Theory of Program Analysis

Saumya K. Debray
Dept. of Computer Science
The University of Arizona
Tucson, AZ 85721, USA

Why Analyse Programs?

- compiler optimizations.
- program testing.
- program transformation and partial evaluation.
- program verification, e.g. proofs of termination or liveness.

Objective : Want to infer run-time properties of programs by analysing the program at compile time.

Problem : exact analysis may be very expensive:

- Most nontrivial program properties are undecidable (Rice's Theorem).
- Even decidable properties may be complete for NP or PSPACE.

Solution : Give decidable approximations that are

(i) sound (i.e., do not lie); and

(ii) (hopefully) computationally more tractable.

Note : To reason about the soundness of an anaylsis it is necessary to establish relationships with the formal semantics of the language.

Programming Language Semantics

- *Denotational* :
  - The domain of computation is typically a complete lattice or CPO.
  - The meaning of a program construct is viewed as a function over this domain.
  - The meaning of a program is typically obtained as the (least) solution of a set of recursive equations.
- *Operational* :
  - the meaning of program constructs is defined in terms of state transformations.
- *Axiomatic* :
  - Manipulate logical formulae representing *preconditions* and *postconditions*.
  - The meaning of a program construct is given in terms of derivation rules that give the relationship between pre- and post-conditions for that statement.

Program Analysis: Formal Frameworks

1. *Abstract Interpretation* [Cousot & Cousot 1977] :
   (a) *Galois Connection Approach* : Define a domain of "descriptions", relate it to the actual computational domain via a Galois connection. Program analysis carried out over the description domain.
   (b) *Widening/Narrowing Approach* : No separate domain of descriptions, analysis carried out over the actual computational domain using "widening" and "narrowing" operators.
   (c) *Hybrid Approach* : Augment the Galois Connection approach with widening and narrowing operators to accelerate convergence.
2. *Projection Analysis* : Uses certain kinds of idempotent functions over the computational domain. Typically used for the analysis of functional languages (e.g., [Wadler & Hughes 1987]).

The Galois Connection Approach: An Overview

- Use descriptions of sets of values, rather than the values themselves.
- Simulate the execution of the program over the domain of descriptions.
  - This can be seen as setting up and solving a set of (recursive) equations
    $$\begin{aligned} x_1 &= f_1(x_1, \ldots, x_n) \\ &\vdots \\ x_n &= f_n(x_1, \ldots, x_n) \end{aligned}$$
  - Existence of solutions, convergence, etc., can be addressed within a lattice-theoretic framework, e.g. using the Knaster-Tarski fixpoint theorem.
- *Reasoning about soundness* : specify relationships between computations over the domain of descriptions and the semantics of the language.

Example 1 : The Rule of Signs

- *concrete computation* : integer multiplication.
- *description domain* : **Sgn** $= \{\bot, 0, -, +, \top\}$.
- *computations over the description domain* :

$\otimes$:

	$\bot$	$+$	0	$-$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$+$	$\bot$	$+$	0	$-$	$\top$
0	$\bot$	0	0	0	0
$-$	$\bot$	$-$	0	$+$	$\top$
$\top$	$\bot$	$\top$	0	$\top$	$\top$

- *concrete computation* :
  $1517 \times -323913 = -49929021$
- *abstract computation* : '$+$' $\otimes$ '$-$' $= -$

Note : $\otimes$ "abstracts" the effects of $\times$ over the description domain.

Relating the Abstract and Concrete Domains :

- For analysis purposes, it is convenient to "lift" operations to sets of values (so $+$ operates over $\wp(\mathbf{Z})$ rather than over $\mathbf{Z}$).
- abstraction function $abs : \wp(\mathbf{Z}) \longrightarrow \mathbf{Sgn}$ specifies the (best) description for a set of values.
- concretization function $conc : \mathbf{Sgn} \longrightarrow \wp(\mathbf{Z})$ specifies the (biggest) set described by an element of **Sgn**.

$$\begin{tikzcd} \mathbf{Sgn} \times \mathbf{Sgn} \arrow[r, "\otimes"] & \mathbf{Sgn} \arrow[d, "conc"] \\ \wp(\mathbf{Z}) \times \wp(\mathbf{Z}) \arrow[u, "abs"] \arrow[r, "\times"] & \wp(\mathbf{Z}) \end{tikzcd}$$

- *soundness* : show that for all $x, y \in \mathbf{Z}$:
  $x \times y \in conc(abs(x) \otimes abs(y))$.

