Abstract interpretation

1 Widening operators

Show that the operator $\nabla$ on Interval with

$\perp \nabla X = X \nabla \perp = X$

and

$[i_1, j_1] \nabla [i_2, j_2] = \begin{cases} 
\text{if } i_2 < i_1 \text{ then } -\infty \text{ else } i_1, & \text{if } j_2 > j_1 \text{ then } \infty \text{ else } j_1 
\end{cases}$

is a widening operator. First, state precisely what you need to show, and then show that these properties are indeed fulfilled.

Solution

- $\nabla$ is an upperbound operator: Let $l_1 = [i_1, j_1], l_2 = [i_2, j_2]$.

  $i_2 < i_1, j_2 > j_1 : \quad l_1 \subseteq [-\infty, +\infty] \supseteq l_2$

  $i_2 < i_1, j_2 \leq j_1 : \quad l_1 \subseteq [-\infty, j_1] \supseteq l_2$

  $i_2 \geq i_1, j_2 > j_1 : \quad l_1 \subseteq [i_1, +\infty] \supseteq l_2$

  $i_2 \geq i_1, j_2 \leq j_1 : \quad l_1 \subseteq [i_1, j_1] \supseteq l_2$

- For all ascending chains $(l_n)_n$, the ascending chain $l_0, l_0 \nabla l_1, (l_0 \nabla l_1) \nabla l_2, \ldots$ eventually stabilizes.

  For an arbitrary element $l_0 = [n, m]$, we have to consider the following cases for $l_1 = [k, l]$:

  $k < n, l > m \Rightarrow l_0 \nabla l_1 = [-\infty, +\infty]$

  $k = n, l > m \Rightarrow l_0 \nabla l_1 = [n, +\infty]$

  $k < n, l = m \Rightarrow l_0 \nabla l_1 = [-\infty, m]$

  $k = n, l = m \Rightarrow l_0 \nabla l_1 = [n, m]$

Hence, if the chain $(l_n)_n$ eventually stabilizes, then so will the chain $(l_n^\nabla)_n$. Otherwise, it converges to the upper bound $[-\infty, +\infty]$.

2 Abstractions

Let $S$ be the set of strings over a (finite) alphabet $\Sigma$. An abstraction of the string is the set of characters/symbols of which the string is built. Example: Program analysis is abstracted by \{P, r, o, g, a, m, ’ ’, n, l, y, s, i\}.

Specify the details of the Galois connection $(\mathcal{P}(S), \alpha, \gamma, \mathcal{P}(\Sigma))$ formally. Is this Galois connection also a Galois insertion?

Solution

Let $\Sigma_s$ be the set of all of the letters that occur in a particular string. We define the abstraction and concretisation function as follows:

$\alpha(S) = \bigcup \{\Sigma_s \mid s \in S\}$

$\gamma(\sigma) = \{s \mid \Sigma_s \subseteq \sigma\}$
\(\alpha\) and \(\gamma\) are clearly monotone. Further, for a set of strings \(S = \{s_1, \ldots, s_n\}\):
\[
\gamma(\alpha(S)) = \gamma(\cup\{\Sigma_s \mid s \in S\}) = \{s' \mid \Sigma_{s'} \subseteq \cup\{\Sigma_s \mid s \in S\}\} \supseteq S
\]
and
\[
\alpha(\gamma(\sigma)) = \alpha(\{s \mid \Sigma_s \subseteq \sigma\}) = \bigcup\{\Sigma_s \mid s \in \{s \mid \Sigma_s \subseteq \sigma\}\} = \sigma
\]
Therefore, the Galois connection is also a Galois insertion.

3 Galois insertions

Let \((L_1, \alpha_1, \gamma_1, M_1)\) and \((L_2, \alpha_2, \gamma_2, M_2)\) be Galois insertions. First define
\[
\alpha(l_1, l_2) = (\alpha_1(l_1), \alpha_2(l_2))
\]
\[
\gamma(m_1, m_2) = (\gamma_1(m_1), \gamma_2(m_2))
\]
and show that \((L_1 \times L_2, \alpha, \gamma, M_1 \times M_2)\) is a Galois insertion. Then define
\[
\alpha(f) = \alpha_2 \circ f \circ \gamma_1
\]
\[
\gamma(g) = \gamma_2 \circ g \circ \alpha_1
\]
and show that \((L_1 \rightarrow L_2, \alpha, \gamma, M_1 \rightarrow M_2)\) is a Galois insertion.

