Lecture Notes on
More Abstract Interpretation

15-411: Compiler Design
André Platzer

Lecture 29

1 Introduction

In this lecture, we continue our short overview of abstract interpretation uses and their connections with compilation and program analysis. This is a wide field and easily the topic of a whole semester. More information on abstract interpretation can be found in [CC92, CC77, CC79] and [WM95, Chapter 10]. Simple examples of abstract interpretation type ideas in more classical situations include sign abstraction of values into $\{-, 0, +, ?\}$ or abstraction of values by remainders mod 4 [WM95, Chapter 10].

2 Abstract Interpretation by Example

Consider the following simple program

```
0
1  x = 1
2
3  while (x<1000) {
4
5      x = x + 1
6
7  }
8
9  y = x
```

A run in the concrete semantics of the above program would start with the concrete state $x = \bot, y = \bot$ where the initial value of x, y in line 0 is unknown. The program would do 999 iterations through the loop after which it terminates with the state $y = x = 1000$. Concrete execution just does not help much for static analysis of programs in general, because we won't know the dynamic data until runtime.

Instead, let us consider an abstract run in an abstract semantics where variables take on intervals as values (due to Cousot and Cousot [CC77]):

$$L = \{[a, b] \ : \ a, b \in \mathbb{N} \cup \{+\infty, -\infty\}\}$$

To unify notation, we write $[-\infty, 5]$ for the left-open interval $(-\infty, 5]$ here. Now a run of the above program in the interval abstract domain gives after 1 iteration

```
{$x = [-\infty, \infty], y = [-\infty, \infty]$}
x = 1
{$x = [1, 1], y = [-\infty, \infty]$}
while (x<1000) {
    {$x = [1, 1], y = [-\infty, \infty]$}
    x = x + 1
    {$x = [2, 2], y = [-\infty, \infty]$}
}

y = x
```

and after 2 iterations

```
{$x = [-\infty, \infty], y = [-\infty, \infty]$}
x = 1
{$x = [1, 1], y = [-\infty, \infty]$}
while (x<1000) {
    {$x = [1, 2], y = [-\infty, \infty]$}
    x = x + 1
    {$x = [2, 3], y = [-\infty, \infty]$}
}

y = x
```

and after 3 iterations

```
{$x = [-\infty, \infty], y = [-\infty, \infty]$}
x = 1
{$x = [1, 1], y = [-\infty, \infty]$}
while (x<1000) {
```

```
4       ${x = [1,3], y = [-\infty,\infty]}$
5       x = x + 1
6       ${x = [2,4], y = [-\infty,\infty]}$
7   }
8
9   y = x
```

We could keep on iterating, but this takes an awfully large number of iterations to figure out, since the loop count is 1000. If the bound is not computable statically, we do not even know how often to iterate. But we can iterate until we reach a fixedpoint. And we can also speed up convergence by jumping ahead in the lattice using a widening operator $\nabla : L \times L \to L$. For intervals let us jump ahead to ±∞ whenever our interval bounds are not inclusive:

$$[a,b] \mathbin{\nabla} [a',b'] \;:=\; \left[\begin{Bmatrix} a & \text{if } a \le a' \\ -\infty & \text{otherwise} \end{Bmatrix}, \begin{Bmatrix} b & \text{if } b' \le b \\ +\infty & \text{otherwise} \end{Bmatrix}\right]$$

So in the 4th iteration, instead of doing a standard iteration, let us widening for computing line 4 from the previous two values $[1,3] \mathbin{\nabla} [1,4]$:

```
0   ${x = [-\infty,\infty], y = [-\infty,\infty]}$
1   x = 1
2   ${x = [1,1], y = [-\infty,\infty]}$
3   while (x<1000) {
4       ${x = [1,\infty], y = [-\infty,\infty]}$
5       x = x + 1
6       ${x = [2,\infty], y = [-\infty,\infty]}$
7   }
8
9   y = x
```

In iteration 5, we obtain precise information by intersection with the guards

```
0   ${x = [-\infty,\infty], y = [-\infty,\infty]}$
1   x = 1
2   ${x = [1,1], y = [-\infty,\infty]}$
3   while (x<1000) {
4       ${x = [1,999], y = [-\infty,\infty]$  since  $x = [1,\infty] \cap [1,999] = [1,999]}$
5       x = x + 1
6       ${x = [2,1000], y = [-\infty,\infty]}$
7   }
8   ${x = [1000,1000], y = [-\infty,\infty]$  since  $x = [2,1000] \cap [1000,\infty] = [1000,1000]}$
```

```
y = x
$\{x = [1000, 1000], y = [1000, 1000]\}$
```

What we want the widening operator ∇ to satisfy is that it is like a union ($\cup$) but could be a bigger element of the lattice:

$$x \leq x \nabla y \quad y \leq x \nabla y$$

We also want iterated uses of the widening operator to become a fixedpoint eventually. That is

$$x_0 \nabla x_1 \nabla x_2 \nabla x_3 \nabla \ldots$$

is a finite sequence, for any $x_i \in L$.

This seems very powerful and it is, as a framework for static program analysis. The particular abstract domain of intervals alone, however, is insufficient. A simple variation of the above example shows that the example is misleading and real programs more complicated:

```
$\{x = [-\infty, \infty], y = [-\infty, \infty]\}$
x = 1
$\{x = [1, 1], y = [-\infty, \infty]\}$
y = 1
$\{x = [1, 1], y = [1, 1]\}$
while (x<1000) {
    $\{x = [1, 999], y = [1, \infty]$  since  $x = [1, \infty] \cap [1, 999] = [1, 999]\}$
    x = x + 1
    $\{x = [2, 1000], y = [1, \infty]\}$
    y = y + 1
    $\{x = [2, 1000], y = [2, \infty]\}$
}
$\{x = [1000, 1000], y = [1, \infty]$  since  $x = [2, 1000] \cap [1000, \infty] = [1000, 1000]\}$
```

This result is perfectly correct but rather useless as far as y is concerned, because it does not constrain the values of y, except for positivity, simply because y did not occur directly in the loop exit condition.

But the abstract interpretation framework still applies. Abstract domains that can handle the above example need correlations of variables, i.e, they need to capture variable correlations like $0 \leq x - y \leq 1$. Difference-bounds matrix [Min01] are a fast abstract domain for this purpose. General convex polyhedra can be useful too. This is possible but out of scope for this lecture. We only show the cheaper difference logic, where a fast implementation are difference-bounds matrices. We adjoin an extra information of difference-bounds to the abstract domain L. As an optimization, we simply bootstrap from the converged values of x and y in our interval domain, since those would be found after some number of iterations anyhow. Better values are possible now, but not worse values. First, we need to figure out what the effect of the assignment $x = x + 1$ will be on the abstract value $l \leq x - y \leq u$ in the difference-bounds.

$$l \leq x - y \leq u \quad \overset{x:=x+1}{\leadsto} \quad l + 1 \leq \underbrace{(x+1)}_{x_{new}} - y \leq u + 1$$

Similarly for the assignment $y = y + 1$:

$$l \leq x - y \leq u \quad \overset{y:=y+1}{\leadsto} \quad l - 1 \leq x - \underbrace{(y+1)}_{y_{new}} - y \leq u - 1$$

After the first iteration, we get

```
 0  {$x = [-\infty, \infty], y = [-\infty, \infty], \infty \leq x - y \leq \infty$}
 1  x = 1
 2  {$x = [1, 1], y = [-\infty, \infty], \infty \leq x - y \leq \infty$}
 3  y = 1
 4  {$x = [1, 1], y = [1, 1], 0 \leq x - y \leq 0$}
 5  while (x<1000) {
 6      {$x = [1, 999], y = [1, \infty], 0 \leq x - y \leq 0$}
 7      x = x + 1
 8      {$x = [2, 1000], y = [1, \infty], 1 \leq x - y \leq 1$}
 9      y = y + 1
10      {$x = [2, 1000], y = [2, \infty], 0 \leq x - y \leq 0$}
11  }
12  {$x = [1000, 1000], y = [1000, 1000], 0 \leq x - y \leq 0$}
```

At which the fixpoint is reached immediately. Note that line 4 uses that the abstract value $x = [1, 1], y = [1, 1]$ in the interval domain is communicated to the best corresponding constraint expressible as difference bounds: $0 \leq x - y \leq 0$:

$$x = [a, b], y = [c, d] \quad \leadsto \quad a - d \leq x - y \leq b - c$$

Hence, it is important that the abstract domains "talk" to each other. Conversely, in line 12, the abstract state $0 \leq x - y \leq 0$ in the difference bounds can "talk" to the interval domain and synchronize to the best constraint that follows from the difference bounds in combination with the known individual interval bounds as follows:

$$x = [a, b], y = [c, d], l \leq x - y \leq u \quad \rightsquigarrow \quad x = [\max(a, c + l), \min(b, d + u)]$$

since $l \leq x - y$ implies $x \geq y + l$, yet $y \geq c$. Similarly $x - y \leq u$ implies $x \leq y + u$ with $y \leq d$.

When widening was too aggressive, a dual operator called narrowing $\Delta : L \times L \to L$ can be used as well. It is supposed to be like an intersection ($\cap$) but could be bigger:

$$x \cap y \leq x \Delta y$$

We also want iterated uses of the widening operator to become a fixedpoint eventually. That is

$$x_0 \Delta x_1 \Delta x_2 \Delta x_3 \Delta \ldots$$

is a finite sequence, for any $x_i \in L$.

Quiz

1. What would happen if you had initialized $x = [-\infty, \infty]$ everywhere to express that you don't know initially what value x would have? Would that be the same as initializing $x = \bot$?
2. Can abstract interpretation with interval bounds be used to perform analysis for possible occurrences of divisions by zero?
3. Show how abstract interpretation with the interval bounds domain can be used to perform array bounds checking optimizations.
4. To convince yourself under which circumstance narrowing Δ may become necessary after widening, consider the example

```

x = 1

while (x<1000) {

    x = x + 1

    if (x > 20) break;

}
```

5. Define a narrowing operator Δ for the above case and show how to use it successfully.

References

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[CC79] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282, 1979.

[CC92] Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *J. Log. Program.*, 13(2&3):103–179, 1992.

[Min01] Antoine Miné. A new numerical abstract domain based on difference-bound matrices. In Olivier Danvy and Andrzej Filinski, editors, *PADO*, volume 2053, pages 155–172. Springer, 2001.

[WM95] Reinhard Wilhelm and Dieter Maurer. *Compiler Design*. Addison-Wesley, 1995.