Abstract Interpretation: A General Treatment

[Cousot & Cousot, 1977]

- *programs* : represented as flow graphs:
  - *nodes* : program statements
  - *arcs* : flow of control ("program points")
  - assume $\exists$ distinguished entry, exit nodes.
- *program semantics* : Scott-Strachey style denotational semantics:
  - Environments (**Env**) = Identifiers $\longrightarrow$ Values
  - States = Arcs $\times$ Environments
- The execution of a program associates a *sequence* of environments with each program point.

  For program analysis purposes, it suffices to consider the *set* of environments corresp. to each program point ("*collecting semantics*").
- "Concrete Domain" $\mathcal{D}_{conc} = \langle \mathcal{P}(\mathbf{Env}), \subseteq \rangle$.

The "Concrete Domain" $\mathcal{D}_{conc}$

- Intuitively, each element of $\mathcal{D}_{conc}$ represents a *set* of possibilities that may be realized during a computation.

  If U represents the "universe of possibilities", then we could think of $\mathcal{D}_{conc}$ as $\langle \mathcal{P}(U), \subseteq \rangle$, which forms a complete lattice.
- In general, semantic definitions include many kinds of domains other than powersets, so we generally consider $\mathcal{D}_{conc}$ to be an arbitrary complete lattice ordered by $\sqsubseteq$.
- The top element of $\mathcal{D}_{conc}, \sqsubseteq$ represents "all possibilities", and thus conveys no information. (This is opposite to the ordering used in domain theory, where the bottom element conveys no information.)

The "Abstract Domain" $\mathcal{D}_{abs}$

- Elements of $\mathcal{D}_{abs}$ describe elements of $\mathcal{D}_{conc}$.
- To facilitate reasoning about fixpoints, $\mathcal{D}_{abs}$ should be a complete lattice.
- The *meaning* of each $d \in \mathcal{D}_{abs}$ should be well defined: there should be a function

  $conc : \mathcal{D}_{abs} \longrightarrow \mathcal{D}_{conc}$
- $conc$ should be 1-1, so that there are no redundant descriptions.
- The structures of $\mathcal{D}_{abs}$ and $\mathcal{D}_{conc}$ should correspond: $conc$ should be monotone.
- Given $x \in \mathcal{D}_{conc}$, we may require the existence of a *best* (i.e., smallest) $y \in \mathcal{D}_{abs}$ that describes x:

  $\mathcal{D}_{abs}$ should be a *Moore family*.

Definition : Let $\langle L, \leq \rangle$ be a complete lattice, then $S \subseteq \mathcal{P}(L)$ is a *Moore family* iff

1. $L \in S$; and
2. $\sqcap T \in S$ for all $T \subseteq S$.

Abstract Interpretation (General) : cont'd

- *concrete interpretation* : Suppose the language has a single primitive operation f, then:

  $\langle \mathcal{D}_{conc}, \sqsubseteq, \sqcup, f \rangle$

  where $\langle \mathcal{D}_{conc}, \sqsubseteq \rangle$ is a complete lattice with join operation $\sqcup$.
- *abstract interpretation* : $\langle \mathcal{D}_{abs}, \leq, \vee, \widehat{f} \rangle$ where $\langle \mathcal{D}_{abs}, \leq \rangle$ is a complete (join semi-)lattice with join operation $\vee$.
- The abstract and concrete domains are related by functions

  $abs : \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{abs}$
  $conc : \mathcal{D}_{abs} \longrightarrow \mathcal{D}_{conc}$
- The abstract operation $\widehat{f} : \mathcal{D}_{abs} \longrightarrow \mathcal{D}_{abs}$ "simulates" the concrete operation $f : \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{conc}$.

