cs611 Lecture 37 **Abstract Interpretation** 2 December 2002

Scribe: Kamal Aboul-Hosn, Stan Dziobiak, Xiangyang Lan, and Zoya Svitkina Lecturer: Radu Rugina

1 Sign Analysis

Consider the desire to know the possible signs of variables in a program at particular points in the execution. We can perform a static sign analysis on the program, where the information we get is a member of the $\mathcal{P}(\mathsf{Sign})$, where $\mathsf{Sign} = \{+, 0, -\}$.

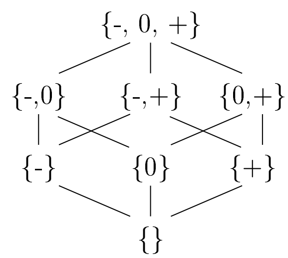
While such an analysis will not tell us exactly what the value of a variable is at a given point, it will still give us information about the range of values, i.e. if a variable X has possible signs $\{0, +\}$, we know $X \geq 0$. The abstract information we get is of the form

$$L_a \quad = \quad \mathsf{Var} \to \mathcal{P}(\mathsf{Sign})$$

We can use this to create lattices:

$$(\mathcal{P}(\mathsf{Sign}), \subseteq)$$
$$(\mathsf{Var} \to \mathcal{P}(\mathsf{Sign}), \sqsubseteq)$$

We can represent the first lattice as a diagram



Note that the reverse diagram is equally valid, however, the approximation is more precise as one moves towards the top.

The concrete information can be represented as

$$L_c \quad = \quad \mathsf{Var} \to \mathcal{P}(\mathbb{Z})$$

where $\mathcal{P}(\mathbb{Z})$ contains the sets of actual possible values a variable can take on.

Consider an example of the concrete and abstract analysis on the following IMP code:

```
x := 0;
n := 1;
while (n ≤ m) {
    x := x + n;
    n := n + 1;
}
```

Our analyses would proceed as follows

Concrete execution		**Abstract execution**
$\{x \to \mathbb{Z}, n \to \mathbb{Z}\}$		$\{x \to \{+,-,0\}, n \to \{+,-,0\}\}$
	`[x := 0]`	
$\{x \to \{0\}, n \to \mathbb{Z}\}$		$\{x \to \{0\}, n \to \{+,-,0\}\}$
	`[n := 1]`	
$\{x \to \{0\}, n \to \{1\}\}$		$\{x \to \{0\}, n \to \{+\}\}$
	`[n ≤ m]`	
$\{x \to \{0\}, n \to \{1\}\}$		$\{x \to \{0\}, n \to \{+\}\}$
	`[x := x + n]`	
$\{x \to \{1\}, n \to \{1\}\}$		$\{x \to \{+\}, n \to \{+\}\}$
	`[n := n + 1]`	
$\{x \to \{1\}, n \to \{2\}\}$		$\{x \to \{+\}, n \to \{+\}\}$

At this point, we must reconsider the analysis for the loop, merging our results from one execution of the loop and no executions of the loop.

Concrete execution		**Abstract execution**
$\{x \to \{0,1\}, n \to \{1,2\}\}$		$\{x \to \{0,+\}, n \to \{+\}\}$
	`[n ≤ m]`	
$\{x \to \{0,1\}, n \to \{1,2\}\}$		$\{x \to \{0,+\}, n \to \{+\}\}$
	`[x := x + n]`	
$\{x \to \{1,2,3\}, n \to \{1,2\}\}$		$\{x \to \{+\}, n \to \{+\}\}$
	`[n := n + 1]`	
$\{x \to \{1,2,3\}, n \to \{2,3\}\}$		$\{x \to \{+\}, n \to \{+\}\}$

At this point, our abstract analysis can stop, because we reached a fixed point. Such an analysis will always terminate at a fixed point, since the lattice is finite. The same is not true of the concrete analysis.

