Software Verification

Bounded Box

Mattia Bottaro - Mauro Carlin
May, 2018

Università degli Studi di Padova

Outline

Project - Domain

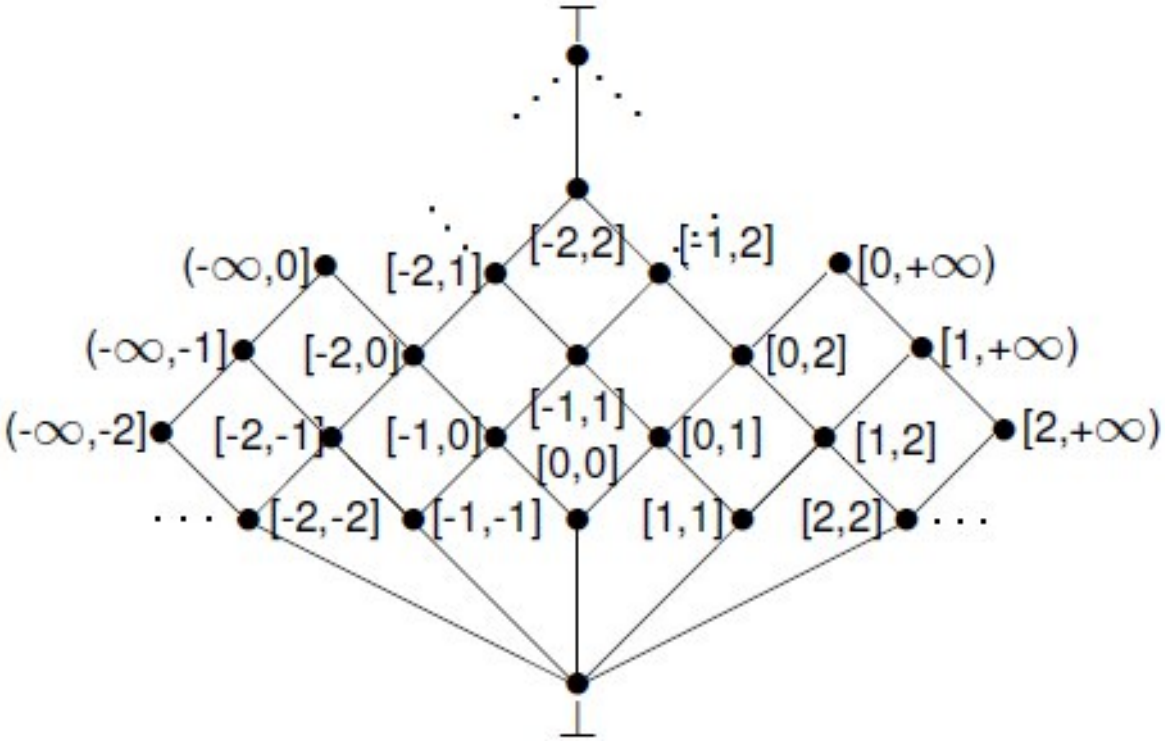
We have chosen the **Bounded Box Domain**, which is a parametric restriction of the interval abstract domain *Int*:

Given $m, n \in \mathbb{Z} \cup \{-\infty, +\infty\}$, then

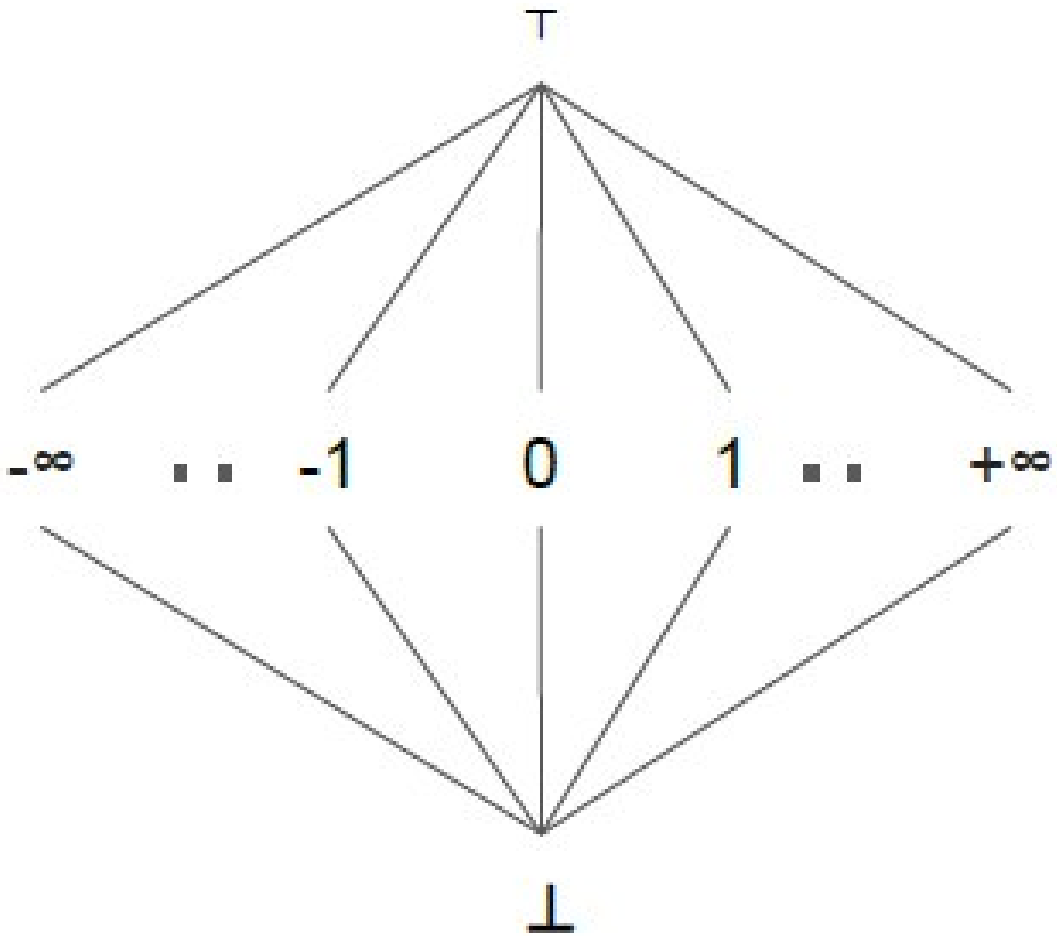
$$\mathbf{Int}_{m,n} := \{\varnothing, \mathbb{Z}\} \cup \{[k,k] \mid k \in \mathbb{Z}\} \cup \{[a,b] \mid a < b,\ [a,b] \subseteq [m,n]\} \cup \{(-\infty, k] \mid k \in [m,n]\} \cup \{[k, +\infty) \mid k \in [m,n]\}$$