$\mathcal{D}_{conc}$ and $\mathcal{D}_{abs}$: Structural Correspondence

- The structure of $\mathcal{D}_{conc}$ and $\mathcal{D}_{abs}$ should correspond. For this, we need

  $$\mathcal{D}_{conc}, \mathcal{D}_{abs} \text{ are complete lattices} \tag{1}$$

  $$abs, conc \text{ monotone.} \tag{2}$$

- Notions of "approximation" in $\mathcal{D}_{conc}$ and $\mathcal{D}_{abs}$ should correspond:

  $$\forall x \in \mathcal{D}_{conc}, y \in \mathcal{D}_{abs} : [abs(x) \leq y \Leftrightarrow x \sqsubseteq conc(y)] \tag{3}$$

  abs and $conc$ are said to be *adjoint*. They define a *Galois connection* between $\mathcal{D}_{conc}$ and $\mathcal{D}_{abs}$.

- (3) can be rewritten as:

  $$\forall x \in \mathcal{D}_{conc} : x \sqsubseteq conc(abs(x)) \tag{3a}$$

  $$\forall x \in \mathcal{D}_{abs} : x \leq abs(conc(x)) \tag{3b}$$

- Want information loss only during abstraction: ($3b$) can be strengthened to

  $$\forall x \in \mathcal{D}_{abs} : x = abs(conc(x)) \tag{3b'}$$

- If (1), (2), ($3a$), ($3b'$) are satisfied by $\mathcal{D}_{conc}$, $\mathcal{D}_{abs}$, abs, $conc$, they are sometimes said to form a Galois insertion.

Soundness of Program Analysis

- *Soundness* $\Rightarrow$ anything that can happen at runtime is predicted by the analysis.

- undecidability problems $\Rightarrow$ the analysis may predict some things that cannot happen at runtime ("conservative analysis").

- Suppose the language has a primitive operation $f : \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{conc}$.

  Let the corresponding abstract operation be $\widehat{f} : \mathcal{D}_{abs} \longrightarrow \mathcal{D}_{abs}$.

- We need at least *local consistency* :

  $$\forall x \in \mathcal{D}_{conc} \ : \ f(x) \sqsubseteq conc(\widehat{f}(abs(x))). \tag{4}$$

$$\begin{tikzcd} \widehat{x} \arrow[r, "\widehat{f}"] & \widehat{f}(\widehat{x}) \arrow[d, "conc"] \\ & \bullet \arrow[d, phantom, "\sqsupseteq"] \\ x \arrow[uu, "abs"] \arrow[r, "f"] & f(x) \end{tikzcd}$$

Soundness of Program Analysis – cont'd

- Suppose properties (1), (2), ($3a$), ($3b'$), and (4) are satisfied, i.e.:
  - $\langle \mathcal{D}_{conc}, \sqsubseteq \rangle$ and $\langle \mathcal{D}_{abs}, \leq \rangle$ are complete lattices;
  - abs and $conc$ are monotone and adjoint; and
  - the abstract operation $\widehat{f}$ is locally consistent with the concrete operation f.

- By induction, we have:

  $\forall n \in \mathbf{N} \ : \ \bigsqcup f^n(\bot) \sqsubseteq conc(\bigvee \widehat{f}^n(abs(\bot))).$

- This may not be enough: we want to prove "global consistency":

  $lfp(f) \sqsubseteq conc(lfp(\widehat{f})),$ or

  $$\bigsqcup_{n=0}^{\infty} f^n(x) \sqsubseteq conc(\bigvee_{n=0}^{\infty} \widehat{f}^n(abs(\bot))).$$

- This would seem to require fixpoint induction, and hence additional assumptions about *admissibility* of the predicates we are considering.

Soundness of Program Analysis – cont'd

Why is this an issue?

- Soundness claims for analyses *must* refer to the meaning of a program.

  The meaning of a program is defined as a (least) fixpoint.