Solution

We have to show that \(\alpha\) and \(\gamma\) are monotone, and that
\[
\gamma \circ \alpha \sqsupseteq \lambda l. l
\]
\[
\alpha \circ \gamma = \lambda m. m
\]

1. \(\alpha\) and \(\gamma\) are monotone, because \(\alpha_1, \alpha_2, \gamma_1, \) and \(\gamma_2\) are monotone. Further, let \(l = (l_1, l_2) \in L_1 \times L_2\).
\[
l \sqsubseteq \gamma(\alpha(l)) \iff l_1 \sqsubseteq \gamma(\alpha(l_1)) \text{ and } l_2 \sqsubseteq \gamma(\alpha(l_2))
\]
This holds because \((L_1, \alpha_1, \gamma_1, M_1)\) and \((L_2, \alpha_2, \gamma_2, M_2)\) are Galois insertions. Similarly, for \((m_1, m_2) \in M_1 \times M_2\), we have
\[
m = \alpha(\gamma(m)) \iff m_1 = \alpha(\gamma(m_1)) \text{ and } m_2 = \alpha(\gamma(m_2))
\]

2. For the first part, consider the Monotone Function Space in the book on p. 398. It remains to show that \(\alpha(\gamma(f)) = f\) for \(f \in M_1 \rightarrow M_2\):
\[
\alpha(\gamma(f)) = \alpha(\gamma_2 \circ f \circ \alpha_1) = \alpha_2 \circ \gamma_2 \circ f \circ \alpha_1 \circ \gamma_1 = id \circ f \circ id = f
\]

4 Types and Effects

Consider the following FUN program:

```fun
new_A x := 1 in
new_B y := 9 in
let f = fn z => x := !y in
let g = fn z => x := 8 in
let h = fn z => !x in
(fn w => w f + w h) (fn v => v 4)
```

What is the result of evaluating this program? What are the types and effects for the functions in this program?
Solution

The program evaluates to 18.

\[
\begin{align*}
\text{fn } z \Rightarrow x := !y & \quad \text{int} \xrightarrow{\{A := !B\}} \text{int} \\
\text{fn } z \Rightarrow x := 8 & \quad \text{int} \xrightarrow{\{A := \}} \text{int} \\
\text{fn } z \Rightarrow !x & \quad \text{int} \xrightarrow{\{!A\}} \text{int} \\
\text{fn } w \Rightarrow w f + w h & \quad (\text{int} \xrightarrow{\{A := !A !B\}} \text{int}) \xrightarrow{\{A := !A !B\}} \text{int} \\
\text{fn } v \Rightarrow v 4 & \quad (\text{int} \rightarrow \text{int}) \rightarrow \text{int}
\end{align*}
\]

5 Control Flow Analysis in a Type and Effect System

The type and effect system for Control Flow Analysis in Chapter 5.1. uses annotations \(\phi\) to denote the set of function definitions that can result in a function of a given type.

Extend the analysis with annotations for the base type \(\text{bool}\) to denote the set of constants that may be the result of evaluating the expression of a respective type.

Solution

We extend the annotations to also include boolean constants:

\[
\phi ::= \{\text{tt}\}|\{\text{ff}\} \cdots
\]

And we now also annotate the boolean type with some effect:

\[
\hat{\tau} ::= \text{bool}_\phi|\cdots
\]

Then, we can adapt the rules for the Control Flow Analysis as shown:

\[
\begin{align*}
[\text{const}_1] & \quad \hat{\Gamma} \vdash \text{true} : \text{bool}_{\{\text{tt}\}} \\
[\text{const}_2] & \quad \hat{\Gamma} \vdash \text{false} : \text{bool}_{\{\text{tt}\}} \\
[\text{if}_1] & \quad \begin{array}{c}
\hat{\Gamma} \vdash e_0 : \text{bool}_{\{\text{tt}\}} \\
\hat{\Gamma} \vdash e_1 : \hat{\tau}_1 \\
\hat{\Gamma} \vdash e_2 : \hat{\tau}_2
\end{array} \quad |\hat{\tau}_1| = |\hat{\tau}_2| \\
& \quad \hat{\Gamma} \vdash \text{if } e_0 \text{ then } e_1 \text{ else } e_2 : \hat{\tau}_1 \\
[\text{if}_2] & \quad \begin{array}{c}
\hat{\Gamma} \vdash e_0 : \text{bool}_{\{\text{tt}\}} \\
\hat{\Gamma} \vdash e_1 : \hat{\tau}_1 \\
\hat{\Gamma} \vdash e_2 : \hat{\tau}_2
\end{array} \quad |\hat{\tau}_1| = |\hat{\tau}_2| \\
& \quad \hat{\Gamma} \vdash \text{if } e_0 \text{ then } e_1 \text{ else } e_2 : \hat{\tau}_2 \\
[\text{and}_1] & \quad \begin{array}{c}
\hat{\Gamma} \vdash e_1 : \text{bool}_{\{\text{tt}\}} \\
\hat{\Gamma} \vdash e_2 : \text{bool}_{\{\text{tt}\}}
\end{array} \quad \hat{\Gamma} \vdash e_1 \& e_2 : \text{bool}_{\{\text{tt}\}}
\end{align*}
\]

All other rules are adapted similarly or remain unchanged.