Project - Domain

Bounded Box with m = -2, n = 2



Bounded Box with m > n

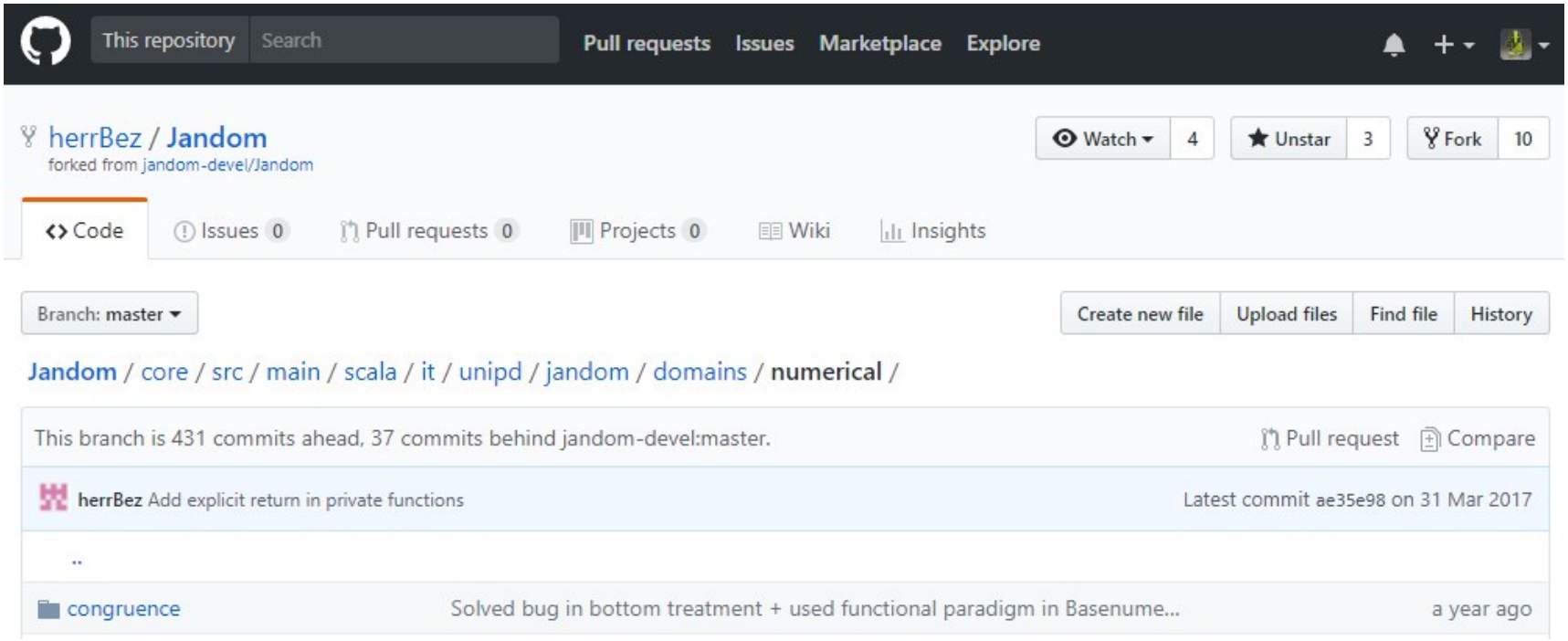


Project - Jandom

- Static Analyzer for **Numerical** and Object Domains (forse togliere object)
- Jandom was created at the University of Chieti-Pescara
- It's a buildup of **RANDOM**, which analyzes **R** code
- Jandon is written in **Scala**
- