Abstract Interpretation: concrete and abstract semantics (ctd.)

Semantics of statements

We can extend the semantics of expressions considered so far to statements, mapping concrete/abstract states to concrete/abstract states.

$$\mu'_{x=e}\,(i)(\Sigma) = \Sigma\,[x/\mu_e(i)]$$

$$\mu'_{S;S'}\,(i)(\Sigma) = \mu'_{S'}(i)\,(\mu'_S\,(i)(\Sigma))$$

$$\begin{aligned}\mu'_{\text{if } e=f \text{ then } S \text{ else } S'}\,(i)(\Sigma) &= \mu'_S(i)\,(\Sigma) \quad \text{if } \mu_e(i) = \mu_f(i)\\ &= \mu'_S(i)\,(\Sigma) \quad \text{otherwise}\end{aligned}$$

$$\sigma'_{x=e}\,(\underline{i})(\Delta) = \Delta\,[x/\sigma_e(\underline{i})]$$

$$\sigma'_{S;S'}\,(\underline{i})(\Delta) = \sigma'_{S'}(\underline{i})\,(\sigma'_S\,(\underline{i})(\Delta))$$

$$\sigma'_{\text{if } e=f \text{ then } S \text{ else } S'}\,(\underline{i})(\Delta) = \text{lub}(\,\sigma'_S(\underline{i})\,(\Delta),\, \sigma'_{S'}(\underline{i})\,(\Delta))$$

Concrete and abstract semantics: while

Observe that:

while b do S

is equivalento to:

if !b skip; else {S; while b do S}

The semantics of the while statement can be expressed (both at the concrete and abstract level) as a fixpoint operator

Concrete semantics: while

$$\mu'_{\text{while } e_1=e_2 \text{ do } S}(i)(\Sigma) = \Sigma \qquad \text{if } \mu_{e_1}(i) \mathrel{!=} \mu_{e_2}(i)$$

$$= \mu'_{S;\ \text{while } e_1=e_2 \text{ do } S}(i)(\Sigma) \quad \text{otherwise}$$

$$\sigma'_{\text{while } e_1=e_2 \text{ do } S}(\underline{i})(\Delta) = \text{lub}(\Delta, \sigma'_{S;\text{while } e_1=e_2 \text{ do } S}(\underline{i})(\Delta))$$

$$= \text{lub}(\Delta, \sigma'_{\text{while } e_1=e_2 \text{ do } S}(\underline{i})(\sigma'_S(\underline{i})(\Delta))$$

$$= \text{lub}(\Delta, \text{lub}(\Delta, \sigma'_{S;\text{while } e_1=e_2 \text{ do } S}(\underline{i})(\sigma'_S(\underline{i})(\Delta)))$$

$$= \text{lub}(\Delta, \sigma'_{S;\text{while } e_1=e_2 \text{ do } S}(\underline{i})(\sigma'_S(\underline{i})(\Delta)) = \ldots$$

Recursion

- As a final step, we add recursive functions (on a single parameter)

$$\mathit{program} = \mathsf{def}\ f(x) = e$$

$$e = \ldots \mid f(e)$$

- Until now, the concrete semantics was defined as:

$$\mu : \mathsf{Exp} \rightarrow \mathsf{Int} \rightarrow \mathsf{Int}_\perp$$

Concrete semantics (function calls)

- In order to take into account the call of functions, the signature of μ becomes as folllows:

$$\mu' : \mathtt{Exp} \to (\mathtt{Int} \to \mathtt{Int}_\perp) \to \mathtt{Int} \to \mathtt{Int}_\perp$$

$$\mu'_{f(e)}(g)(j) = g(\mu'_e(g)(j))$$

$$\mu'_x(g)(j) \quad = j$$

$$\mu'_{e_1+e_2}(g)(j) = \mu'_{e_1}(g)(j) + \mu'_{e_2}(g)(j)$$

Semantics of recursive functions

$$\mu' : \text{Exp} \to (\text{Int} \to \text{Int}_\perp) \to \text{Int} \to \text{Int}_\perp$$

