Lecture notes – Correctness of abstract interpretation

September 8, 2017

Slide 3/21. Collecting semantics stated as an abstract interpretation. This slide points out that if we use the 2-state lattice, with set-union as join, and the ustate' functions as the transfer functions of statements, then the abstract JOP at each point using this 3-tuple is nothing but the collecting semantics at that point. In other words, the collecting semantics is nothing but the abstract JOP using the concrete lattice and concrete transfer functions.

Slide 4. This slide defines the notion of one abstract interpretation \((D, F_D, \gamma_D)\) being a consistent abstraction of another abstract interpretation \((C, F_C, \gamma_C)\). Here, \(F_D\) is shorthand for the set of all transfer functions in the first A.I., while \(F_C\) is shorthand for the set all transfer functions in the second framework. The definition is based on a given pair of monotonic functions \((\alpha_{CD}, \gamma_{DC})\) such that this pair of functions form a Galois connection (to be defined later). Note that \(\gamma_{DC}\) is different from \(\gamma_D\) and from \(\gamma_C\).

Consider a program \(P\) with \(N\) program points. Given lattice \(D\), \(\overline{D}\) represents a lattice of vectors, with each vector having \(N\) elements from \(D\). The \(\leq\) relation on \(\overline{D}\) is nothing but pointwise ordering; i.e., if \(\overline{d_1}, \overline{d_2} \in \overline{D}\), then \(\overline{d_1}[i] \leq \overline{d_2}[i]\) for all \(i\) in \([1...N]\), \(\overline{d_1}[i] \leq \overline{d_2}[i]\). Similarly, \(\overline{C}\) represents the vectorized version of \(C\).

Let \(c_0 \in C\) and \(d_0 \in D\) be some two elements. Let \(\text{JOP}_{\overline{D}}^{c_0}\) denote the abstract JOP at all points in \(P\) using the \(C\)-framework and using \(c_0\) as the initial abstract state. Let \(\text{JOP}_{\overline{D}}^{d_0}\) denote the abstract JOP at all points in \(P\) using the \(D\)-framework and using \(d_0\) as the initial abstract state. The definition is that the \(D\)-framework is a consistent abstraction of the \(C\)-framework under the given pair \((\alpha_{CD}, \gamma_{DC})\) if \(\text{JOP}_{\overline{D}}^{c_0} \leq \gamma_{DC}(\text{JOP}_{\overline{D}}^{d_0})\) for any \(c_0\) and \(d_0\) chosen such that \(\gamma_{DC}(d_0) \geq c_0\). Intuitively, what this means is that at any program point the \(\gamma_{DC}\) image of the JOP computed using the \(D\)-framework dominates the JOP computed at the same point using the \(C\)-framework.

The \(C\)-framework is said to be more precise than the \(D\)-framework under \((\alpha_{CD}, \gamma_{DC})\) iff the \(D\)-framework is a consistent abstraction of the \(C\)-framework under \(\gamma_{DC}\).

(From here on, we use \(\gamma\) to mean \(\gamma_{DC}\) and \(\alpha\) to mean \(\alpha_{CD}\).)

Slide 5. The notion of two functions \((\alpha, \gamma)\) forming a Galois Connection is defined here.

Generally \(C\) is a larger (more precise) lattice than \(D\). The basic property that the Galois Connection enforces is that each element of \(D\) represents one or more elements of \(C\). Each element \(c_1\) of \(C\) that is in the range of \(\gamma_{DC}\) is represented precisely by the element \(\alpha(c_1)\), as \(\gamma(\alpha(c_1)) = c_1\). On the other hand, each element \(c_2\) of \(C\) that is not in the range \(\gamma_{DC}\) is represented imprecisely (i.e., over-approximated) by \(\alpha(c_2)\), as \(\gamma(\alpha(c_2)) > c_2\).

Slide 6. The notion of two functions \((\alpha, \gamma)\) forming a Galois Connection is defined here.

It can be shown that the given \((\alpha, \gamma)\) for this illustration form a Galois connection.

Note, to assert this consistent abstraction property, we need to also define the transfer functions for both domains. The \(f_{L_1}\) transfer functions are as defined in Slide 5 of the “Abstract interpretation” slides. For any statement \(n\), the transfer function \(f_{L_2}^n\) can be defined as follows:

\[
\gamma(f_{L_2}^n(S)) = \{(x, y) | (x', y') \in S, (x, y) \in \gamma(f_{L_1}^n(x', y'))\}
\]

Note, in the “Introduction” slides we showed a program where this \(L_2\) interpretation produces more pre-
cise results than the $L_1$ interpretation.

**Slide 9.** The first bullet brings out the relationship between the definitions of consistent abstraction and correctness. (Correctness of an abstract interpretation framework was defined earlier in the “Introduction to abstract interpretation” slides.)

The second bullet points out that “consistent abstraction of” is a transitive relation if the $\gamma$’s are monotonic.

**Slide 10.** This slide gives the sufficient condition under which one A.I. is a consistent abstraction of another A.I. This is the main theorem of abstract interpretation.

Say a designer has proposed a new abstract lattice $D$, and a set of $F_D$ transfer functions based on $D$. In order to prove that this proposed abstract interpretation is a consistent abstraction of some existing abstraction interpretation based on a lattice $C$, the designer of the $D$-abstract interpretation should prove that the $F_D$ transfer functions that they have provided are abstractions of the corresponding $F_C$ transfer functions. Note, if the designer wants to prove that the $D$-abstract interpretation is correct, then they need to show that the $F_D$ transfer functions are abstractions of the corresponding nstate$^\gamma$ transfer functions.

**Slide 11.**

Lemma 2 intuitively states that by taking a join in the $D$ lattice we are over-approximating a corresponding join in the $C$ lattice. This imprecision is inevitable because the $D$ lattice is smaller than the $C$ lattice.

**Slide 14.** The basic requirement, intuitively, is that for any pair of adjacent program points $M, N$, $f_{D,MN}$ over-approximates $f_{C,MN}$. This can be stated in two ways, that are equivalent to each other:

- for any $c \in C$, $\alpha(f_{C,MN}(c)) \leq f_{D,MN}(\alpha(c))$
- for any $d \in D$, $f_{C,MN}(\gamma(d)) \leq \gamma(f_{D,MN}(d))$

The rest of slides are straightforward to follow.