Abstract Interpretation: concrete and abstract semantics

Concrete semantics

- We consider a very tiny language that manages arithmetic operations on integers values.
- The (concrete) semantics of the languages cab be defined by the funzcion μ defined by:

$$e = i \mid e * e$$

$$\mu : Exp \to Int$$

$$\begin{aligned} \mu(i) &= i \\ \mu(e_1 * e_2) &= \mu(e_1) \times \mu(e_2) \end{aligned}$$

Abstract Semantics

- Consider now an abstract semantics over the domain of signs

$$\sigma : \text{Exp} \to \{+,-,0\}$$

$$\sigma(i) = \begin{pmatrix} + & \text{if } i > 0 \\ 0 & \text{if } i = 0 \\ - & \text{if } i < 0 \end{pmatrix}$$

$$\sigma(e_1 * e_2) = \sigma(e_1) \;\bar{\times}\; \sigma(e_2)$$

$\bar{\times}$	$+$	0	$-$
$+$	$+$	0	$-$
0	0	0	0
$-$	$-$	0	$+$

From a different perspective

- We can associate to each abstract value the set of concrete elements it represents.
- The concretization function :

$$\gamma : \{+, 0, -\} \rightarrow 2^{Int}$$

$$\begin{aligned} \gamma(+) &= \{ i \mid i > 0 \} \\ \gamma(0) &= \{ 0 \} \\ \gamma(-) &= \{ i \mid i < 0 \} \end{aligned}$$

Concretization

- The concretization function γ maps an abstract value to a set of concrete elements
- Let D denote the comncrete domain and A denote the abstract domain. The correctness of the abstract semantics wrt the concrete one can be expressed by:

$$\mu(e) \in \gamma(\sigma(e))$$

$$\begin{tikzcd} & A \arrow[d, "\gamma"] \\ \mathrm{Exp} \arrow[ur, "\sigma"] \arrow[r, "\mu"'] & 2^{D} \end{tikzcd}$$

Abstract Interpretation

- Abstract Interpretation is:
  - Computing the semantics of a program in an abstract domain
  - In the case of signs, the domain so far is {+,0,-}.

- The abstract semantics should be correct
  - it is an over approximation of the concrete semantics

- The relatrion between te two domains is given by a concretization function

Consider the unary operator -

- Let us add to our language the unary operator -

$$\mu(-e) = -\mu(e)$$

$$\sigma(-e) = \bar{-}\sigma(e)$$

$\bar{-}$	$+$	0	$-$
	$-$	0	$+$

Consider the binary operation +

- Adding the addition operator focrces us to modify the domain, as the previous one is not able to represent the result of adding numbers of opposite sign

$$\mu(e_1 + e_2) = \mu(e_1) + \mu(e_2)$$

$$\sigma(e_1 + e_2) = \sigma(e_1) \; \overline{+} \; \sigma(e_2)$$

$\overline{+}$	$+$	0	$-$
$+$	$+$	$+$	$?$
0	$+$	0	$-$
$-$	$?$	$-$	$-$

So…

- We add to the domain a new element that represents all the integer numbers (both positive and negative, and zero)

$$\gamma(\mathrm{T}) = \mathrm{Int}$$

$\bar{+}$	$+$	0	$-$	T
$+$	$+$	$+$	T	T
0	$+$	0	$-$	T
$-$	T	$-$	$-$	T
T	T	T	T	T

The operations should be revisited

$\bar{\times}$	+	0	−	T
+	+	0	−	T
0	0	0	0	0
−	−	0	+	T
T	T	0	T	T

$\bar{-}$	+	0	−	T
	−	0	+	T

Examples

- Sometimes there is information loss due to the abstract operations

$$\mu((1+2)+-3) \quad = \quad 0$$

$$\sigma((1+2)+-3) \quad = \quad (+ \;\bar{+}\; +) \;\bar{+}\; (\bar{-}+) = \top$$

- Sometimes there is no information loss, with respect to the abstraction

$$\mu((5*5)+6) \quad = \quad 31$$

$$\sigma((5*5)+6) \quad = \quad (+ \;\bar{\times}\; +) \;\bar{+}\; + = +$$

Consider the division operator /

- Problem: what is the result of dividing by zero? No number!
- So we need a new element in our domain that represents the empty set of integers (i.e. a failure state)

