)

\[\iff \forall y \in T : \alpha_f(\varphi)(y) \sqsubseteq \psi(y)\]  
\[\iff \forall y \in T : \varphi \circ f(y) \sqsubseteq \psi(y)\]  
\[\iff \forall x \in S : \forall y \in T : (f(y) = x) \implies (\varphi(x) \sqsubseteq \psi(y))\]  
\[\iff \forall x \in S : \varphi(x) \sqsubseteq \bigcap \{\psi(y) \mid y \in T \land f(y) = x\}\]  
\[\iff \varphi \sqsubseteq \lambda \varphi \circ f\]  
\[\iff \alpha_f(\varphi) \sqsubseteq \psi\]  
\[\iff \alpha_f(\varphi) \sqsubseteq \psi\]  
\[\iff \gamma_f(\psi) \sqsubseteq \psi\]  
\[\iff \gamma_f(\psi) \sqsubseteq \psi\]  

- We get the previous example (on page 140) with $L = \{\tt, \emptyset\}$ and $f$ is an injection.

Lattice abstraction of a function at a set of points

If

- $S$ and $T$ are sets
- $f \in T \mapsto S$
- $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice
- $\alpha_f \defeq \lambda \varphi \cdot \varphi \circ f$
- $\gamma_f \defeq \lambda \psi \cdot \lambda x \cdot \bigcap \{\psi(y) \mid y \in T \land f(y) = x\}$

then

\[\langle S \mapsto L, \sqsubseteq \rangle \xrightarrow{\gamma_f \alpha_f} \langle T \mapsto L, \sqsubseteq \rangle\]

Proof.

\[\alpha_f(\varphi) \sqsubseteq \psi\]

Abstraction of a set of functions by a function

If

- $\Phi \subseteq D \mapsto C$ is a set of functions
- $\alpha_f(\Phi) \defeq \lambda \varphi \cdot \{\varphi(x) \mid \varphi \in \Phi\}$
- $\gamma_f(\Psi) \defeq \{\varphi \mid \forall x \in D : \varphi(x) \in \Psi(x)\}$
- $\alpha_p(\Phi) \defeq \lambda X \cdot \bigcup \{\Phi(x) \mid x \in X\}$
- $\gamma_p(\Psi) \defeq \lambda X \cdot \Phi(\{x\})$

then

\[\langle \rho(D \mapsto C), \sqsubseteq \rangle \xrightarrow{\gamma_f \alpha_f} \langle D \mapsto \rho(C), \sqsubseteq \rangle \xrightarrow{\rho \alpha_f} \langle \rho(D) \mapsto \rho(C), \sqsubseteq \rangle\]

Proof.

Given $\Phi \in \rho(D \mapsto C)$, $\Psi \in D \mapsto \rho(C)$
Given $\psi \in D \mapsto p(C)$ and $X_i \subset D$, $i \in \Delta$, we have:

$$\alpha_p(\psi)(\bigcup_{i \in \Delta} X_i) = \bigcup_{x \in \bigcup_{i \in \Delta} X_i} \Phi(x)$$

proving that $\alpha_p$ is a bijection with inverse $\gamma_p$.

$$\alpha_p(\psi) \subseteq \psi \quad \gamma_p \circ \alpha_p(\psi) \subseteq \gamma_p(\psi) \quad \phi \subseteq \gamma_p(\psi)$$

$\square$

- Example 1: often used as a set-transformer collecting semantics $p(D) \mapsto p(C)$ for functions (possible values of the result as a function of the possible values of the argument, not general enough as a collecting semantics (use $p(D \mapsto C)$, see [4])

- Example 2 (Cartesian abstraction):

  - Given a set $X$ of variables and $V$ of values, properties $P$ of environments $\rho \in X \mapsto V$ belong to $p(X \mapsto V)$ that are sets of functions
Abstraction of lattice functionals

If
- $S, T$ are sets
- $(L, \subseteq, \perp, \sqcup, \sqcap)$ is a complete lattice
- $F \subseteq G \triangleq \forall f \in S \rightarrow T : F(f) \subseteq G(f)$
- $\varphi \subseteq \psi \triangleq \forall s \in S : \forall y \in T : \varphi(z)(y) \subseteq \psi(z)(y)$
- $\alpha(F) \triangleq \lambda z \cdot \lambda y \cdot \bigcup \{F(f) \mid f(x) = y\} \subseteq \varphi$
- $\gamma(\varphi) \triangleq \lambda f \in S \rightarrow T \cdot \prod \{\varphi(z)(f(x)) \mid z \in S\}$

then
$$(S \mapsto T) \mapsto L, \sqsubseteq \overset{\gamma}{\mapsto} (S \mapsto (T \mapsto L), \sqsubseteq)$$

Proof.

$\alpha(F) \sqsubseteq \varphi$

$\iff \lambda z \cdot \lambda y \cdot \bigcup \{F(f) \mid f(x) = y\} \subseteq \varphi$

$\iff \forall z \in S : \forall y \in T : \bigcup \{F(f) \mid f(x) = y\} \subseteq \varphi(z)(y)$

$\iff \forall z \in S : \forall y \in T : \forall f \in S \rightarrow T : f(x) = y \Rightarrow (F(f) \subseteq \varphi(z)(y))$

$\iff \forall z \in S : \forall f \in S \rightarrow T : F(f) \subseteq \varphi(z)(f(x))$

$\iff \forall f \in S \rightarrow T : \forall z \in S : F(f) \subseteq \varphi(z)(f(x))$

$\iff \gamma(\varphi)$

$\Box$

- The particular case of the previous example on page 144 is obtained with $L = \{tt, ff\}$ with $ff \sqsubseteq tt$

An important particular case is that of a pair $\langle x, y \rangle$ understood as $\{0 \mapsto x, 1 \mapsto y\}$ i.e. a map from selectors to fields.