Consideriamo una funzione $\text{def f} = e$

Definiamo una catena ascendente $f_0, f_1, \ldots$ in $\text{Int} \to \text{Int}_\perp$

$$f_0 = \lambda x. \perp$$

$$f_{i+1} = \mu'_e(f_i)$$

Definiamo $\mu_\text{f} = \bigcup_i f_i$

def f = if x=0 then 1 else f(x - 1)

$f_0(i) = \perp$ for every i

$f_1(i) = \mu'_{\text{if } x=0 \text{ then } 1 \text{ else } f(x-1)}(f_0)(i) =$

$$\begin{cases} \mu'_1(f_0)(i) = 1 & \text{if } i=0 \\ \mu'_{f(x-1)}(f_0)(i) & \text{otherwise} \end{cases}$$

$= f_0\,(\mu'_{x-1}(f_0)(i))$

$= f_0\,(\mu'_x\,(f_0)(i) - \mu'_1\,(f_0)(i))$

$= f_0\,(i - 1)$

$= \perp$

def f = if x=0 then 1 else f(x - 1)

$f_0(i) = \perp$ for each i

$f_1(i) = 1$ if $i=0$, $\perp$ otherwise

$f_2(i) = \mu'_{\text{if } x=0 \text{ then } 1 \text{ else } f(x - 1)}(f_1)(i) =$

$$\begin{cases} \mu'_1(f_1)(i)=1 & \text{if } i=0 \\ \mu'_{f(x-1)}(f_1)(i) & \text{otherwise} \\ \quad = f_1(\mu'_{x-1}(f_1)(i)) & \end{cases}$$

$= f_1(\mu'_x(f_1)(i) - \mu'_1(f_1)(i))$

$= f_1(i - 1) = 1$ if $i=0$, and $\perp$ otherwise

def f = if x=0 then 1 else f(x - 1)

$f_0(i) = \perp$ for every i

$f_1(i) = 1$ if $i=0$, $\perp$ otherwise

$f_2(i) = 1$ if $i=0,1$, $\perp$ otherwise

$f_3(i) = 1$ if $i=0,1,2$, $\perp$ otherwise

$f_4(i) = 1$ if $i=0,1,2,3$, $\perp$ otherwise

…

$\mu(f) = \bigcup_{i>=0} f_i$

Abstract Semantics

- Tn the same way, we need to extend the definition of the abstract semantics σ.
- We require that all the operations are monotone.

$$\sigma' : \mathrm{Exp} \to (A \to A) \to A \to A$$

$$\sigma'_{f(e)}(g)(\bar{i}) = g(\sigma'_e(g)(\bar{i}))$$

$$\sigma'_x(g)(\bar{i}) \quad = \bar{i}$$

$$\sigma'_{e_1+e_2}(g)(\bar{i}) = \sigma'_{e_1}(g)(\bar{i}) + \sigma'_{e_2}(g)(\bar{i})$$

Abstract semantics of recursion

$\sigma' : \text{Exp} \rightarrow (A \rightarrow A) \rightarrow A \rightarrow A$

Consideriamo una funzione $\text{def f} = \text{e}$

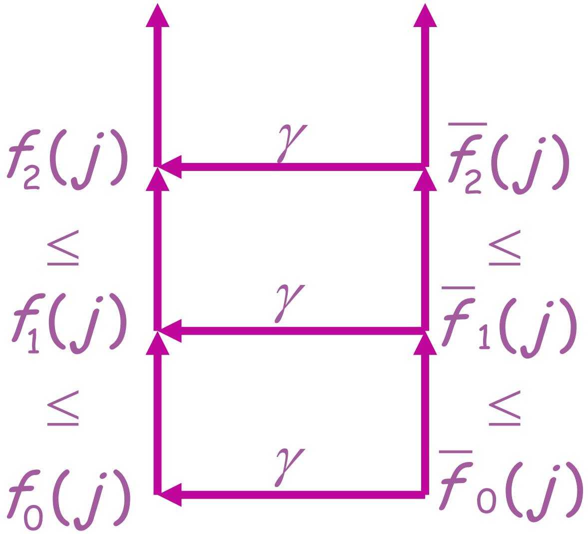
Definiamo una catena ascendente $\bar{f}_0, \bar{f}_1, \ldots$ in $A \rightarrow A$

