« Mathematical foundations: (6) Abstraction — Part I »

Patrick Cousot

Jerome C. Hunsaker Visiting Professor
Massachusetts Institute of Technology
Department of Aeronautics and Astronautics

cousot@mit.edu
www.mit.edu/~cousot

Course 16.399: "Abstract interpretation"

http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/

THESE

présentée à

Université Scientifique et Médicale de Grenoble
Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR ES SCIENCES MATHEMATIQUES

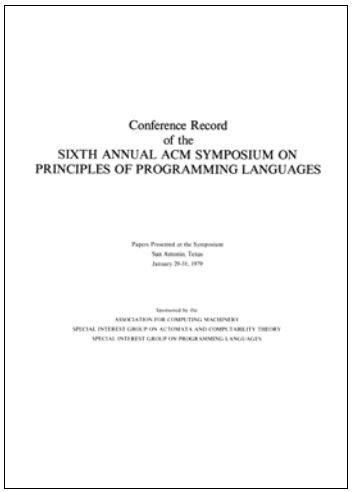
par

Patrick COUSOT

METHODES ITERATIVES DE CONSTRUCTION ET D'APPROXIMATION DE POINTS FIXES D'OPERATEURS MONOTONES SUR UN TREILLIS, ANALYSE SEMANTIQUE DES PROGRAMMES.

Thèse soutenue le 21 mars 1978 devant la Commission d'Examen :

Président : L. BOLLIET
Examinateurs : C. BENZAKEN
Ph. JORRAND
B. LORHO
C. PAIR
F. ROBERT
M. SINTZOFF



Conference Record
of the
SIXTH ANNUAL ACM SYMPOSIUM ON
PRINCIPLES OF PROGRAMMING LANGUAGES

Papers Presented at the Symposium
San Antonio, Texas
January 29-31, 1979

Sponsored by the
ASSOCIATION FOR COMPUTING MACHINERY
SPECIAL INTEREST GROUP ON AUTOMATA AND COMPUTABILITY THEORY
SPECIAL INTEREST GROUP ON PROGRAMMING LANGUAGES

Informal introduction to abstract interpretation

A little graphical language

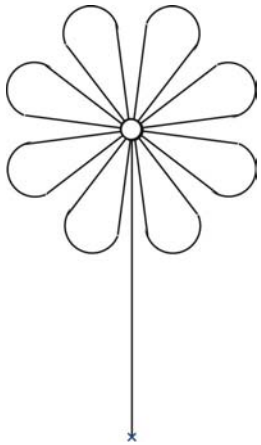
– objects;
– operations on objects.

Objects

An object is a pair:

- an origin (a reference point $\times$);
- a finite set of black pixels (on a white background).

Example of an object: a flower



Operations on objects : constants

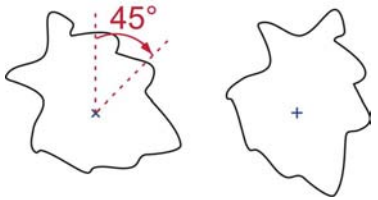
- constant objects;

  for example:

petal $=$ 

Operations on objects : rotation

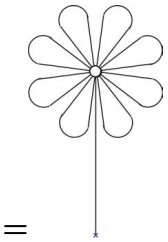
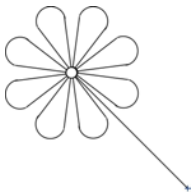
- rotation $\mathrm{r}[a](o)$ of objects o (of some angle a around the origin):



Example 1 of rotation

petal =  r[45](petal) = 

Example 2 of rotation

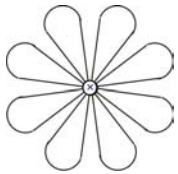
flower =  r[-45](flower) = 

Operations on objects : union

– union $o_1 \cup o_2$ of objects o_1 and o_2 = superposition at the origin;

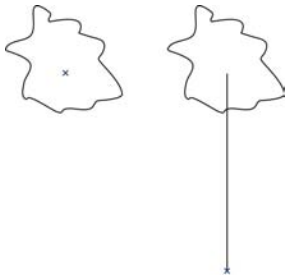
for example:

corolla = petal ∪ r[45](petal) ∪ r[90](petal) ∪ r[135](petal) ∪ r[180](petal) ∪ r[225](petal) ∪ r[270](petal) ∪ r[315](petal)



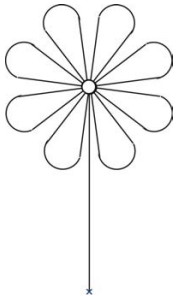
Operations on objects : add a stem

– stem(o) adds a stem to an object o (up to the origin, with new origin at the root);



Flower

flower = stem(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 degres:
    $$\text{r}[45](X) \subseteq X$$
– Or, equivalently [1]:
  $$F(X) \subseteq X, \quad \text{where} \quad F(X) = \text{petal} \cup \text{r}[45](X)$$

[1] By Tarski's fixpoint theorem, the least solution is $\text{lfp}^{\subseteq} F$.

Fixpoints

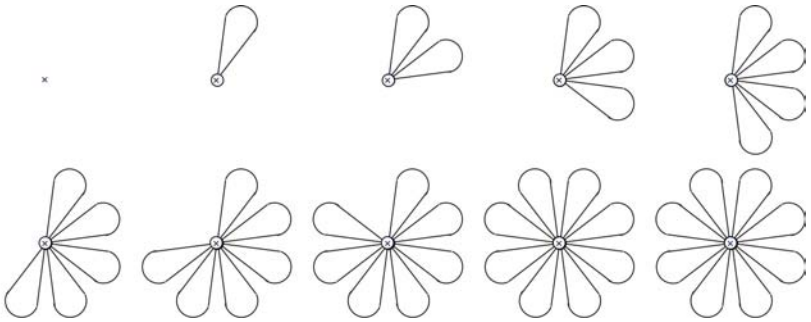
– corolla $= \text{lfp}^{\subseteq} F$
  $F(X) = \text{petal} \cup \text{r}[45](X)$

Iterates to fixpoints

– The iterates of F from the infimum $\emptyset$ are:

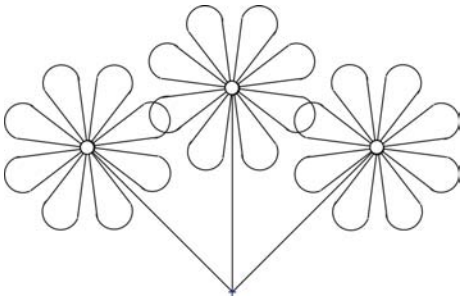
$$\begin{aligned} X^0 &= \emptyset\ , \\ X^1 &= F(X^0)\ , \\ &\cdots\cdots\cdots, \\ X^{n+1} &= F(X^n)\ , \\ &\cdots\cdots\cdots\ , \\ \text{lfp}^{\subseteq} F &= X^\omega = \bigcup_{n \geq 0} X^n\ . \end{aligned}$$

Iterates for the corolla



The bouquet

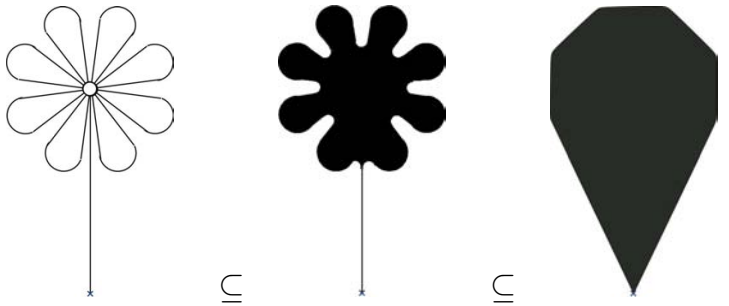
- bouquet = r[-45](flower) ∪ flower ∪ r[45](flower)
- The bouquet :



Upper-approximation

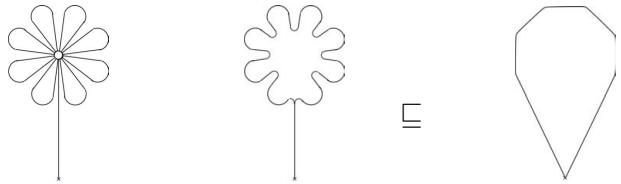
- An upper-approximation of an object is a object with:
  - same origin;
  - more pixels.

Examples of upper-approximations of flowers



Abstract objects

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



concrete object abstract object more abstract object

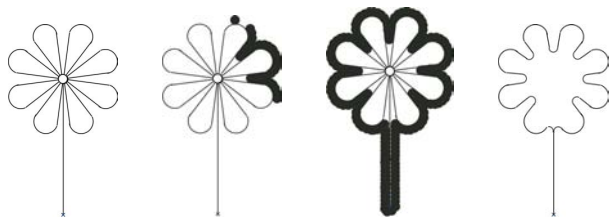
Abstract domain

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

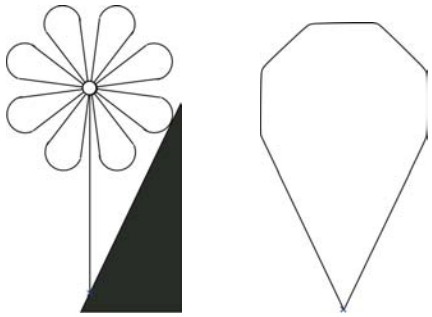
Abstraction

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

Example 1 of abstraction



Example 2 of abstraction



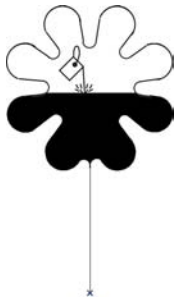
Comparing abstractions

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

Concretization

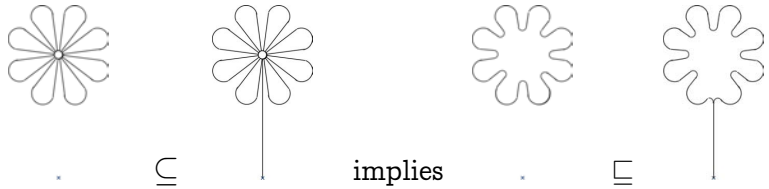
- a concretization function γ maps an abstract object $\overline{o}$ to the concrete object $\gamma(\overline{o})$ that is represents (that is to its concrete meaning/semantics).

Example of concretization



Galois connection 1/4

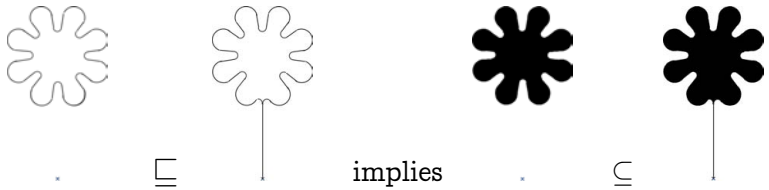
– α is monotonic.



$\subseteq$ implies $\sqsubseteq$

Galois connection 2/4

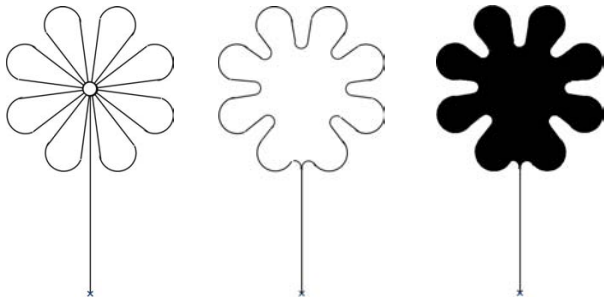
– γ is monotonic.