- **But.. What's wrong in the table below?**

$$\gamma(\perp) = \emptyset$$

$\overline{/}$	$+$	0	$-$	$\top$	$\perp$
$+$	$+$	0	$-$	$\top$	$\perp$
0	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
$-$	$-$	0	$+$	$\top$	$\perp$
$\top$	$\top$	0	$\top$	$\top$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

$$\perp \mathbin{\overline{+}} x = \perp$$

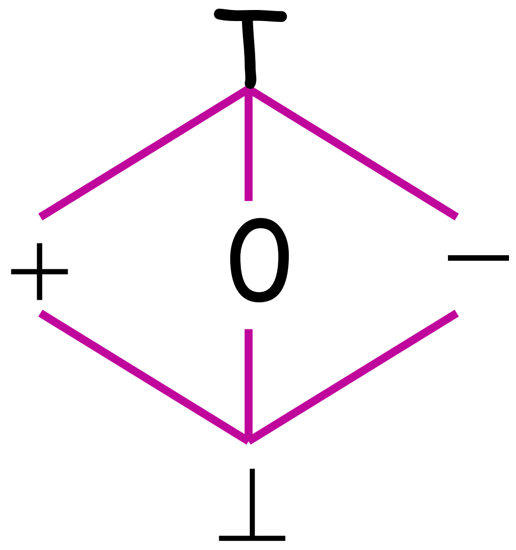
$$x \mathbin{\overline{\times}} \perp = \perp$$

$$\overline{-} \perp = \perp$$

The resulting abstract domain

- It is a finite complete lattice
- The partial order is coherent wrt the concretization function:

$$x \leq y \Leftrightarrow \gamma(x) \subseteq \gamma(y)$$



The abstraction function

- The concretization function γ has an adjoint function, the abstraction function α.
- Function α maps a set of concrete values into the best representation of this set in the abstract domain (the smaller element f the abstract domain that represents of of these elements)
- In our example:,

$$\alpha : 2^{\mathrm{Int}} \to A$$

$$\alpha(S) = \mathrm{lub}\left(\left\{- \mid i < 0 \wedge i \in S\right\}, \left\{0 \mid 0 \in S\right\}, \left\{+ \mid i > 0 \wedge i \in S\right\}\right)$$

$$\sigma(i) = \alpha(\{i\})$$

A general definition

- An Abstract Interpretation consists of:
  - An abstract domain *A* and a concrete domain *D*
  - A and D are complete lattices . Smaller means “more precise”
  - Two monotone adjoint function that enjoy che formino una inserzione di Galois.
  - Abstract operations that are correct wrt the concrete ones
  - A fixpoint algorithm

- Galois insertion:

$$\forall x \in 2^D.\ \ x \subseteq \gamma(\alpha(x))$$

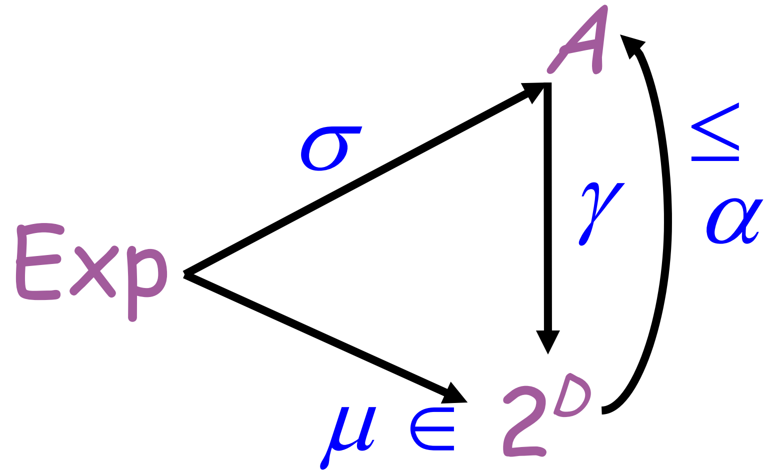
$$\forall a \in A.\ \ x = \alpha(\gamma(x))$$

Correctness revisited

- If case of Galois insertion, these correctness conditions are equivalent (prove it !)

$$\mu(e) \in \gamma(\sigma(e))$$

$$\sigma(e) \geq \alpha(\{\mu(e)\})$$



Correctness

- We show that in order to ensure the correctness of the whole analysis the following conditions are sufficient:
  1. The function α and γ are monotone
  2. The function α and γ form a Galois insertion
  3. The abstract operations are locally correct, i.e.

$$\gamma(\overline{\mathrm{op}}(s_1,\ldots,s_n)) \supseteq \mathrm{op}(\gamma(s_1),\ldots,\gamma(s_n))$$

- Notice that there is always a way to define a locally correct abstract operation. It is sufficient to consider the operations that returns the top element of the abstract domain.

Local correctness

$$\begin{tikzcd}
A \arrow[r, "\overline{op}"] \arrow[d, "\gamma"'] & A \arrow[d, "\gamma"] \\
2^{\mathbf{D}} \arrow[r, "op"] & 2^{\mathbf{D}}
\end{tikzcd}$$

$$\gamma(\overline{op}(s_1,\ldots,s_n))$$

$$\cup\!\mathsf{I}$$

$$op(\gamma(s_1),\ldots,\gamma(s_n))$$

Correctness proof

- We show by structural induction on e that:

$$\mu(e) \in \gamma(\sigma(e))$$

- Basic step:

$$\begin{aligned} & \mu(i) \\ = \quad & i \\ \in \quad & \{i\} \\ \subseteq \quad & \gamma(\alpha(\{i\})) \\ = \quad & \gamma(\sigma(\mathrm{i})) \end{aligned}$$