$\bar{f}_0 = \lambda a. \perp$

$\bar{f}_{i+1} = \sigma'_e(\bar{f}_i)$

Definiamo $\sigma_{\text{f}} = \bigcup_i \bar{f}_i$

Correctness



- The abstract chain always covers the corresponding element in the concrete chain

Correctness (ctd.)

$$\forall i.\ f_i(j) \in \gamma(\overline{f}_i(\overline{j}))$$

$$\Rightarrow \quad \bigcup_{i \geq 0} f_i(j) \in \bigcup_{i \geq 0} \gamma(\overline{f}_i(\overline{j}))$$

$$\Rightarrow \quad \bigcup_{i \geq 0} f_i(j) \in \gamma\left(\bigcup_{i \geq 0} \overline{f}_i(\overline{j})\right)$$

$$\Rightarrow \quad \mu_f(j) \in \gamma(\sigma_f(\overline{j}))$$

def f = if x=0 then 1 else f(x - 1)

$f_0(a) = \perp$ for every a in Sign

$f_1(a) = \sigma'_{\text{if } x=0 \text{ then } 1 \text{ else } f(x-1)}(f_0)(a) =$

if a=0 $\sigma'_1(f_0)(a) = +$

if a=T $\mathrm{lub}(\sigma'_1(f_0)(a),\ \sigma'_{f(x-1)}(f_0)(a)) =$

$\mathrm{lub}(+, f_0(\sigma'_{x-1}(f_0)(a))) =$

$\mathrm{lub}(+, \perp) = +$

if a=+,- $\sigma'_{f(x-1)}(f_0)(a) =$

$f_0(\sigma'_{x-1}(f_0)(a)) =$

$f_0(\sigma'_x(f_0)(a) - \sigma'_1(f_0)(a)) =$

$f_0(a - +) = f_0(T) = \perp$

def f = if x=0 then 1 else f(x - 1)

$f_0(a) = \perp$ for every a in Sign

$f_1(a) = +$ for 0,T and $\perp$ for +,-

$f_2(a) = \sigma'_{\text{if } x=0 \text{ then } 1 \text{ else } f(x-1)}(f_1)(a) =$

if a=0 $\sigma'_1(f_1)(a) = +$

if a=T $\mathrm{lub}(\sigma'_1(f_1)(a),\ \sigma'_{f(x-1)}(f_1)(a)) =$

$\mathrm{lub}(+, f_1(\sigma'_{x-1}(f_0)(a))) =$

$\mathrm{lub}(+, f_1(\sigma'_x(f_1)(a) - \sigma'_1(f_1)(a))) =$

$\mathrm{lub}(+, f_1(T - +)) = \mathrm{lub}(+, f_1(T)) = +$

if a=+ $\sigma'_{f(x-1)}(f_1)(a) = f_1(\sigma'_{x-1}(f_1)(a)) =$

$f_1(\sigma'_x(f_1)(a) - \sigma'_1(f_1)(a)) =$

$f_1(+ - +) = f_1(T) = +$

if a=- $\sigma'_{f(x-1)}(f_1)(a) = f_1(\sigma'_{x-1}(f_1)(a)) =$

$f_1(\sigma'_x(f_1)(a) - \sigma'_1(f_1)(a)) =$

$f_1(- - +) = f_1(-) = \perp$

def f = if x=0 then 1 else f(x - 1)

$f_0(a) = \perp$ for every a in Sign

$f_1(a) = +$ for 0,T and $\perp$ for +,-

$f_2(a) = +$ for 0, +,T and $\perp$ for –

$f_3(a) = f_2(a)$ fixpoint

$\sigma(f) = \bigcup_{i>=0} f_i = f_2(a)$

Summary

- Abstract Interpretation means
  - Define a concrete (collecting) smenatics
  - Define an abstract semantics (domain and operations)
  - Prove the local correctness of the operations
  - Apply a fixpoint algorithm to compute the abstract semantics
  - Use a widening operator to ensure convergence (if the ascending chain condition does not hold)