$\sqsubseteq$ implies $\subseteq$

Galois connection 3/4

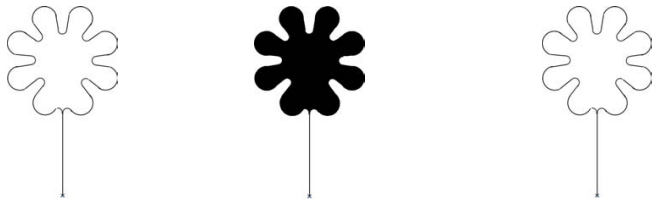
– for all concrete objects x, $\gamma \circ \alpha(x) \supseteq x$.



flower α(flower) $\gamma(\alpha$(flower))

Galois connection 4/4

– for all abstract objects y, $\alpha \circ \gamma(y) \sqsubseteq y$.



abstract flower γ(abstract flower) $\alpha(\gamma$(abstract flower))

Galois connections

$$\langle \mathcal{D}, \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{\mathcal{D}}, \sqsubseteq \rangle$$

iff $\quad \forall x, y \in \mathcal{D} : x \subseteq y \Longrightarrow \alpha(x) \sqsubseteq \alpha(y)$

$\wedge\ \forall \overline{x}, \overline{y} \in \overline{\mathcal{D}} : \overline{x} \sqsubseteq \overline{y} \Longrightarrow \gamma(\overline{x}) \subseteq \gamma(\overline{y})$

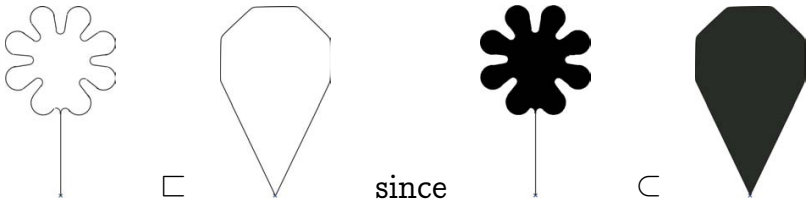
$\wedge\ \forall x \in \mathcal{D} : x \subseteq \gamma(\alpha(x))$

$\wedge\ \forall \overline{y} \in \overline{\mathcal{D}} : \alpha(\gamma(\overline{y})) \sqsubseteq \overline{x}$

iff $\quad \forall x \in \mathcal{D}, \overline{y} \in \overline{\mathcal{D}} : \alpha(x) \sqsubseteq y \Longleftrightarrow x \subseteq \gamma(y)$

Abstract ordering

– $x \sqsubseteq y$ is defined as $\gamma(x) \subseteq \gamma(y)$.



Specification of abstract operations

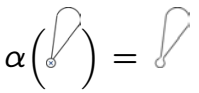
– $\overline{\mathrm{op}/0} \stackrel{\text{def}}{=} \alpha(\mathrm{op}/0)$ 0-ary

– $\overline{\mathrm{op}/1}(y) \stackrel{\text{def}}{=} \alpha(\mathrm{op}/1(\gamma(y)))$ unary

– $\overline{\mathrm{op}/2}(y, z) \stackrel{\text{def}}{=} \alpha(\mathrm{op}/2(\gamma(y), \gamma(z)))$ binary

– ...

Abstract petal

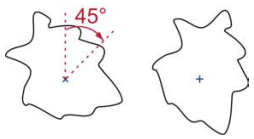


Abstract rotations

– $\overline{\mathrm{r}}[a](y) = \alpha(\mathrm{r}[a](\gamma(y)))$

Abstract rotations

– $\overline{\mathrm{r}}[a](y) = \alpha(\mathrm{r}[a](\gamma(y)))$
$= \mathrm{r}[a](y)$

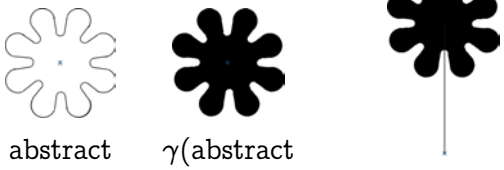


A commutation theorem on abstract rotations

– $\alpha(\mathrm{r}[a](x))$
$= \alpha(\gamma(\alpha(\mathrm{r}[a](x))))$
$= \alpha(\gamma(\mathrm{r}[a](\alpha(x))))$
$= \alpha(\mathrm{r}[a](\gamma(\alpha(x))))$
$= \overline{\mathrm{r}}[a](\alpha(x))$

Abstract stems

– $\overline{\mathrm{stem}}(y) = \alpha(\mathrm{stem}(\gamma(y)))$



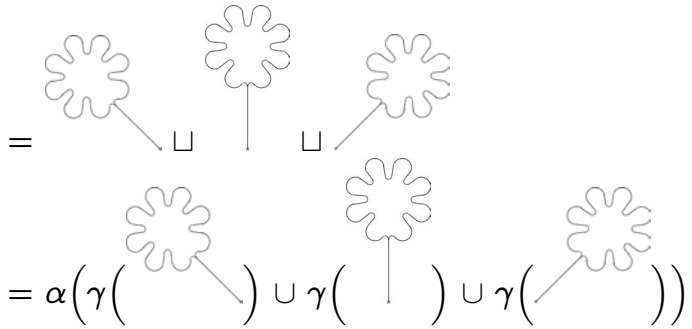
abstract corolla | γ(abstract corolla) | stem(γ(abstract corolla)) | α(stem(γ(abstract corolla)))

Abstract union

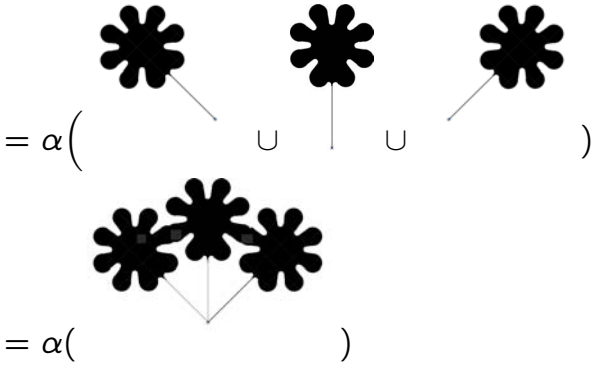
– $x \sqcup y = \alpha(\gamma(x) \cup \gamma(y))$

Abstract bouquet:

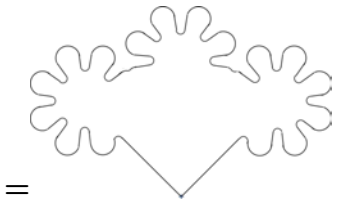
abstract bouquet



Abstract bouquet: (cont'd)



Abstract bouquet: (end)



A theorem on the abstract bouquet

abstract flower $= \alpha$(concrete flower)

abstract bouquet

$= \overline{\mathrm{r}}$[-45](abstract flower) $\sqcup$ abstract flower $\sqcup \overline{\mathrm{r}}$[-45](abstract flower)

$= \overline{\mathrm{r}}$[-45](α(concrete flower)) $\sqcup \alpha$(concrete flower) $\sqcup \overline{\mathrm{r}}$[-45]($\alpha$(concrete flower))

$= \alpha$(r[-45](concrete flower)) $\sqcup \alpha$(concrete flower) $\sqcup \alpha$(r[-45](concrete flower))

$= \alpha$(r[-45](concrete flower) $\cup$ concrete flower $\cup$ r[-45](concrete flower))

$= \alpha$(concrete bouquet)

Abstract fixpoint

– abstract corolla $= \alpha$(concrete corolla) $= \alpha(\mathrm{lfp}^{\subseteq} F)$

where $F(X) =$ petal $\cup$ r[45](X))

Abstract transformer $\overline{F}$

– $\alpha(F(X))$

$= \alpha$(petal $\cup$ r[45](X))

$= \alpha$(petal) $\sqcup \alpha$(r[45](X))

$= \alpha$(petal) $\sqcup \overline{\mathrm{r}}$[45]($\alpha(X)$)

$=$ abstract petal $\sqcup \overline{\mathrm{r}}$[45]($\alpha(X)$)

$= \overline{F}(\alpha(X))$

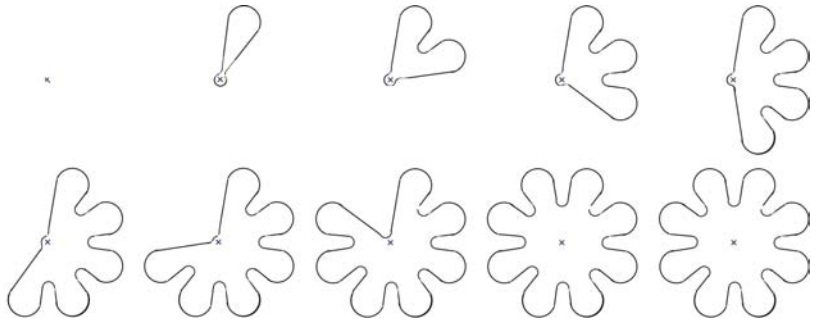
by defining

$$\overline{F}(X) = \text{abstract petal} \sqcup \overline{\mathrm{r}}[45](X)$$

and so:

– abstract corolla $= \alpha$(concrete corolla) $= \alpha(\mathrm{lfp}^{\subseteq} F) = \mathrm{lfp}^{\sqsubseteq} \overline{F}$

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

– 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?

On abstracting properties of graphic objects

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

$$\langle \wp(\wp(\mathcal{P})), \subseteq \rangle \underset{\alpha_0}{\overset{\gamma_0}{\leftrightarrows}} \langle \wp(\mathcal{P}), \subseteq \rangle$$

where:
$\mathcal{P}$ is a set of pixels (e.g. pairs of coordinates)
$\alpha_0(X) = \bigcup X$
$\gamma_0(Y) = \{G \in \mathcal{P} \mid G \subseteq Y\}$

Is it for fun (only)?

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

> J. Serra. Morphological filtering: An overview, Signal Processing 38 (1994) 3–11.

It can be entirely formalized by abstract interpretation.

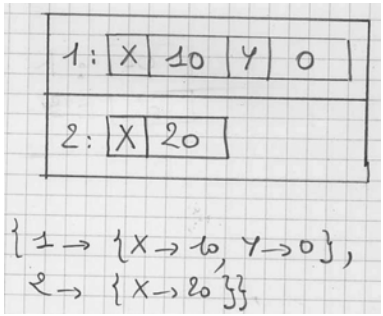
Property abstraction

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: $\mathbb{X}$
- Environments: $\mathbb{X} \mapsto \mathcal{V}$
- Stacks: assigning values to variables in the context of block-structured languages: $\bigcup_{n \geq 0} ([1, n] \mapsto (\mathbb{X} \nrightarrow \mathcal{V}))$


```
begin
   new X,Y;
   X := 10; Y := 0;
   begin
      new X;
  X := 20;
  ...
   end;
   ...
end;
```



- 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: $\{z \in \mathbb{Z} \mid \text{minint} \leq z \leq \text{maxint}\} \cup \{\Omega_m \mid m \in \mathcal{M}\}$ where $\mathcal{M}$ is a set of error messages

- equality of two variables X and Y: $\{\rho \in \mathbb{X} \mapsto\!\!\!\!\to \mathcal{V} \mid \mathtt{X}, \mathtt{Y} \in \text{dom}(\rho) \wedge \rho(\mathtt{X}) = \rho(\mathtt{Y})\}$
- invariance property (of a program with states in Σ): $I \in \wp(\Sigma)$
- trace property: $T \in \wp(\Sigma^{\vec{\infty}})$
- trace semantics property: $P \in \wp(\wp(\Sigma^{\vec{\infty}}))$
- ...