Correctness proof

Inductive Step

$$\mu(e) \in \gamma(\sigma(e))$$

$$\begin{aligned}
& \mu(e_1 \ \ op \ \ e_2) \\
= \ & \mu(e_1) \ \ op \ \ \mu(e_2) \\
\in \ & \gamma(\sigma(e_1)) \ \ op \ \ \gamma(\sigma(e_2)) \\
\subseteq \ & \gamma(\sigma(e_1) \ \ \overline{op} \ \ \sigma(e_2)) \\
= \ & \gamma(\sigma(e_1 \ \ op \ \ e_2))
\end{aligned}$$

Adding an input

- We can extend our tiny language with the possibility to get an input value from the user
- This means that we have a variable x in the expressions

$$e = i \mid e * e \mid -e \mid \ldots \mid x$$

Concrete semantics

- The semantic function μ becomes

$$\mu : \mathrm{Exp} \to \mathrm{Int} \to \mathrm{Int}$$

- And we may express it in terms of a family of functions, having expressions as indeces and a single parameter (the input value)

$$\begin{array}{rcl}
\mu_i(j) & = & i \\
\mu_x(j) & = & j \\
\mu_{e_1 * e_2}(j) & = & \mu_{e_1}(j) \ * \mu_{e_2}(j) \\
\mu_{e_1 + e_2}(j) & = & \mu_{e_1}(j) \ + \mu_{e_2}(j) \\
\ldots & = & \ldots
\end{array}$$

Abstract semantics

- The same holds for the abstract semantic function σ

$$\sigma : \mathrm{Exp} \rightarrow A \rightarrow A$$

- Also in this case we can express σ by a family of functions:

$$\begin{array}{lcl}
\sigma_i(\bar{j}) & = & \bar{i} \\
\sigma_x(\bar{j}) & = & \bar{j} \\
\sigma_{e_1 * e_2}(\bar{j}) & = & \sigma_{e_1}(\bar{j}) \;\bar{*}\; \sigma_{e_2}(\bar{j}) \\
\sigma_{e_1 + e_2}(\bar{j}) & = & \sigma_{e_1}(\bar{j}) \;\bar{+}\; \sigma_{e_2}(\bar{j}) \\
\ldots & = & \ldots \\
\bar{i} & = & \alpha(\{i\})
\end{array}$$

Correctness

- The following conditions are equivalent

$$\forall i.\ \ \mu_e(i) \in \gamma(\sigma_e(\alpha(\{i\})))$$

$$\mu_e \leq_D \gamma \circ \sigma_e \circ \alpha$$

$$\alpha \circ \mu_e \leq_A \sigma_e \circ \alpha$$

$$\begin{tikzcd}
A \arrow[r, "\sigma_e"] & A \arrow[d, "\gamma"] \\
2^D \arrow[u, "\alpha"] \arrow[r] & \mu_e \in 2^D \arrow[u, bend right=60, "\leq_\alpha"']
\end{tikzcd}$$

Local correctness

- We can express the local correntess condition by:

$$op\left(\gamma(\sigma_{e_1}(\overline{j})),\ldots,\gamma(\sigma_{e_n}(\overline{j}))\right) \subseteq \gamma\left(\overline{op}(\sigma_{e_1}(\overline{j}),\ldots,\sigma_{e_n}(\overline{j}))\right)$$

Conditional statement

$$e = \ldots \mid \text{if } e = e \text{ then } e \text{ else } e \mid \ldots$$

- Concrete semantics

$$\mu_{\text{if } e_1 = e_2 \text{ then } e_3 \text{ else } e_4}(i) = \begin{pmatrix} \mu_{e_3}(i) & \text{if } \mu_{e_1}(i) = \mu_{e_2}(i) \\ \mu_{e_4}(i) & \text{if } \mu_{e_1}(i) \neq \mu_{e_2}(i) \end{pmatrix}$$

- Abstract semantics

$$\sigma_{\text{if } e_1 = e_2 \text{ then } e_3 \text{ else } e_4}(\bar{i}) = \sigma_{e_3}(\bar{i}) \sqcup \sigma_{e_4}(\bar{i})$$

- Notice the role of the lub in the abstract domain

Correctness of the conditional statm.

- Assume that the condition is true (the other case is analogous)

$$
\begin{aligned}
& \mu_{e_3}(i) \\
\in\ & \gamma(\sigma_{e_3}(\bar{i})) \\
\subseteq\ & \gamma(\sigma_{e_3}(\bar{i})) \sqcup \gamma(\sigma_{e_4}(\bar{i})) \\
\subseteq\ & \gamma\left(\sigma_{e_3}(\bar{i}) \sqcup \sigma_{e_4}(\bar{i})\right) \\
=\ & \gamma(\sigma_{\text{if } e_1 = e_2 \text{ then } e_3 \text{ else } e_4}(\bar{i}))
\end{aligned}
$$