1.1 Transfer functions

Consider the function

$$[\![X := a]\!] \quad : \quad L_a \to L_a$$

The transfer function takes a mapping of variables to signs and returns a mapping from variables to signs. The transfer function $[\![s]\!]$ is monotonic if $\forall x, y \in L_a, x \sqsubseteq y \Rightarrow [\![s]\!]x \sqsubseteq [\![s]\!]y$. For an assignment, we have

$$[\![X := a]\!]s \quad = \quad s[x \to sign(a, s)]$$

We must define the *sign* function over all arithmetic expressions, which can be of the form

$$a \quad ::= \quad a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1 \mid n \mid X$$

The *sign* function is defined as follows

$$\begin{array}{rcl} sign(a,s) \in \mathcal{P}(\mathsf{Sign}) & & \\ sign(X,s) & = & s(X) \\ sign(n,s) & = & \begin{cases} \{+\} & \text{if } n > 0 \\ \{0\} & \text{if } n = 0 \\ \{-\} & \text{if } n < 0 \end{cases} \\ sign(a_0 + a_1, s) & = & sign(a_0, s) +_s sign(a_1, s) \\ sign(a_0 - a_1, s) & = & sign(a_0, s) -_s sign(a_1, s) \\ sign(a_0 \times a_1, s) & = & sign(a_0, s) \times_s sign(a_1, s) \end{array}$$

⟩From here, we must define the $+_s, -_s$, and $\times_s$ operators, which return the possible signs of the result given the possible signs of the operands. For instance,

$$+_s : \mathcal{P}(\mathsf{Sign}) \times \mathcal{P}(\mathsf{Sign}) \to \mathcal{P}(\mathsf{Sign})$$

We define the $+_s$ operator in terms of an operator, $\oplus$, which looks at operands of a specific sign

$$\oplus : \mathsf{Sign} \times \mathsf{Sign} \to \mathcal{P}(\mathsf{Sign})$$

Now, we define $+_s$ as

$$s_1 +_s s_2 = \bigcup_{(x \in s_1, y \in s_2)} x \oplus y$$

The result of $\oplus$, depending on the signs of the operands, is summarized in this table

$\oplus$	$-$	0	$+$
$-$	$\{-\}$	$\{-\}$	$\{-,0,+\}$
0	$\{-\}$	$\{0\}$	$\{+\}$
$+$	$\{-,0,+\}$	$\{+\}$	$\{+\}$

We define $\times_s$ similarly, basing its result on the result of an operand $\otimes$, defined as follows

$\otimes$	$-$	0	$+$
$-$	$\{+\}$	$\{0\}$	$\{-\}$
0	$\{0\}$	$\{0\}$	$\{0\}$
$+$	$\{-\}$	$\{0\}$	$\{+\}$

1.2 Concrete vs. Abstract Domains

We can define an abstraction function $\alpha : L_c \to L_a$, which takes our concrete values and returns the signs.

$$\begin{aligned} \alpha(S_c) &= \lambda x \in Var.\{sign(n) \mid n \in S_c(x)\} \\ sign(n) &= \begin{cases} + & \text{if } n > 0 \\ 0 & \text{if } n = 0 \\ - & \text{if } n < 0 \end{cases} \end{aligned}$$

We can also define a concretization function $\gamma : L_a \to L_c$, which returns the possible concrete values given our abstract (sign) value. We define the function as

$$\gamma(S_a) = \lambda x \in \mathsf{Var}.\{n \in \mathbb{Z} \mid sign(n) \in S_a(x)\}$$

These two functions have the following properties:

- α, γ are both monotonic
- $\forall x \in L_c, x \sqsubseteq \gamma(\alpha(x))$
  $\forall y \in L_a, y \sqsupseteq \alpha(\gamma(y))$

Note that the $(L_c, \alpha, \gamma, L_a)$ satisfying these properties is a Galois connection. Although moving between L_c and L_a is safe, we may lose precision, as can be seen in the following example:

$$\begin{aligned} \mathsf{Var} &= \{x\} \\ S_c &= (x, \{0,1\}) \\ \alpha(S_c) &= (x, \{0,+\}) \\ \gamma(\alpha(S_c)) &= (x, \{n \mid n \geq 0\}) \end{aligned}$$

$$\begin{aligned} S_a &= (x, \{0,+\}) \\ \gamma(S_a) &= (x, \{n \mid n\rangle = 0\}) \\ \alpha(\gamma(S_a)) &= (x, \{0,+\}) \end{aligned}$$