The complete lattice of concrete properties

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

$$\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$$

where

- A property $P \in \wp(\Sigma)$ 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 ($o \in P \Longrightarrow o \in Q$)

- $\emptyset$ is *false* (ff)
- Σ is *true* (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 replace someting "concrete" [2] 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 [3]

[2] real, actual, material, corporeal, ...

[3] since otherwise this property would have to be "exactly that object" i.e. $\{o\}$

- So an abstraction of properties in $\wp(\Sigma)$ of objects in Σ is essentially a subset $A \subseteq \wp(\Sigma)$ such that:
  - The properties in A are the concrete properties that can be described exactly by the abstraction, without any loss of information
  - The properties in $\wp(\Sigma) \setminus A$ are the properties that cannot be described exactly by the abstraction, and have to be referred to by being approximated in some way or another by abstract properties in A

Intuitive example 1 of abstraction

Cars $\xrightarrow{\alpha}$ Trademarks [4]

- 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

[4] Formally, if $t \in \text{Cars} \mapsto \text{Trademarks}$ yield the trademark $t(c)$ of a car $c \in \text{Cars}$ then the abstraction of $P \in \wp(\text{Cars})$ is $\alpha(P) = \{t(c) \mid c \in P\}$ and the set of cars described by an abstract property $T \subseteq \text{Trademarks}$ is $\gamma(T) = \{c \in \text{Cars} \mid t(c) \in T\}$.

Intuitive example 2 of abstraction

Scientific papers $\xrightarrow{\alpha}$ set of keywords [5]

- 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

[5] Formally if $w \in$ Scientific papers $\mapsto$ Words provides the set of words $w(p)$ appearing in a paper $p \in$ Scientific papers and Keywords $\mapsto$ Words is a given list of keywords, then a property $P \in \wp(\text{Scientific papers})$ is abstracted by the set of keywords $\alpha(P) = \{w(p) \cap \text{Keywords} \mid p \in P\}$ and a set K of keyworks stands for the concrete property $\gamma(K) = \{p \in \text{Scientific papers} \mid K \subseteq w(p) \cap \text{Keywords}\}$.

Abstraction, definition of concrete and abstract properties

Abstraction in a reasoning/computation such that:

– Only some properties $A \subseteq \wp(\Sigma)$ of the objects in Σ can be used;

– The properties $P \in A$ that can be used are called *abstract*;

– The properties $P \in \wp(\Sigma)$ are called *concrete*;

– 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 $P \in \wp(\Sigma) \setminus A$ 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*.

Direction of abstraction

– When approximating a concrete property $P \in \wp(\Sigma)$, 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 estabblish that

"$\overline{P} \in A$ is an approximation/abstraction of $P \in \wp(\Sigma)$"

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

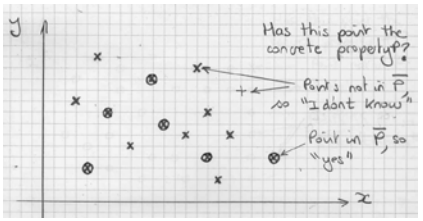
– We consider essentially two cases:
  - Approximation from above: $P \subseteq \overline{P}$
  - Approximation from below: $P \supseteq \overline{P}$

– 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

Abstraction from below

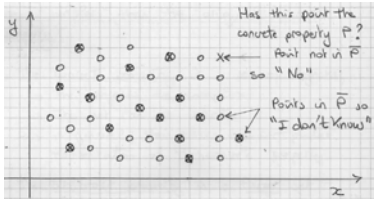


×: points which have the concrete property P

○: points which have the abstract property $\overline{P}$

– To answer the question "$\langle x, y\rangle \in P$?" using only $\overline{P}$ (such that $P \subseteq \overline{P}$):
  - If $\langle x, y\rangle \notin \overline{P}$ then "I don't know"
  - If $\langle x, y\rangle \in \overline{P}$ then "Yes"

Abstraction from above



×: points which have the concrete property P

○: points which have the abstract property $\overline{P}$

– To answer the question "$\langle x, y\rangle \in P$?" using only $\overline{P}$ (such that $P \supseteq \overline{P}$):
  - If $\langle x, y\rangle t \in \overline{P}$ then "I don't know"
  - If $\langle x, y\rangle t \notin \overline{P}$ then "No"

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

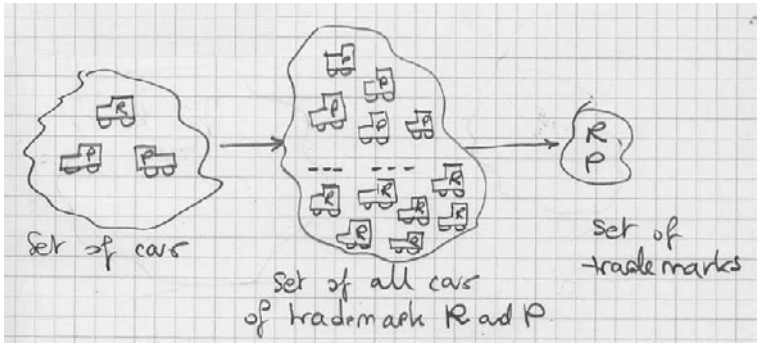
– The concrete property is a set of objects
  → The objects are complex
  → The set can be infinite
  → In general their exists no suitable computer representation of the concrete property

– 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 Σ, 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?

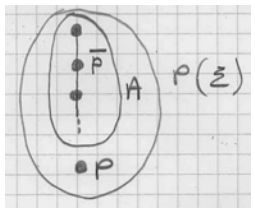
- block?
- ask for help?
- fail?
- *answer something sensible*!

- 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. $\Sigma \in A$, Σ 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

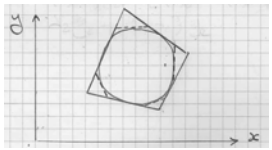


- If a concrete property $P \in \wp(\Sigma)$ has minimal upper approximations $\overline{P} \in A$:
  - $P \subseteq \overline{P}$
  - $\nexists \overline{P}' : P \subseteq \overline{P}' \sqsubset \overline{P}$

  then such minimal approximations are more precise than the non-minimal ones
- So minimal abstract upper approximations, if any, should be prefered
- 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]
  - $\Sigma = \mathbb{R} \times \mathbb{R}$
  - $A =$ convex polyhedra
  - Absence of minimal approximation is shown by Euclide's construction:



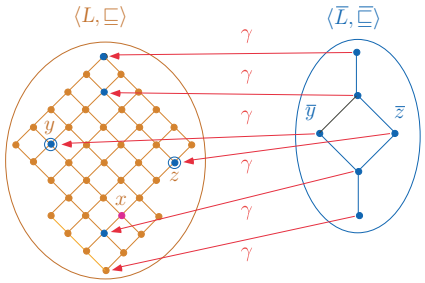
– In absence of minimal approximations, the approximation $P \subset P_1$ can always be approximated by a better one $P \subset P_2 \subset P_1$!
– Some arbitrary choice has to be performed. This case will be studied later (see [2]). So, in the following, we assume the existence of minimal approximations

References

[1] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, NY, U.S.A.

[2] P. Cousot and R. Cousot. Abstract Interpretation Frameworks. *Journal of Logic and Computation*, 2(4):511–547, August 1992.

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 γ cannot be the upper adjoint of a Galois connection since it is not a complete meet morphism: $\gamma(\overline{y}) \wedge \gamma(\overline{z}) \neq \gamma(\overline{y} \sqcap \overline{z})$
– Abstraction in absence of best approximation is studied in [2]

Which minimal abstraction to choose?

– If there are several minimal possible abstract approximations $\overline{P}_1$, $\overline{P}_2$, ... [6] 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 chose '+', because of the rule '+' + '+' = '+', while '+' + '-' yields no information (I don't know)
  - In "(-1)+0", it is better to chose '-' because of the rules (- '+') = '-' and '-' + '-' = '-', while '-' + '+' yields no information (I don't know)
  - Both cases have to be tried (backtracking)

[6] There can even be infinitely many ones.

– In absence of unicity of the minimal approximation, it may be necessary to try all of them (at the cost of an exponential blow up of the mechanical reasoning). This case will be studied later (see [2]).
– To start with, we will assume the existence of a *best approximation* (i.e. a unique minimal upper approximation).

Best abstraction

– A very handy choice of the abstract properties $A \subseteq \wp(\Sigma)$ is when every concrete property P has a best approximation $\overline{P} \in P$:
  - $P \subseteq \overline{P}$
  - $\forall \overline{P}' \in A : (P \subseteq \overline{P}') \Longrightarrow (\overline{P} \subseteq \overline{P}')$
– It follows that $\overline{P}$ is the glb of the over-approximations of P in A:

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

PROOF. – We have $\forall \overline{P} \in \{\overline{P}' \in A \mid P \subseteq \overline{P}'\} : P \subseteq \overline{P}$ so $P \subseteq \bigcap\{\overline{P}' \in A \mid P \subseteq \overline{P}'\}$ by def. glb $\sqcap$
– Moreover $\forall \overline{P}' \in A : (P \subseteq \overline{P}') \Longrightarrow (\bigcap\{\overline{P}'' \in A \mid P \subseteq \overline{P}''\} \subseteq \overline{P}')$
– It follows that $\overline{P} = \bigcap\{\overline{P}' \in A \mid P \subseteq \overline{P}'\}$
– So $[\exists \overline{P} : (P \subseteq \overline{P}) \wedge (\forall \overline{P}' \in A : (p \subseteq \overline{P}') \Longrightarrow (\overline{P} \subseteq \overline{P}')] \iff [\overline{P} = \bigcap\{\overline{P}' \in A \mid P \subseteq \overline{P}'\} \in A$

□

The abstract domain is a Moore family

THEOREM. The hypothesis that any concrete property $P \in \wp(\Sigma)$ has a *best abstraction* $\overline{P} \in A$, implies that

The abstract domain A is a Moore family.

■

PROOF. – Let $X \subseteq A$ be a set of apstract properties. Its intersection $\bigcap X$ has a best approximation $\overline{P} \in A$. We have therefore

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

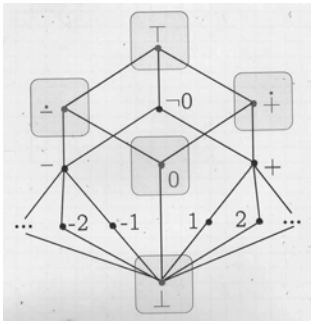
But $\forall \overline{P}' \in X : \bigcap X \subseteq \overline{P}'$ and $X \subseteq A$ so $X \subseteq \{\overline{P}' \in A \mid \bigcap X \subseteq \overline{P}'\}$ and therefore $\bigcap\{\overline{P}' \in A \mid \bigcap X \subseteq \overline{P}'\} \subseteq \bigcap X$ by def. of glb. By antisymmetry, $\bigcap X = \bigcap\{\overline{P}' \in A \mid \bigcap X \subseteq \overline{P}'\} = \overline{P} \in A$, proving A to be a Moore family.