- Consider *finiteness analysis* for deductive databases.

  Problem : Given a logic program P, determine whether the relation defined by a predicate in the least model of P is finite or infinite.

  Obvious Analysis : $\mathcal{D}_{abs} = \{$finite, infinite$\}$.

  - "least model" $\Rightarrow$ start with $\emptyset$, which is finite.
  - any rule, invoked on finite relations, will produce only finite relations [T_P operator].
  - conclude that all relations are always finite.

    What about:

    ```
    int(0).
    int(s(X)) :- int(X).
    ```

Soundness of Program Analysis – cont'd

Theorem [Cousot & Cousot 77] : Consider an abstract interpretation satisfying:

– $\langle \mathcal{D}_{conc}, \sqsubseteq \rangle$ and $\langle \mathcal{D}_{abs}, \leq \rangle$ are complete lattices;

– abs and $conc$ are monotone and adjoint; and

– the abstract operation $\widehat{f}$ is locally consistent with the concrete operation f.

Then, $lfp(f) \sqsubseteq conc(lfp(\widehat{f}))$, and
$gfp(f) \sqsubseteq conc(gfp(\widehat{f}))$.

Remark : For a given language (hence, $\mathcal{D}_{conc}$), suppose we define $\mathcal{D}_{abs}, abs, conc$ and ensure that $\widehat{f}$ is locally consistent with f. If we find that the analysis is not sound, i.e.,

$$lfp(f) \not\sqsubseteq conc(lfp(\widehat{f})).$$

Then it must be that $\mathcal{D}_{conc}, \mathcal{D}_{abs}, abs, conc$ do not form a Galois connection.

Design of Abstract Operations

- For fixed $\mathcal{D}_{conc}$ and $\mathcal{D}_{abs}$, there can be many different sound abstract interpretations.
- For some concrete operation f, let $\widehat{f}_1$ and $\widehat{f}_2$ be two abstract operations satisfying the soundness condition (4).

  Given a program P and input I, let $A[f : \widehat{f}_1](P, I)$ denote the result of analysing $\langle P, I \rangle$ using $\widehat{f}_1$ to abstract f (sim. for $A[f : \widehat{f}_2](P, I)$).

  Then, $\widehat{f}_1$ is *better than* $\widehat{f}_2$ iff for every program P and input I, the analysis of $\langle P, I \rangle$ is more precise using $\widehat{f}_1$ than using $\widehat{f}_2$, i.e., iff

  $$conc(A[f : \widehat{f}_1](P, I)) \subseteq conc(A[f : \widehat{f}_2](P, I)).$$
- Of all sound abstract operators, there is a *best* one, given by:

  $$\widehat{f} = abs \circ f \circ conc.$$

The Galois Connection Approach : Summary

Let $\langle \mathcal{D}_{conc}, \sqsubseteq \rangle$ and $\langle \mathcal{D}_{abs}, \leq \rangle$ be complete lattices, and

$abs : \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{abs}$, and
$conc : \mathcal{D}_{abs} \longrightarrow \mathcal{D}_{conc}$

be monotone and adjoint. Then:

- $\langle abs, conc \rangle$ defines a Galois connection between $\mathcal{D}_{conc}$ and $\mathcal{D}_{abs}$.
- $conc \circ abs$ is a upper closure operator on $\mathcal{D}_{conc}$, and $abs \circ conc$ is a lower closure operator (identity function in the case of Galois insertions).
- abs and $conc$ determine each other:

  $abs(x) = \bigwedge \{y \in \mathcal{D}_{abs} \mid x \sqsubseteq conc(y)\}$
  $conc(x) = \bigsqcup \{y \in \mathcal{D}_{conc} \mid abs(x) \leq y\}$
- abs is a complete join-morphism:

  $$\forall S \subseteq \mathcal{D}_{conc} : abs(\sqcup S) = \vee \{abs(s) \mid s \in S\}$$

  $conc$ is a complete meet-morphism:

  $$\forall S \subseteq \mathcal{D}_{abs} : conc(\wedge S) = \sqcap \{conc(s) \mid s \in S\}$$
- *Termination* : require that the abstract domain contain no infinite (ascending) chains.

