Designing an abstract interpretation-based analysis

Computer Science and Artificial Intelligence Laboratory
MIT

Nov 13, 2011
Consider the following program

```java
n = pos_input();
if(n > 25){
    n = 25;
}
A = new int[n];
x=0
t= __input();
v= t*t + 1;
while(x < n){
    A[x] = x;
    x = x + v;
}
```

Can x ever be less than zero?
Can x ever be > 25?
Can x ever be > n inside the loop?
Consider the following program

```java
n = pos_input();
t = n-2;
A = new int[n];
x=0
t= __input();
v= t*t + 1;
while(x < n){
    A[x] = x;
    x = x + v;
}
```
Concrete Interpretation

Denotation Functions
- Describe the meaning of the program as a function
- $\llbracket \text{Exp} \rrbracket : \Sigma \rightarrow \mathcal{V}$
- $\llbracket \text{Stmt} \rrbracket : \Sigma \rightarrow \Sigma$
- Just another way of expressing the program semantics

\[
\begin{align*}
\llbracket n \rrbracket (\sigma) &= n \\
\llbracket x \rrbracket (\sigma) &= \sigma(x) \\
\llbracket e_1 + e_2 \rrbracket (\sigma) &= \llbracket e_1 \rrbracket (\sigma) + \llbracket e_2 \rrbracket (\sigma) \\
\llbracket x = e \rrbracket (\sigma) &= \sigma[x \rightarrow \llbracket e \rrbracket (\sigma)]
\end{align*}
\]
Designing an abstract domain

Define the abstraction and concretization functions

- $\beta: V \rightarrow Abs$
- $\alpha: P(V) \rightarrow Abs$
  - $\alpha(S) = \bigcup_{s \in S} \beta(s)$
- $\gamma: Abs \rightarrow P(V)$
- Decision 1: States or individual values?

Make sure they form a Galois connection

- Both are monotonic
- $\alpha(\gamma(a)) = a$
- $\gamma(\alpha(V)) \supseteq V$ for $V \in P(V)$
Designing an Abstract Interpretation

For every expression and statement, define how to operate in the abstract domain

- $\mathcal{F} [Exp]: \Sigma^\# \rightarrow \text{Abs}$
- $\mathcal{F} [Stmt]: \Sigma^\# \rightarrow \Sigma^\#$
- If you defined the abstract domain on states, $\bar{\sigma} \in \Sigma^\#$ is an abstract state
- An abstract domain on values can be lifted to an abstract domain on states
  - How?
Key property

For expressions

Let $\sigma \in \gamma(\bar{\sigma})$

And similarly for statements
A general definition

\[ \mathcal{I} [e](\bar{\sigma}) = \alpha( \{ v \mid v = \llbracket e \rrbracket \sigma \land \sigma \in \gamma(\bar{\sigma}) \} ) \]

Sometimes it is hard to compute this directly

- When in doubt, approximate!
Ideas for creating domains

- Start from known domains

- Products

- Predicate Abstraction
Key design decisions

Should you be aware of branch conditions?

Should you widen?