□

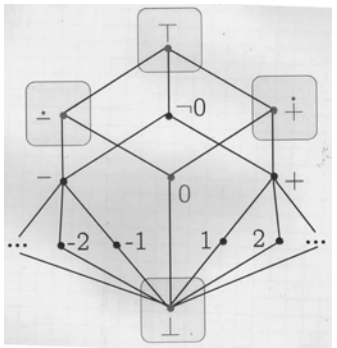
In particular $\bigcap \emptyset = \Sigma \in A$, which is consistent with our hypothesis that A should contain Σ to have the ability to express "I don't know".

Example and counter-example of Moore family based abstraction

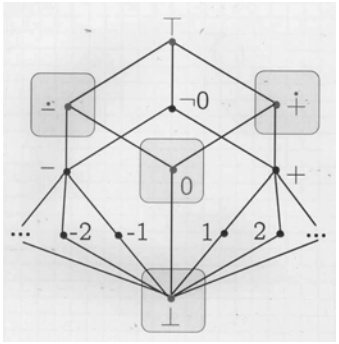
– Example: rule of signs with best approximation of 0



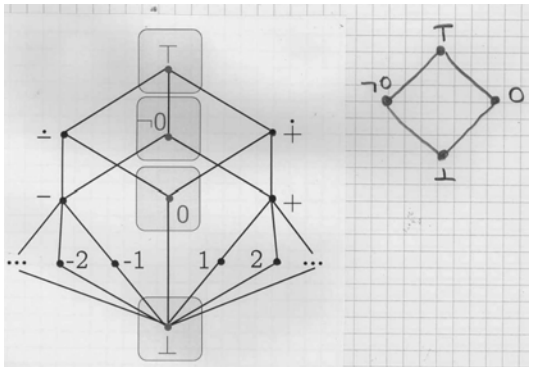
– Counter-example: rule of signs without best approximation of 0



– Counter example: rule of sign without upper approximation of "different from zero"



– Example: abstraction to 0 or different from 0



Closure operator based abstraction

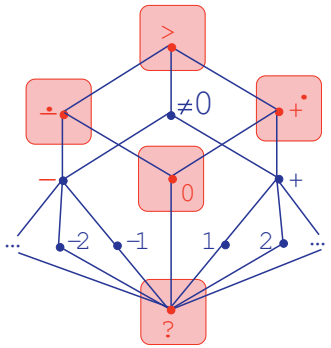
Assume that the abstract domain A is a Moore family of the concrete domain $\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$. Then the abstraction map is

$$\rho \in \wp(\Sigma) \mapsto A$$
$$\rho(P) \stackrel{\text{def}}{=} \bigcap \{ \overline{P} \in A \mid P \subseteq \overline{P} \}$$

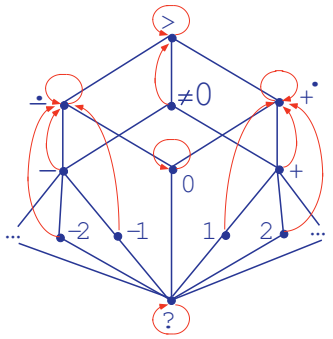
Then ρ is an upper closure operator on $\wp(\Sigma)$.

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

Example of abstraction map



Moore family



Abstraction map
(closure operator)

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 ρ

PROOF. – Given $\mathcal{M}$ defined $\rho(P) = \bigcap\{\overline{P} \in \mathcal{M} \mid P \subseteq \overline{P}\} \in \mathcal{M}\}$ so that $A = \mathcal{M} = \rho(\wp(\Sigma))$

– Conversely, given a closure operator ρ, define $A = \rho(\wp(\Sigma)) = \{\rho(P) \mid P \in \wp(\Sigma)\}$ which is therefore the set of fixpoints of ρ whence a Moore family since ρ operates on a complete lattice

□

Examples of specifications of an abstractionby a Moore family and a closure operator

– The most imprecise abstraction is "I don't know"
  - $\mathcal{M} = \{\Sigma\}$
  - $\rho = \lambda P \cdot \Sigma$
– The most precise abstraction is "identity"
  - $\mathcal{M} = \wp(\Sigma)$
  - $\rho = \lambda P \cdot P$

Generalizing to complete lattices

– The reasoning on abstractions of concrete properties $\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$ 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 $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$
– This allow a compositional approach where $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is abstracted to $\langle A_1, \sqsubseteq_1, \bot_1, \top_1, \sqcup_1, \sqcap_1 \rangle$ which itself can be further abstracted to $\langle A_2, \sqsubseteq_2, \bot_2, \top_2, \sqcup_2, \sqcap_2 \rangle$, ...

Correspondance between concrete and abstract properties

Given a closure operator ρ on a poset $\langle L, \sqsubseteq \rangle$ (typically L is $\wp(\Sigma)$), 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 ($1_L \stackrel{\text{def}}{=} \lambda x \in L \cdot x$):

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

PROOF. We must prove $\forall x \in L : \forall y \in \rho(L) : (\rho(x) \sqsubseteq y) \iff (x \sqsubseteq 1_L(y))$. We have $y \in \rho(L)$ iff $\exists z \in L : \rho(z) = y$ so that this condition is equivalent to $\forall x, z \in L : (\rho(x) \sqsubseteq \rho(z)) \iff (x \sqsubseteq \rho(z))$ which directly follows from Morgado's theorem. Moreover, ρ is surjective on $\rho(L)$. □

Correspondance between concrete and representations of abstract properties

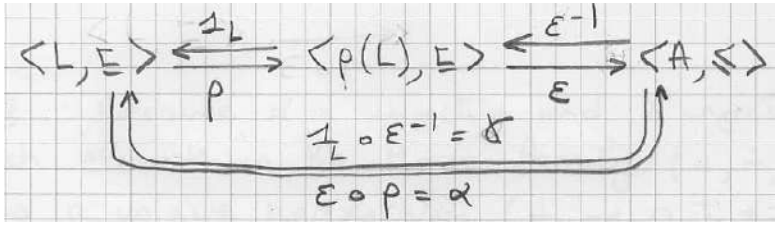
– Let $\langle A, \leq \rangle$ be an order-isomorphic representation of the abstract domain $\langle \rho(L), \sqsubseteq \rangle$. We have

$$\langle \rho(L), \sqsubseteq \rangle \underset{\epsilon^{-1}}{\overset{\epsilon}{\leftrightarrows}} \langle A, \leq \rangle$$

where ϵ^{-1} is the inverse of the bijection $\epsilon \in \rho(L) \rightarrowtail\!\!\!\!\!\rightarrow A$ and $\epsilon \in \rho(L) \stackrel{\text{m}}{\longmapsto} A$.

By composition, we get:

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



Specification of an abstract domain by a Galois surjection

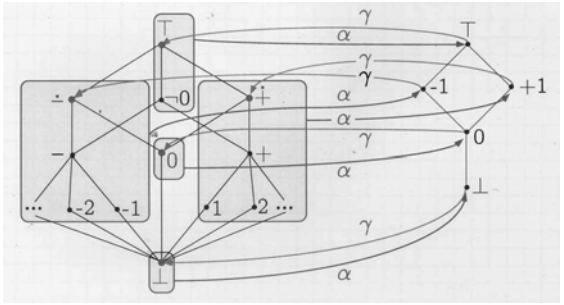
– Inversely, we can consider a Galois surjection

$$\langle L, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq \rangle$$

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

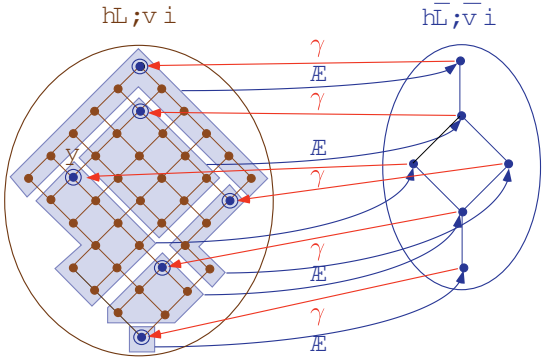
– We have an order-isomorphic representation of the abstract domain $\langle \rho(L), \sqsubseteq \rangle$, which is a Moore family.

Specification of an abstract domain by a Galois surjection, example



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

Galois Connection $\langle L, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \overline{L}, \overline{\sqsubseteq} \rangle$

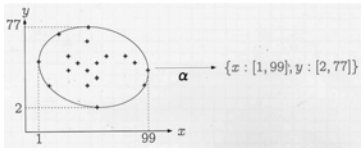


– ◉: Moore family of best approximations;

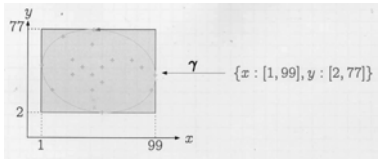
– ▭: concrete values with the same abstraction.

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

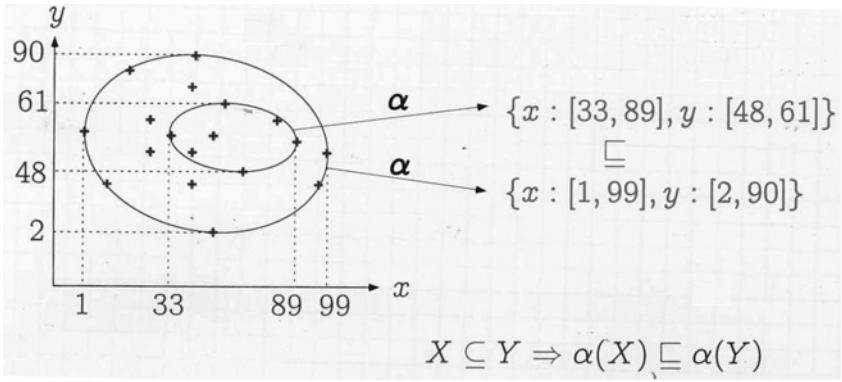
– Abstraction of a set of point in $\mathbb{R}^2$ by an interval:



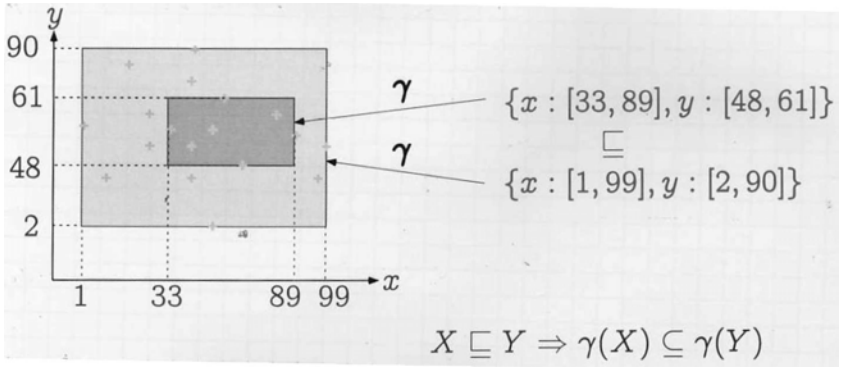
– Concretization:



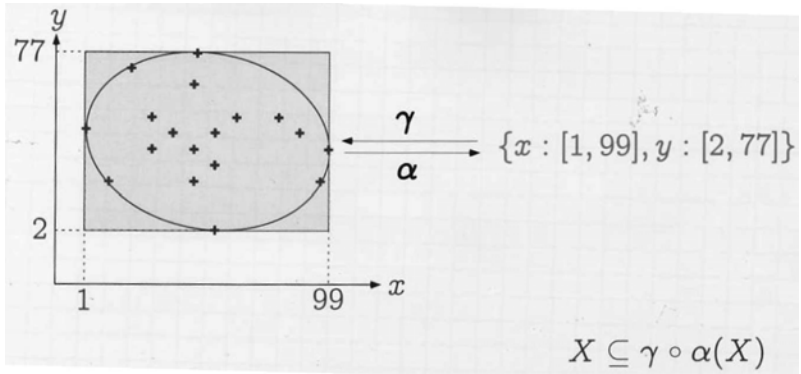
– The abstraction α is monotone:



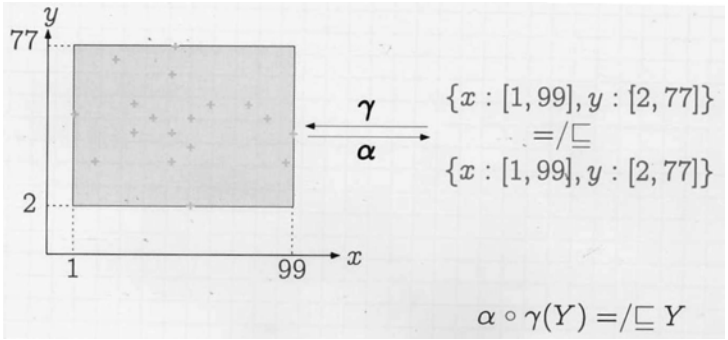
– The concretization γ is monotone:



– $\gamma \circ \alpha$ is extensive (indeed an upper-closure operator):



– The composition $\alpha \circ \gamma$ is:
  - The identity for Galois surjections
  - Reductive (ideed a lower-closure operator) for Galois connections [7]



[7] providing the least abstract properties with similar expressive power that is same concretization.

– The intuition of $\sqsubseteq$ is that $\overline{P} \sqsubseteq \overline{P}'$ implies $\gamma(\overline{P}) \subseteq \gamma(\overline{P}')$ so that $\overline{P}$ is more precise than $\overline{P}'$ when expressed in the concrete

– So $\alpha \circ \gamma(\overline{P}) \sqsubseteq \overline{P}$ means that concretization can loose no information, since if the concrete property P is overapproximated by $\overline{P}$ then

$$\begin{array}{rl} & P \subseteq \gamma(\overline{P}) \\ \iff & P \subseteq \gamma(\alpha \circ \gamma(\overline{P})) \end{array}$$

so that using $\overline{P}$ or $\alpha \circ \gamma(\overline{P})$ 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 $\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$ where objects in Σ represent program computations and the elements of $\wp(\Sigma)$ 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 lattiec 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.

Reference

[3] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY, U.S.A.

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

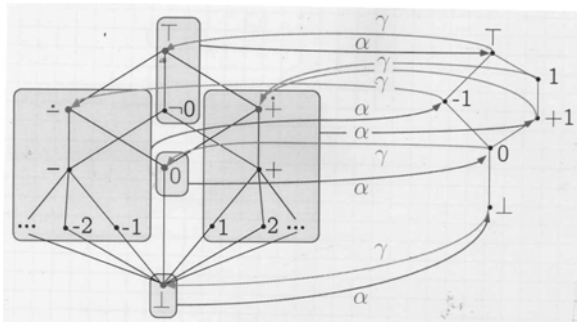
- Assume the correspondence between concrete and abstract properties is a non-surjective Galois connection:

$$\langle L, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq \rangle$$

- γ is not surjective, which means that at least two different abstract properties $\overline{P}_1$ and $\overline{P}_2$ have exactly the same concretization:

$$\overline{P}_1 \neq \overline{P}_2 \wedge \gamma(\overline{P}_1) = \gamma(\overline{P}_2)$$

Example of non-surjective Galois connection based abstraction



Here "1" and "+1" are two different encodings of the same concrete property $\dot{+}$ (i.e; positive or zero).

Reduction

- With non-surjective Galois connections $\langle L, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq \rangle$, 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 $\langle L, \sqsubseteq \rangle \underset{\alpha_\equiv}{\overset{\gamma_\equiv}{\leftrightarrows}} \langle A_\equiv, \leq_\equiv \rangle$ where $\overline{P} \equiv \overline{P}' \iff \gamma(\overline{P}) = \gamma(\overline{P}')$, $\alpha_\equiv(P) = [\alpha(P)]_\equiv$, $\gamma([\overline{P}]_\equiv) = \gamma(\overline{P})$ and $[\overline{P}]_\equiv \leq_\equiv [\overline{P}']_\equiv \iff \overline{P} \leq \overline{P}'$

- Example:
  - Abstract properties are intervals $[a, b]$ meaning $\gamma([a, b]) \stackrel{\text{def}}{=} \{x \mid \text{minint} \leq a \leq x \leq b \leq \text{maxint}\}$
  - The empty set is represented by any $[a, b]$ with $b < a$. This can be left as is or normalized as e.g. $[\text{maxint}, \text{minint}]$
  - The supremum is represented by any $[a, b]$ with $a \leq \text{minint}$ and $\text{maxint} \leq b$. This can be left as is or better normalized as e.g. $[\text{minint}, \text{maxint}]$
- 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
- $\alpha_A(X) \stackrel{\text{def}}{=} X \cap A$
- $\gamma_A(Y) \stackrel{\text{def}}{=} (Y \cup \neg A)$ where $\neg A \stackrel{\text{def}}{=} C \setminus A$

then

$$\langle \wp(C), \subseteq \rangle \underset{\alpha_A}{\overset{\gamma_A}{\leftrightarrows}} \langle \wp(A), \subseteq \rangle$$

PROOF. $\alpha_A(X) \subseteq Y \iff (X \cap A) \subseteq Y \iff X \subseteq (Y \cup \neg A) \iff X \subseteq \gamma_A(Y)$.
If $Y \in \wp(A)$ then $\alpha(A) = A$ proving α to be onto. □

– 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
– $\alpha_S(X) \stackrel{\text{def}}{=} (X \subseteq S)$
– $\gamma_S(b) \stackrel{\text{def}}{=} (\!| b \,?\, S : C |\!)$
then

$$\langle \wp(C), \subseteq \rangle \underset{\alpha_S}{\overset{\gamma_S}{\leftrightarrows}} \langle \mathbb{B}, \Leftrightarrow \rangle$$

PROOF. $\alpha_S(X) \Leftrightarrow b \iff b \Rightarrow (X \subseteq S) \iff (\!| b \,?\, (X \subseteq S) : \mathtt{tt} |\!) \iff (\!| b \,?\, (X \subseteq S) : (X \subseteq C) |\!) \iff X \subseteq (\!| b \,?\, S : C |\!) \iff X \subseteq \gamma_S(b)$. α_S is onto since $\alpha_S(S) = \mathtt{tt}$ and $\alpha_S(C) = (C \subseteq S) = \mathtt{ff}$ since $S \subset C$. □

– The intuition is to record whether a given property holds ($\mathtt{tt}$) or whether it is not known whether it holds or not ($\mathtt{ff}$).
– Example 1: this is called "predicate abstraction" (to be further developped later)
– Example 2: $\Sigma = \langle c, m \rangle$, c is a control point, m is a memory state, $S = \{\langle c, m \rangle \mid m(X) = 0\}$ records whether variable X is null.
– Example 3: green mark on tags of vegetarian participants in CS conferences

Elementwise/homomorphic abstraction

If
– $@ : C \mapsto A$ elementwise abstraction
– $\alpha_@(P) \stackrel{\text{def}}{=} \{@(p) \mid p \in P\}$ set abstraction
– $\gamma_@(Q) \stackrel{\text{def}}{=} \{p \mid @(p) \in Q\}$ concretization
then

$$\langle \wp(C), \subseteq \rangle \underset{\alpha_@}{\overset{\gamma_@}{\leftrightarrows}} \langle \wp(A), \subseteq \rangle$$

PROOF. Given $X \subseteq C$ and $Y \subseteq A$, we have

$$\begin{array}{ll} & \alpha_@(X) \subseteq Y \\ \iff & \{@(x) \mid x \in X\} \subseteq Y & \text{〖def. } \alpha_@\text{〗} \\ \iff & \forall x \in X : @(x) \in Y & \text{〖def./ } \subseteq\text{〗} \end{array}$$

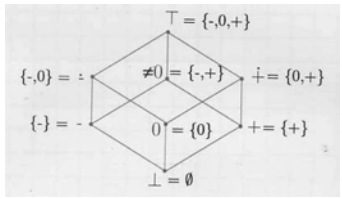
$\iff X \subseteq \{x \in C \mid @(x) \in Y\}$ ⟨def./ $\subseteq$⟩

$\iff X \subseteq \gamma_@(Q)\}$ ⟨def./ $\gamma_@$⟩

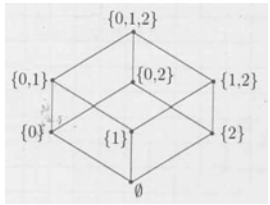
□

– The intuition is to remember only one characteristics $@(x)$ 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
  - $@(c)$: trademark of car c
  - $\alpha_@(X)$: trademarks of cars in set X
  - $\gamma_@(T)$: all cars which trademark is in T

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



– Example 3: congruence
  - $@ \in \mathbb{Z} \mapsto \mathbb{Z}/n$, $n = 3$
  - $@(x) \stackrel{\text{def}}{=} |x| \bmod n$



– $\langle \wp(C), \subseteq \rangle \underset{\alpha_@}{\overset{\gamma_@}{\leftrightarrows}} \langle \wp(A), \subseteq \rangle$ when $@ : C \mapsto\!\!\!\!\to A$ is onto
– Often rediscovered (e.g. abstract model-checking)
– Often misunderstood (e.g.: $\langle @, \gamma \rangle$ instead of $\langle \alpha, \gamma \rangle$)

Functional abstraction

If
– $D^\natural$ and $D^\sharp$ are sets
– $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice
– $@ : D^\natural \mapsto D^\sharp$
– $\alpha \in (D^\natural \mapsto L) \mapsto (D^\sharp \mapsto L)$
  $\alpha(f) \stackrel{\text{def}}{=} \lambda y \cdot \bigsqcup \{f(x) \mid @(x) = y\}$
– $\gamma \in (D^\sharp \mapsto L) \mapsto (D^\natural \mapsto L)$
  $\gamma(g) \stackrel{\text{def}}{=} g \circ @$

then

$$\langle D^\natural \mapsto L, \dot{\sqsubseteq} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D^\sharp \mapsto L, \dot{\sqsubseteq} \rangle$$

PROOF. $\alpha(f) \dot{\sqsubseteq} g \iff \forall y : \alpha(f)(y) \sqsubseteq g(y) \iff \forall y : \bigsqcup\{f(x) \mid @(x) = y\} \sqsubseteq g(y) \iff \forall y : \forall x : (@(x) = y) \Longrightarrow (f(x) \sqsubseteq g(y)) \iff \forall x : f(x) \sqsubseteq g(@(x)) \iff f \dot{\sqsubseteq} g \circ @ \iff f \dot{\sqsubseteq} \gamma(g)$ □

– Example: this is a generalization of the previous homomorphic abstraction with $L = \{\mathrm{ff}, \mathrm{tt}\}$ ordered with $\mathrm{ff} \sqsubset \mathrm{tt}$ using the representation of sets by their characteristic function.