Reference

Componentwise/Cartesian abstraction of a set of pairs

If
- \( \alpha_p(S) \) \stackrel{\text{def}}{=} \{ x \mid \exists y : (x, y) \in S \}, \{ y \mid \exists x : (x, y) \in S \} \)
- \( \gamma_p((X, Y)) \) \stackrel{\text{def}}{=} \{ (x, y) \mid x \in X \land y \in Y \}
- \( (x, y) \subseteq (x', y') \) \stackrel{\text{def}}{=} x \subseteq x' \land y \subseteq y'

then
\[ \langle \rho(D_1 \times D_2), \subseteq \rangle \iff \langle \rho(D_1) \times \rho(D_2), \subseteq \rangle \]

Proof.
- \( \alpha_p(S) \subseteq (X, Y) \)
- \( \iff \{ x \mid \exists y : (x, y) \in S \} \subseteq X \land \{ y \mid \exists x : (x, y) \in S \} \subseteq Y \)
- \( \iff S \subseteq \{ (x, y) \mid x \in X \land y \in Y \} \)
- \( \iff S \subseteq \gamma_p((X, Y)) \) \hspace{1cm} \Box \)

Intuition: abstraction of relational to independent properties, same as on page 144, up to the encoding

Example:
- \( \alpha_p(\{(1, 2), (3, 4)\}) = \{\{1, 3\}, \{2, 4\}\} \)
- \( \gamma_p(\{(1, 3), \{2, 4\}\}) \) \stackrel{\text{def}}{=} \{\{1, 2\}, \{1, 4\}, \{3, 2\}, \{3, 4\}\}

Pointwise abstraction composition

If
\[ \langle P, \leq \rangle \xrightarrow{\gamma} \langle Q, \subseteq \rangle \]
- \( \alpha(f) \) \stackrel{\text{def}}{=} \lambda s \in S. \alpha(f(s)) \quad \text{pointwise abstraction} \)
- \( \gamma(g) \) \stackrel{\text{def}}{=} \lambda s \in S. \gamma(g(s)) \)
- \( f \subseteq g \) \stackrel{\text{def}}{=} \forall x \in S : f(x) \subseteq g(x) \quad \text{pointwise ordering} \)

then
\[ \langle S \mapsto P, \leq \rangle \xrightarrow{\gamma} \langle S \mapsto Q, \subseteq \rangle \]

Proof.
- \( \alpha(f) \subseteq g \)
- Used to lift abstractions to vectors (e.g. from one to many variables, from one to many program points, etc.)
Example 1: attribute independent environment abstraction

- \(x \in X\): set of variables
- \(\mathcal{V}\): set of values
- \(\varphi(\mathcal{V})\): properties of values
- \(\rho \in X \mapsto \mathcal{V}\): environments
- \(\varphi(X \mapsto \mathcal{V})\): properties of environments

\[
\begin{align*}
\langle \varphi(X \mapsto \mathcal{V}), \subseteq \rangle & \xleftarrow{\gamma_f / \alpha_f} \langle X \mapsto \varphi(\mathcal{V}), \subseteq \rangle \\
\alpha_f(P) & \overset{\text{def}}{=} \lambda x \in X. \{\rho(x) | \rho \in P\}
\end{align*}
\]

Example 2: local invariants

- Same idea to abstract \(\varphi(C \mapsto (X \mapsto \mathcal{V}))\) where \(C\) is the set of control points
- \(\langle \varphi(C \mapsto (X \mapsto \mathcal{V})), \subseteq \rangle \leftrightarrow \langle C \mapsto (X \mapsto L), \subseteq \rangle\)

Componentwise abstraction composition

A particular case of pointwise abstraction applied to pairs (more generally to vectors) is the following. If

- \(\langle P, \leq \rangle \xleftarrow{\gamma_1 / \alpha_1} \langle P\# \text{, } \leq\# \rangle\)
- \(\langle P, \leq \rangle \xleftarrow{\gamma_2 / \alpha_2} \langle P\# \text{, } \leq\# \rangle\)
- \(\alpha \times \overset{\text{def}}{=} \lambda (x, y). (\alpha_1(x), \alpha_2(y))\) Componentwise abstraction
- \(\gamma \overset{\text{def}}{=} \lambda (x, y). (\gamma_1(x), \gamma_2(y))\) then
Higher-order abstraction composition

If
\[
\begin{align}
\langle P, \leq \rangle & \xrightarrow{\alpha_d} \langle P^\|, \leq \| \rangle \\
\langle Q, \leq \rangle & \xrightarrow{\alpha_c} \langle Q^\|, \leq \| \rangle \\
\alpha & \equiv \lambda \psi \cdot \alpha_c \circ \varphi \circ \gamma_d \\
\gamma & \equiv \lambda \psi \cdot \gamma_c \circ \psi \circ \alpha_d
\end{align}
\]
then
\[
\langle P \xrightarrow{m} Q, \leq \rangle \xrightarrow{\gamma} \langle P^\| \xrightarrow{m} Q^\|, \leq \| \rangle
\]

PROOF. Given monotonic \( f \in P \xrightarrow{m} Q \) and \( g \in P^\| \xrightarrow{m} Q^\| \), we have:
\[
\alpha(f) \preceq g
\]

- Intuition:

- Used to lift abstractions at higher-order (in conjunction with fixpoint approximation/transfer), see later.
THE END

My MIT web site is http://www.mit.edu/~cousot/
The course web site is http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/.

Course 16.399: "Abstract interpretation", Tuesday, April 12, 2005 — 165 — @ P. Cousot, 2005