Example 2 [Cousot & Cousot, 1976] :

Given a Pascal-like language with integer variables only, analyze programs to estimate upper and lower bounds on the values each variable can take on.

- $\mathcal{D}_{conc} = \langle \mathcal{P}(\mathbf{Z}), \subseteq \rangle$;
- $\mathcal{D}_{abs} = \{\bot\} \cup \{[l, u] \mid l \in \mathbf{Z} \cup \{-\infty\}, u \in \mathbf{Z} \cup \{+\infty\}\}$;
- $abs(\emptyset) = \bot$;
  $abs(S) = [\min(S), \max(S)]$ if $S \neq \emptyset$;
- $conc(\bot) \quad = \emptyset$;
  $conc([l, u]) = \{n \in \mathbf{Z} \mid l \leq n \leq u\}$;
- ordering on the abstract domain: given $s_1, s_2 \in \mathcal{D}_{abs}$,

  $$s_1 \leq s_2 \text{ iff } conc(s_1) \subseteq conc(s_2).$$

Note : $\langle \mathcal{D}_{abs}, \leq \rangle$ contains infinite ascending chains, so termination is not guaranteed.

Alternative Formulation: Axiomatic Semantics

- $\mathcal{D}_{conc}$ is the lattice of assertions (modulo logical equivalence), ordered by logical implication.
- $\mathcal{D}_{abs}$ is a sublattice of $\mathcal{D}_{conc}$.
- $abs, conc$ are as before.
- The meaning of a primitive operation f, and its abstract counterpart $\widehat{f}$, is given as a predicate transformer.

Note : The semantics μ_f of a primitive operation f can be given in two equivalent ways:

1. given a postcondition φ, $\psi = \mu_f(\varphi)$ is the *weakest precondition* such that

   $\{\psi\}f\{\varphi\}$.

2. given a precondition ψ, $\varphi = \mu_f(\varphi)$ is the *strongest postcondition* such that

   $\{\psi\}f\{\varphi\}$.

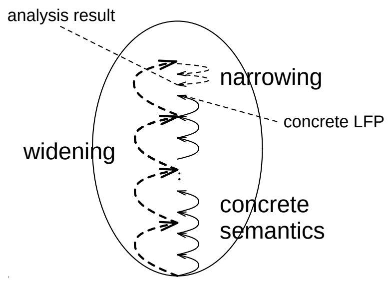
Accordingly, any forward dataflow equation can be reformulated as an equivalent backward dataflow equation.

The Widening/Narrowing Approach

- Conceptually, no separate domain of descriptions: the analysis uses the concrete domain $\mathcal{D}_{conc}$.

  (However, the elements considered must have finite representations: these can be considered to be descriptions.)

- *Basic Idea* :
  1. Use a *widening* operator $\bigtriangledown$ to get a sound approximation to the meaning of the program.
  2. Use a *narrowing* operator $\bigtriangleup$ to improve this approximation while remaining sound.



Widening

$\bigtriangledown : \mathcal{D}_{conc} \times \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{conc}$ must satisfy

1. $\forall x, y \in \mathcal{D}_{conc} : x \sqsubseteq (x \bigtriangledown y)$ and $y \sqsubseteq (x \bigtriangledown y)$;

2. for all increasing chains $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \cdots$ in $\mathcal{D}_{conc}$, the chain

   $y_0 = x_0, \ldots, y_{i+1} = y_i \bigtriangledown x_{i+1}, \ldots$

   is finite.