Relational Abstraction

If

– $@ \in D^\natural \times D^\sharp$

– $\alpha \in \wp(D^\natural) \mapsto \wp(D^\sharp)$
$\alpha(X) \stackrel{\text{def}}{=} \{y \mid \exists x \in X : \langle x, y\rangle \in @\} = \mathrm{post}[@]X$

– $\gamma \in \wp(D^\sharp) \mapsto \wp(D^\natural)$
$\gamma(Y) \stackrel{\text{def}}{=} \{x \mid \forall y : (\langle x, y\rangle \in @) \Longrightarrow (y \in Y)\} = \widetilde{\mathrm{pre}}[@]Y$

then

$$\langle \wp(D^\natural), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(D^\sharp), \subseteq\rangle$$

PROOF. $\alpha(X) \subseteq Y \iff \{y \mid \exists x \in X : \langle x, y\rangle \in @\} \subseteq Y \iff \forall y : (\exists x \in X : \langle x, y\rangle \in @) \Longrightarrow (y \in Y) \iff \forall y : \forall x \in X : (\langle x, y\rangle \in @) \Longrightarrow (y \in Y) \iff \forall x \in X : \forall y : (\langle x, y\rangle \in @) \Longrightarrow (y \in Y) \iff X \subseteq \{x \mid \forall y : (\langle x, y\rangle \in @) \Longrightarrow (y \in Y)\} \iff X \subseteq \gamma(Y)$. □

– A generalization of the homomorphic/elementwise abstraction using a relation instead of a function @.

Relational lattice abstraction

If

– $\langle A, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ Abstract domain (complete lattice)

– $\rho \in \wp(C \times A)$ Abstraction relation

– $\alpha_\rho(P) \stackrel{\text{def}}{=} \bigsqcup\{a \in A \mid \exists p \in P : \langle p, a\rangle \in \rho\}$
Set abstraction

– $\gamma_\rho(a) \stackrel{\text{def}}{=} \{p \mid \forall a' \in A : (\langle p, a'\rangle \in \rho) \Longrightarrow (a' \sqsubseteq a)\}$
Concretization

then

$$\langle \wp(C), \subseteq\rangle \underset{\alpha_\rho}{\overset{\gamma_\rho}{\leftrightarrows}} \langle A, \sqsubseteq\rangle$$

PROOF. $\forall P \subseteq C, \forall Y \in A$:

$$
\begin{array}{lll}
 & \alpha_\rho(P) \sqsubseteq Y & \\
\iff & \bigsqcup\{a \in A \mid \exists p \in P : \langle p, a\rangle \in \rho\} \sqsubseteq Y & \text{(def. } \alpha\text{)} \\
\iff & \forall a \in A : (\exists p \in P : \langle p, a\rangle \in \rho) \Longrightarrow (a \sqsubseteq Y) & \text{(def. } \bigsqcup\text{)} \\
\iff & \forall a \in A : \forall p \in P : (\langle p, a\rangle \in \rho) \Longrightarrow (a \sqsubseteq Y) & \text{(def. } \Longrightarrow\text{)} \\
\iff & \forall p \in P : \forall a \in A : (\langle p, a\rangle \in \rho) \Longrightarrow (a \sqsubseteq Y) & \text{(def. } \forall\text{)} \\
\iff & P \subseteq \{p \mid \forall a \in A : (\langle p, a\rangle \in \rho) \Longrightarrow (a \sqsubseteq Y)\} & \text{(def. } \subseteq\text{)} \\
\iff & P \subseteq \gamma_\rho(a) & \text{(def. } \gamma_\rho\text{)}
\end{array}
$$

□

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

– Example 1:

If

- $@ \in D^\natural \mapsto \wp(D^\sharp)$
- $\alpha \in \wp(D^\natural) \mapsto \wp(D^\sharp)$
  $\alpha(X) \stackrel{\text{def}}{=} \bigcup\{@(x) \mid x \in X\}$
- $\gamma \in \wp(D^\sharp) \mapsto \wp(D^\natural)$
  $\gamma(Y) \stackrel{\text{def}}{=} \{y \mid @(y) \subseteq Y\}$

then

$$\langle \wp(D^\natural), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(D^\sharp), \subseteq\rangle$$

PROOF. $\alpha(X) \subseteq Y \iff \bigcup\{@(x) \mid x \in X\} \subseteq Y \iff \forall x \in X : @(x) \subseteq Y$
$\iff X \subseteq \{x \mid @(x) \in \subseteq Y\} \iff X \subseteq \gamma(Y)$. □

– Example 2 (semi-dual of example 1):

If

- $@ \in D^\natural \mapsto \wp(D^\sharp)$
- $\alpha \in \wp(D^\natural) \mapsto \wp(D^\sharp)$
  $\alpha(X) \stackrel{\text{def}}{=} \bigcap\{@(x) \mid x \in X\}$
- $\gamma \in \wp(D^\sharp) \mapsto \wp(D^\natural)$
  $\gamma(Y) \stackrel{\text{def}}{=} \{y \mid @(y) \supseteq Y\}$

then

$$\langle \wp(D^\natural), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(D^\sharp), \supseteq\rangle$$

PROOF. $\alpha(X) \supseteq Y \iff \bigcap\{@(x) \mid x \in X\} \supseteq Y \iff \forall x \in X : @(x) \supseteq Y$
$\iff X \subseteq \{x \mid @(x) \in \supseteq Y\} \iff X \subseteq \gamma(Y)$. □

Minimal Abstraction

If

– $\langle L, \leq, 0, 1, \wedge, \vee\rangle$ complete lattice

– $\alpha_m \in \wp(L) \mapsto L$
  $\alpha_m(S) \stackrel{\text{def}}{=} \bigwedge S$

– $\gamma_m \in L \mapsto \wp(L)$
  $\gamma_m(\ell) \stackrel{\text{def}}{=} \{x \in L \mid \ell \leq x\}$

then

$$\langle \wp(L), \subseteq\rangle \underset{\alpha_m}{\overset{\gamma_m}{\leftrightarrows}} \langle L, \geq\rangle$$

PROOF.

$$\alpha_m(S) \geq \ell$$

$\iff \bigwedge S$ ⟨def. α⟩

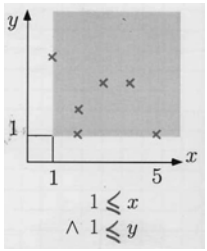
$\iff \forall x \in S : x \geq \ell$ ⟨def. $\bigwedge$⟩

$\iff S \subseteq \{x \in L \mid x \geq \ell\}$ ⟨def. $\subseteq$⟩

$\iff S \subseteq \gamma_m(\ell)$ ⟨def. γ_m⟩

Given $S \in L$, $\alpha(\{S\}) = \bigwedge\{S\} = S$ proving α onto. □

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



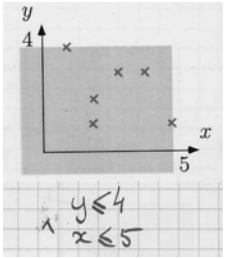
Maximal abstraction

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

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

Proof. Duality on L. □

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



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

Interval abstraction

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

$$\langle \wp(L), \subseteq \rangle \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \langle L \times L, \sqsubseteq \rangle$$

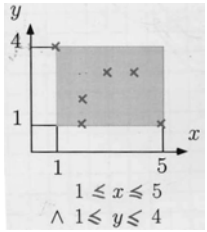
where $[a, b] \sqsubseteq [a', b'] \stackrel{\text{def}}{=} (a' \leq a \wedge b \leq b')$

PROOF.

$$
\begin{array}{ll}
& \alpha_m(S) \geq [a,b] \\
\iff & [\bigwedge S, \bigvee S] \sqsubseteq [a,b] & \text{(def. } \alpha_i\text{)} \\
\iff & a \leq \bigwedge S \wedge \bigvee S \leq b & \text{(def. } \sqsubseteq\text{)} \\
\iff & \forall x \in S : a \leq x \wedge \forall y \in S : y \leq b & \text{(def. } \bigwedge \text{ and } \bigvee\text{)} \\
\iff & \forall x \in S : a \leq x \leq b & \text{(def. } \forall \text{ and } \wedge\text{)} \\
\iff & S \subseteq \{x \in L \mid a \leq x \leq b\} & \text{(def. } \subseteq\text{)} \\
\iff & S \subseteq \gamma_i([a,b]) & \text{(def. } \gamma_i\text{)}
\end{array}
$$

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

- Approximate a set by the pair of its glb and lub
- Example: $\langle \mathbb{Z} \cup \{-\inf, +\inf\}, \leq, -\infty, +\infty, \min, \max \rangle$
- $\alpha_i(\emptyset) = [+\infty, -\infty]$ is the infimum
- $\alpha_i(L) = [-\infty, +\infty]$ is the supremum



Abstraction of a function at a point

If

- S is a set with element $s \in S$
- $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice
- $\alpha_s(f) \stackrel{\text{def}}{=} f(s)$
- $\gamma_s(v) \stackrel{\text{def}}{=} \lambda x \cdot (\!| x = s \,?\, v : \top |\!)$

then

$$\langle S \mapsto L, \dot{\sqsubseteq} \rangle \underset{\alpha_s}{\overset{\gamma_s}{\leftrightarrows}} \langle L, \sqsubseteq \rangle$$

PROOF. $\alpha_s(f) \sqsubseteq v \iff f(s) \sqsubseteq v \iff \forall x \in S : f(x) \sqsubseteq (\!| x = s \,?\, v : \top |\!)$ $\iff \forall x \in S : f(x) \sqsubseteq \gamma_s(v)(x) \iff f \,\dot{\sqsubseteq}\, \gamma_s(v)$ □

Abstraction of a function at a set of points

If

- S is a set with subset $T \in S$
- $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice
- $\alpha_T(f) \stackrel{\text{def}}{=} \lambda y \in Y \cdot f(y)$
- $\gamma_T(\varphi) \stackrel{\text{def}}{=} \lambda x \in S \cdot (\!| x \in T \,?\, \varphi(x) : \top |\!)$

then

$$\langle S \mapsto L, \dot{\sqsubseteq} \rangle \underset{\alpha_s}{\overset{\gamma_s}{\leftrightarrows}} \langle L, \sqsubseteq \rangle$$

PROOF. $\alpha_T(f) \,\dot{\sqsubseteq}\, \varphi \iff \forall y \in T : \alpha_T(f)(y) \sqsubseteq \varphi(y) \iff \forall y \in T : f(y) \sqsubseteq \varphi(y) \iff \forall x \in S : f(x) \sqsubseteq (\!| x \in T \,?\, \varphi(x) : \top |\!) \iff f \,\dot{\sqsubseteq}\, \lambda x \in S \cdot (\!| x \in T \,?\, \varphi(x) : \top |\!) \iff f \,\dot{\sqsubseteq}\, \gamma_s(\varphi)$. □

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
- $T = \{\text{entry point, exit point}\}$
- $L = \wp(\mathbb{X} \mapsto \mathcal{V})$, assertions
- $I \in S \mapsto L$, invariants attached to each program point
- The top is $\mathbb{X} \mapsto \mathcal{V}$ ("I don't know")

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
- $\alpha_f \stackrel{\text{def}}{=} \lambda\varphi \cdot \varphi \circ f$
- $\gamma_f \stackrel{\text{def}}{=} \lambda\psi \cdot \lambda x \cdot \bigsqcap\{\psi(y) \mid y \in T \wedge f(y) = x\}$

then $\langle S \mapsto L, \dot{\sqsubseteq} \rangle \underset{\alpha_f}{\overset{\gamma_f}{\leftrightarrows}} \langle T \mapsto L, \dot{\sqsubseteq} \rangle$

Proof.