Note : Let $\mathcal{X}$ be the chain $x_0, x_1, \ldots, x_i, \ldots$, where

$$x_0 = \bot, x_1 = x_0 \sqcup f(\bot), \ldots, x_{i+1} = x_i \sqcup f^{i+1}(\bot), \ldots$$

so that $\mathit{lfp}(f) = \sqcup\mathcal{X}$. Let $\mathcal{Y}$ be the chain $y_0, y_1, \ldots, y_i, \ldots$, where

$$y_0 = x_0 = \bot, \ldots, y_{i+1} = y_i \bigtriangledown x_{i+1}, \ldots$$

Then:

- From (1), $x_i \sqsubseteq y_i$ for all $i \geq 0$, so that $\mathit{lfp}(f) \sqsubseteq \sqcup\mathcal{Y}$.
- From (2), $\sqcup\mathcal{Y}$ can be computed in finitely many steps.

Narrowing

$\bigtriangleup : \mathcal{D}_{conc} \times \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{conc}$ must satisfy

1. $\forall x, y, l : [l \sqsubseteq x$ and $l \sqsubseteq y] \Rightarrow [l \sqsubseteq (x \bigtriangledown y) \sqsubseteq x]$;

2. for all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq x_2 \sqsupseteq \cdots$ in $\mathcal{D}_{conc}$, the chain

   $y_0 = x_0, \ldots, y_{i+1} = y_i \bigtriangleup x_{i+1}, \ldots$

   is finite.

Note : Let $\mathcal{X}$ be any chain $x_0, x_1, \ldots, x_i, \ldots$, where

$\mathit{lfp}(f) \sqsubseteq x_i$ for all $i \geq 0$.

Let $\mathcal{Y}$ be the chain $y_0, y_1, \ldots, y_i, \ldots$, where

$y_0 = x_0$;
$y_{i+1} =$ **if** $f(x_i) = x_i$ **then** x_i **else** $x_i \bigtriangledown f(x_i)$.

Then:

- From (1), $\mathit{lfp}(f) \sqsubseteq y_i$ for all $i \geq 0$, so that $\mathit{lfp}(f) \sqsubseteq \sqcap\mathcal{Y}$.
- From (2), $\sqcap\mathcal{Y}$ can be computed in finitely many steps.

Program Analysis using Widening/Narrowing

Suppose the meaning of a program P is given by a function $f_P : \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{conc}$. The analysis proceeds as follows:

1. *Widening* : Compute the limit of the chain $\mathcal{X}$ given by

   $x_0 = \bot$;
   $x_{i+1} = x_i \bigtriangledown (x_i \sqcup f_P(x_i))$.

   From the definition of $\bigtriangledown$,

   1. $lfp(f_P) \sqsubseteq \sqcup\mathcal{X}$; and
   2. $\sqcup\mathcal{X}$ is computable in finitely many steps.

2. *Narrowing* : Improve the result obtained from the widening phase: compute the limit of the chain $\mathcal{Y}$, given by:

   $y_0 = \sqcup\mathcal{X}$;
   $y_{i+1} = \textbf{if } f_P(y_i) = y_i \textbf{ then } y_i \textbf{ else } y_i \bigtriangleup f_P(y_i)$

   From the definition of $\bigtriangleup$,

   1. $lfp(f_P) \sqsubseteq \sqcap\mathcal{Y} \sqsubseteq \sqcup\mathcal{X}$, i.e., $\sqcap\mathcal{Y}$ is a sound approximation to $lfp(f_P)$; and
   2. $\sqcap\mathcal{Y}$ is computable in finitely many steps.

Examples of Widening :

- *Type Graph Normalization* [Bruynooghe *et al.* 87, Janssens 90, van Hentenryck *et al.* 94] :

  Given a type graph τ, produce a type graph τ' such that, for some given $c \geq 0$:

  (i) the no. of occurrences of any constructor along an acyclic path from the root of τ' is bounded by c; and

  (ii) $conc(\tau) \subseteq conc(\tau')$.

- *Star Abstraction* [Codish *et al.* 91]:

  Given a description G of a goal, produce a description G' such that G' has at most one occurrence of each predicate symbol.

- *Finiteness Analysis* [Bigot *et al.* 92]:

  Finiteness of relations is not an “admissible” property, so conventional abstract interpretation does not work, but we can use an admissible property that implies finiteness.

- *Abstract Depth Abstraction* [Debray 92]:

  “Abstract terms” involving constructors yield abstract domains with infinite chains. Depth abstraction can be used on abstract terms to ensure termination.

The Hybrid Approach

- Augments the Galois Connection approach with widening and narrowing operators to accelerate convergence.

- Useful if the abstract domain contains infinite chains: widening/narrowing can be used to guarantee termination.

- *Basic Idea* :

  - Given a concrete domain operation

    $f : \mathcal{D}_{conc} \longrightarrow \mathcal{D}_{conc}$

    let $\widehat{f} : \mathcal{D}_{abs} \longrightarrow \mathcal{D}_{abs}$ be the corresponding abstract operation.

  - Use widening: compute the limit $X^\star$ of the sequence

    $x_0 = \bot$;
    $x_{i+1} = x_i \bigtriangledown (x_i \vee \widehat{f}(x_i))$.

  - Use narrowing: compute the limit $Y^\star$ of the sequence

    $y_0 = X^\star$;
    $y_{i+1} = \textbf{if } \widehat{f}(y_i) = y_i \textbf{ then } y_i \textbf{ else } y_i \bigtriangleup \widehat{f}(y_i)$

Example 3 [Cousot & Cousot, 1976] :

Consider the analysis problem of Example 2.

- $\mathcal{D}_{abs} = \{\bot\} \cup \{[l, u] \mid l \in \mathbf{Z} \cup \{-\infty\}, u \in \mathbf{Z} \cup \{+\infty\}\}$;

- *widening* : gives up all information on a bound that is “unstable”:

  $\bot \bigtriangledown x = x$;
  $x \bigtriangledown \bot = x$;
  $[l_0, u_0] \bigtriangledown [l_1, u_1] =$
  $[\textbf{if } l_1 < l_0 \textbf{ then } -\infty \textbf{ else } l_0,$
  $\textbf{if } u_1 > u_0 \textbf{ then } +\infty \textbf{ else } u_0]$;

- *narrowing* : only improves infinite bounds:

  $\bot \bigtriangleup x = \bot$;
  $x \bigtriangleup \bot = \bot$;
  $[l_0, u_0] \bigtriangleup [l_1, u_1] =$
  $[\textbf{if } l_0 = -\infty \textbf{ then } l_1 \textbf{ else } \min(l_0, l_1),$
  $\textbf{if } u_1 = +\infty \textbf{ then } u_1 \textbf{ else } \max(u_0, u_1)]$;

Note : Unlike Example 2, termination can now be guaranteed.

Galois Connection vs. Widening/Narrowing

Suppose:

1. For reasons of precision, we want to analyze our programs over an infinite domain; and

2. For reasons of practicality, we want analyses that always terminate.

Then, it can be shown that there exist infinite domains and widening/narrowing operators such that the following are true [Cousot & Cousot 91]:

- For any given program, there is a finite lattice L such that Galois connection analysis over L gives results equivalent to those obtainable via widening/narrowing.

- In general, no such finite lattice will do for all programs.

- For all programs, infinitely many abstract values are necessary.

- For a particular program, it is not possible to infer the set of needed abstract values by a simple inspection of the text of the program.

An Alternative Approximation Method

Consider a language with an operation $f : \mathcal{D} \longrightarrow \mathcal{D}$ where :

- $\langle \mathcal{D}, \sqsubseteq \rangle$ is a complete lattice;

- the least element of this lattice is $\perp$; and

- the greatest element of this lattice is $\top$.

We want to obtain an approximation A to $\mathit{lfp}(f)$.

Approach 1 : An analysis starting with $\perp$ and iterating over an ascending Kleene sequence (using widening/narrowing if there are infinite chains) can be used to obtain an approximation A_0 satisfying

$$\mathit{lfp}(f) \sqsubseteq A_0.$$

Approach 2 : An analysis starting at $\top$ and iterating over a descending Kleene sequence can be used to obtain an approximation A_1 satisfying

$$\mathit{lfp}(f) \sqsubseteq \mathit{gfp}(f) \sqsubseteq A_1.$$

In general, the two approaches are not comparable in precision.