$\alpha_f(\varphi) \dot{\sqsubseteq} \psi$

$$
\begin{array}{lll}
\iff & \forall y \in T : \alpha_f(\varphi)(y) \sqsubseteq \psi(y) & \text{〈def. } \dot{\sqsubseteq}\text{〉} \\
\iff & \forall y \in T : \varphi \circ f(y) \sqsubseteq \psi(y) & \text{〈def. } \alpha_f\text{〉} \\
\iff & \forall x \in S : \forall y \in T : (f(y) = x) \Longrightarrow (\varphi(x) \sqsubseteq \psi(y)) & \text{〈def. } \circ \text{ and } =\text{〉} \\
\iff & \forall x \in S : \varphi(x) \sqsubseteq \bigsqcap\{\psi(y) \mid y \in T \wedge f(y) = x\} & \text{〈def. } \textstyle\bigsqcap\text{〉} \\
\iff & \varphi \dot{\sqsubseteq} \lambda x \in S \cdot \bigsqcap\{\psi(y) \mid y \in T \wedge f(y) = x\} & \text{〈def. } \dot{\sqsubseteq}\text{〉} \\
\iff & \varphi \dot{\sqsubseteq} \gamma_f(\psi) & \text{〈def. } \gamma_f\text{〉}
\end{array}
$$

□

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

Abstraction of a set of functions by a function

If

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

then

$$\langle \wp(D \mapsto C), \subseteq \rangle \underset{\alpha_f}{\overset{\gamma_f}{\leftrightarrows}} \langle D \mapsto \wp(C), \dot{\subseteq} \rangle \underset{\alpha_p}{\overset{\gamma_p}{\leftrightarrows}} \langle \wp(D) \overset{\cup}{\longmapsto} \wp(C), \dot{\subseteq} \rangle$$

Proof. Given $\Phi \in \wp(D \mapsto C)$, $\Psi \in D \mapsto \wp(C)$

$$
\begin{array}{ll}
& \alpha_f(\Phi) \dot{\subseteq} \Psi \\
\iff & \lambda x \cdot \{\varphi(x) \mid \varphi \in \Phi\} \dot{\subseteq} \Psi & \text{〈def. } \alpha_f\text{〉} \\
\iff & \forall x \in D : \{\varphi(x) \mid \varphi \in \Phi\} \subseteq \Psi(x) & \text{〈def. } \dot{\subseteq}\text{〉} \\
\iff & \forall x \in D : \forall \varphi \in \Phi : \varphi(x) \in \Psi(x) & \text{〈def. } \subseteq\text{〉} \\
\iff & \forall \varphi \in \Phi : \forall x \in D : \varphi(x) \in \Psi(x) & \text{〈def. } \forall\text{〉} \\
\iff & \Phi \subseteq \{\varphi \mid \forall x \in D : \varphi(x) \in \Psi(x)\} & \text{〈def. } \subseteq\text{〉} \\
\iff & \Phi \subseteq \gamma_f(\Psi) & \text{〈def. } \subseteq\text{〉}
\end{array}
$$

Given $\psi \in D \mapsto \wp(C)$ and $X_i \subset D$, $i \in \Delta$, we have:

$$
\begin{array}{ll}
- & \alpha_p(\Psi)(\bigcup\limits_{i\in\Delta} X_i) \\
= & \bigcup\limits_{x\in\bigcup_{i\in\Delta} X_i} \Phi(x)
\end{array}
$$

$$
\begin{array}{ll}
= & \bigcup\limits_{i\in\Delta} \Phi(X_i) \\
= & \bigcup\limits_{i\in\Delta} \bigcup \{\Phi(x) \mid x \in X_i\} \\
= & \bigcup\limits_{i\in\Delta} \bigcup\limits_{x\in X_i} \Phi(x) \\
= & \bigcup\limits_{i\in\Delta} \alpha_p(\Phi)(X_i)
\end{array}
$$

proving that $\alpha_p(\Phi) \in \wp(D) \stackrel{\sqcup}{\longmapsto} \wp(C)$ is a complete join morphism.

$$
\begin{array}{ll}
- & \alpha_p \circ \gamma_p(\Psi) & \text{〈assuming } \Psi \in \wp(D) \stackrel{\sqcup}{\longmapsto} \wp(C)\text{〉} \\
= & \lambda X \cdot \bigcup\limits_{x\in X} \gamma_p(\Psi)(x) \\
= & \lambda X \cdot \bigcup\limits_{x\in X} \Psi(\{x\})
\end{array}
$$

$$
\begin{array}{ll}
= & \lambda X \cdot \Psi(\bigcup\limits_{x\in X} \{x\}) \\
= & \lambda X \cdot \Psi(X) \\
= & \Psi \\
- & \gamma_p \circ \alpha_P(\phi) & \text{〈assuming } \phi \in D \mapsto \wp(C)\text{〉} \\
= & \lambda x \cdot \alpha_p(\phi)(\{x\}) \\
= & \lambda x \cdot \bigcup\limits_{y\in\{x\}} \Phi(y) \\
= & \lambda x \cdot \Phi(y) \\
= & \Phi
\end{array}
$$

proving that α_p is a bijection with inverse γ_p.

$$
\begin{array}{ll}
- & \alpha_p(\Phi) \subseteq \Psi \\
\Longrightarrow & \gamma_p \circ \alpha_p(\Phi) \subseteq \gamma_p(\Psi) & \text{〈}\gamma_p \text{ monotone}\text{〉} \\
\Longrightarrow & \Phi \subseteq \gamma_p(\Psi) & \text{〈bijection〉}
\end{array}
$$

$$
\begin{array}{ll}
\Longrightarrow & \alpha_p(\Phi) \subseteq \alpha_p \circ \gamma_p(\Psi) & \text{〈}\alpha_p \text{ monotone}\text{〉} \\
\Longrightarrow & \alpha_p(\Phi) \subseteq \Psi & \text{〈bijection〉}
\end{array}
$$

□

- Example 1: often used as a set-transformer collecting semantics $\wp(D) \mapsto \wp(C)$) 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 $\wp(D \mapsto C)$, see [4])
- Example 2 (Cartesian abstraction):
  - Given a set $\mathbb{X}$ of variables and $\mathcal{V}$ of values, properties P of environments $\rho \in \mathbb{X} \mapsto \mathcal{V}$ belong to $\wp(\mathbb{X} \mapsto \mathcal{V})$ that is are sets of functions

- The abstraction $\alpha_f(P)$ consists in considering an abstract environmenty $R \in \mathbb{X} \mapsto \wp(V)$ which records the possible values of the variables $\mathtt{x}X \in \mathbb{X}$ in anyone of the concrete environments $\rho \in P$:

$$\alpha_f(P) = \lambda \mathtt{x} \in \mathbb{X} . \{\rho(\mathtt{x}) \mid \rho \in P\}$$

- All relations between values of variables are therefore lost. For example:

$$\begin{aligned} &\alpha_f(\{\{\mathtt{x} \mapsto 0, \mathtt{y} \mapsto 1\}, \{\mathtt{x} \mapsto 1, \mathtt{y} \mapsto 0\}\}) \\ &= \{\mathtt{x} \mapsto \{0,1\}, \mathtt{y} \mapsto \{0,1\}\} \end{aligned}$$

so the fact that $\mathtt{x} + \mathtt{y} = 1$ is lost.

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

[4] P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In *Proceedings of the 1994 International Conference on Computer Languages*, Toulouse, France, pages 95–112. IEEE Computer Society Press, Los Alamitos, California, USA, 16–19 May 1994.

Abstraction of lattice functionals

If

- S, T are sets
- $\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice
- $F \mathrel{\dot\sqsubseteq} G \stackrel{\text{def}}{=} \forall f \in S \mapsto T : F(f) \sqsubseteq G(f)$
- $\varphi \mathrel{\ddot\sqsubseteq} \psi \stackrel{\text{def}}{=} \forall s \in S : \forall y \in T : \varphi(x)(y) \sqsubseteq \psi(x)(y)$
- $\alpha(F) \stackrel{\text{def}}{=} \lambda x \in S . \lambda y \in T . \bigsqcup \{F(f) \mid f(x) = y\}$
- $\gamma(\varphi) \stackrel{\text{def}}{=} \lambda f \in S \mapsto T . \bigsqcap \{\varphi(x)(f(x)) \mid x \in X\}$

then

$$\langle (S \mapsto T) \mapsto L, \mathrel{\dot\sqsubseteq} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle S \mapsto (T \mapsto L), \mathrel{\ddot\sqsubseteq} \rangle$$

Proof.

$$\begin{aligned} & \alpha(F) \mathrel{\ddot\sqsubseteq} \varphi \\ \iff & \lambda x . \lambda y . \bigsqcup \{F(f) \mid f(x) = y\} \mathrel{\ddot\sqsubseteq} \varphi \\ \iff & \forall x \in S : \forall y \in T : \bigsqcup \{F(f) \mid f(x) = y\} \sqsubseteq \varphi(x)(y) \\ \iff & \forall x \in S : \forall y \in T : \forall f \in S \mapsto T : (f(x) = y) \Longrightarrow (F(f) \sqsubseteq \varphi(x)(y)) \\ \iff & \forall x \in S : \forall f \in S \mapsto T : F(f) \sqsubseteq \varphi(x)(f(x)) \\ \iff & \forall f \in S \mapsto T : \forall x \in S : F(f) \sqsubseteq \varphi(x)(f(x)) \\ \iff & \forall f \in S \mapsto T : F(f) \sqsubseteq \bigsqcap \{\varphi(x)(f(x)) \mid x \in S\} \\ \iff & F \mathrel{\dot\sqsubseteq} \lambda f \in S \mapsto T . \bigsqcap \{\varphi(x)(f(x)) \mid x \in S\} \\ \iff & F \mathrel{\dot\sqsubseteq} \gamma(\varphi) \end{aligned}$$

□

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

Componentwise/Cartesian abstraction of a set of pairs

If

- $\alpha_p(S) \stackrel{\text{def}}{=} \langle \{x \mid \exists y : \langle x, y \rangle \in S\}, \{y \mid \exists x : \langle x, y \rangle \in S\} \rangle$
- $\gamma_p(\langle X, Y \rangle) \stackrel{\text{def}}{=} \{\langle x, y \rangle \mid x \in X \wedge y \in Y\}$
- $\langle x, y \rangle \subseteq_2 \langle x', y' \rangle \stackrel{\text{def}}{=} x \subseteq x' \wedge y \subseteq y'$

then

$$\langle \wp(D_1 \times D_2), \subseteq \rangle \leftrightarrows \langle \wp(D_1) \times \wp(D_2), \subseteq_2 \rangle$$

Proof.

$$\begin{array}{ll} & \alpha_p(S) \subseteq_2 \langle X, Y \rangle \\ \iff & \langle \{x \mid \exists y : \langle x, y \rangle \in S\}, \{y \mid \exists x : \langle x, y \rangle \in S\} \rangle \subseteq_2 \langle X, Y \rangle \end{array}$$

$$\begin{array}{ll} \iff & \{x \mid \exists y : \langle x, y \rangle \in S\} \subseteq X \wedge \{y \mid \exists x : \langle x, y \rangle \in S\} \subseteq Y \\ \iff & S \subseteq \{\langle x, y \rangle \mid x \in X \wedge y \in Y\} \\ \iff & S \subseteq \gamma_p(\langle X, Y \rangle) \end{array}$$

□

- Intuition: abstraction of relational to independent properties, same as on page 144, up to the encoding
- Example:
  - $\alpha_p(\{\langle 1, 2 \rangle, \langle 3, 4 \rangle\}) = \langle \{1, 3\}, \{2, 4\} \rangle$
  - $\gamma_p(\langle \{1, 3\}, \{2, 4\} \rangle) \stackrel{\text{def}}{=} \{\langle 1, 2 \rangle, \langle 1, 4 \rangle, \langle 3, 2 \rangle, \langle 3, 4 \rangle\}$

Pointwise abstraction composition

If

- $\langle P, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Q, \sqsubseteq \rangle$
- $\dot{\alpha}(f) \stackrel{\text{def}}{=} \lambda s \in S \cdot \alpha(f(s))$ pointwise abstraction
- $\dot{\gamma}(g) \stackrel{\text{def}}{=} \lambda s \in S \cdot \gamma(g(s))$
- $f \mathrel{\dot{\sqsubseteq}} g \stackrel{\text{def}}{=} \forall x \in S : f(x) \sqsubseteq g(x)$ pointwise ordering

then

$$\langle S \mapsto P, \mathrel{\dot{\leq}} \rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle S \mapsto Q, \mathrel{\dot{\sqsubseteq}} \rangle$$

Proof.

$$\dot{\alpha}(f) \mathrel{\dot{\sqsubseteq}} g$$

$$\begin{array}{ll} \forall x \in S : \dot{\alpha}(f)(x) \sqsubseteq g(x) & \text{〈def. } \mathrel{\dot{\sqsubseteq}}\text{〉} \\ \forall x \in S : \alpha(f(x)) \sqsubseteq g(x) & \text{〈def. } \dot{\alpha}\text{〉} \\ \forall x \in S : f(x) \leq \gamma(g(x)) & \text{〈Galois connection〉} \\ \forall x \in S : f(x) \leq \dot{\gamma}(g)(x) & \text{〈def. } \dot{\gamma}\text{〉} \\ f \mathrel{\dot{\leq}} \dot{\gamma}(g) & \text{〈def. } \mathrel{\dot{\leq}}\text{〉} \end{array}$$

□

- Used to lift abstractions to vectors (e.g. from one to many variables, from one to many program points, etc.)

Example 1: attribute independant environment abstraction

- $\mathtt{x} \in \mathbb{X}$: set of variables
- $\mathcal{V}$: set of values
- $\wp(\mathcal{V})$: properties of values
- $\rho \in \mathbb{X} \mapsto \mathcal{V}$: environments
- $\wp(\mathbb{X} \mapsto \mathcal{V})$: properties of environments
- $\langle \wp(\mathbb{X} \mapsto \mathcal{V}),\ \subseteq \rangle \underset{\alpha_f}{\overset{\gamma_f}{\leftrightarrows}} \langle \mathbb{X} \mapsto \wp(\mathcal{V}),\ \dot{\sqsubseteq} \rangle$

  $\alpha_f(P) \overset{\text{def}}{=} \lambda \mathtt{x} \in \mathbb{X} . \{ \rho(\mathtt{x}) \mid \rho \in P \}$

- $\langle \wp(\mathcal{V}),\ \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq \rangle$ abstraction of value properties
- $\langle \mathbb{X} \mapsto \wp(\mathcal{V}),\ \dot{\subseteq} \rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{X} \mapsto L,\ \dot{\sqsubseteq} \rangle$ pointwise extension

  $\dot{\alpha}(R) \overset{\text{def}}{=\!=} \lambda \mathtt{x} \in \mathbb{X} . \alpha(R(\mathtt{x}))$
- $\langle \wp(\mathbb{X} \mapsto \mathcal{V}),\ \subseteq \rangle \underset{\dot{\alpha} \circ \alpha_f}{\overset{\gamma_f \circ \dot{\gamma}}{\leftrightarrows}} \langle \mathtt{x} \mapsto L,\ \sqsubseteq \rangle$ by composition of Galois connection

Example 2: local invariants

- Same idea to abstract $\wp(\mathbb{C} \mapsto (\mathbb{X} \mapsto \mathcal{V}))$ where $\mathbb{C}$ is the set of control points
- $\langle \wp(\mathbb{C} \mapsto (\mathbb{X} \mapsto \mathcal{V})),\ \ddot{\subseteq} \rangle \leftrightarrows \langle \mathbb{C} \mapsto (\mathbb{X} \mapsto L),\ \ddot{\sqsubseteq} \rangle$

Componentwise abstraction composition

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

- $\langle P,\ \leq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle P^\sharp,\ \leq^\sharp \rangle$
- $\langle P,\ \preceq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle P^\sharp,\ \preceq^\sharp \rangle$
- $\alpha^\times \overset{\text{def}}{=} \lambda \langle x,\ y \rangle . \langle \alpha_1(x),\ \alpha_2(y) \rangle$ Componentwise abstraction
- $\gamma^\times \overset{\text{def}}{=} \lambda \langle x,\ y \rangle . \langle \gamma_1(x),\ \gamma_2(y) \rangle$

then

$$\langle P \times Q,\ \leq \times \preceq \rangle \underset{\alpha^\times}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp \times Q^\sharp,\ \leq^\sharp \times \preceq^\sharp \rangle$$

PROOF.

$$\begin{array}{ll}
& \alpha^\times(\langle P_1,\ P_2\rangle) \leq^\sharp \times \preceq^\sharp \langle P_1^\sharp,\ P_2^\sharp\rangle \\
\iff & \langle \alpha_1(P_1),\ \alpha_2(P_2)\rangle \leq^\sharp \times \preceq^\sharp \langle P_1^\sharp,\ P_2^\sharp\rangle \\
\iff & \alpha_1(P_1) \leq^\sharp P_1^\sharp \wedge \alpha_2(P_2) \preceq^\sharp P_2^\sharp \\
\iff & P_1 \leq \gamma_1(P_1^\sharp) \wedge P_2 \preceq \gamma_2(P_2^\sharp) \\
\iff & \langle P_1,\ P_2\rangle \leq \times \preceq \langle \gamma_1(P_1^\sharp),\ \gamma_2(P_2^\sharp)\rangle \\
\iff & \langle P_1,\ P_2\rangle \leq \times \preceq \gamma^\times(\langle P_1^\sharp,\ P_2^\sharp\rangle)
\end{array}$$

□

– Used to lift abstractions componentwise to pairs/vectors.

Higher-order abstraction composition

If

– $\langle P,\ \leq\rangle \underset{\alpha_d}{\overset{\gamma_d}{\leftrightarrows}} \langle P^\sharp,\ \leq^\sharp\rangle$

– $\langle Q,\ \preceq\rangle \underset{\alpha_c}{\overset{\gamma_c}{\leftrightarrows}} \langle Q^\sharp,\ \preceq^\sharp\rangle$

– $\vec{\alpha} \stackrel{\text{def}}{=} \lambda\varphi\,.\,\alpha_c \circ \varphi \circ \gamma_d$

– $\vec{\gamma} \stackrel{\text{def}}{=} \lambda\psi\,.\,\gamma_c \circ \psi \circ \alpha_d$

then

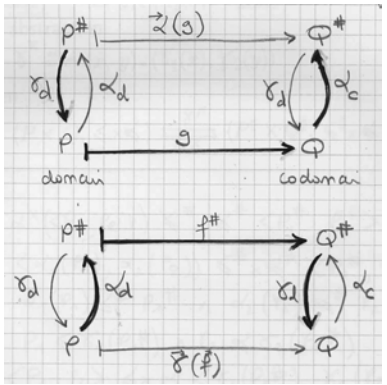
$$\langle P \overset{\text{m}}{\longmapsto} Q,\ \dot{\preceq}\rangle \underset{\vec{\alpha}}{\overset{\vec{\gamma}}{\leftrightarrows}} \langle P^\sharp \overset{\text{m}}{\longmapsto} Q^\sharp,\ \dot{\preceq}^\sharp\rangle$$

PROOF. Given monotonic $f \in P \overset{\text{m}}{\longmapsto} Q$ and $g \in P^\sharp \overset{\text{m}}{\longmapsto} Q^\sharp$, we have:

$$\begin{array}{lll}
& \vec{\alpha}(f) \dot{\preceq}^\sharp g & \\
\iff & \forall x \in P^\sharp : \vec{\alpha}(f)(x) \preceq^\sharp g(x) & \text{(def. } \dot{\preceq}^\sharp\text{)} \\
\iff & \forall x \in P^\sharp : \alpha_c \circ f \circ \gamma_d(x) \preceq^\sharp g(x) & \text{(def. } \vec{\alpha}\text{)} \\
\iff & \forall x \in P^\sharp : f \circ \gamma_d(x) \preceq \gamma_c(g(x)) & \text{(Galois connection)} \\
\implies & \forall y \in P : f \circ \gamma_d \circ \alpha_d(y) \preceq \gamma_c \circ g \circ \alpha_d(y) & \text{(for } x = \alpha_d(y) \text{ and def. } \circ\text{)} \\
\implies & \forall y \in P : f(y) \preceq \gamma_c \circ g \circ \alpha_d(y) & \text{(} f \text{ monotone and } \gamma_d \circ \alpha_d \text{ extensive)} \\
\iff & f \dot{\preceq} \gamma_c \circ g \circ \alpha_d & \text{(def. } \dot{\preceq}\text{)} \\
\iff & f \dot{\preceq} \vec{\gamma}(g) & \text{(def. } \vec{\gamma}\text{)} \\
\implies & \forall y \in P : f(y) \preceq \gamma_c \circ g \circ \alpha_d(y) & \text{(def. } \vec{\gamma} \text{ and } \dot{\preceq}\text{)} \\
\implies & \forall x \in P^\sharp : f(\gamma_d(x)) \preceq \gamma_c \circ g \circ \alpha_d(\gamma_d(x)) & \text{(for } x = \gamma_d(x)\text{)} \\
\implies & \forall x \in P^\sharp : f(\gamma_d(x)) \preceq \gamma_c \circ g(x) & \text{(} \alpha_d \circ \gamma_d \text{ reductive, } \gamma_c \text{ and } g \text{ monotone)} \\
\iff & \forall x \in P^\sharp : \alpha_c \circ f \circ \gamma_d(x) \preceq^\sharp g(x) & \text{(Galois connection and def. } \circ\text{)} \\
\iff & \vec{\alpha}(f) \dot{\preceq}^\sharp g & \text{(def. } \vec{\alpha} \text{ and } \dot{\preceq}^\sharp\text{)}
\end{array}$$

□

– 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/.

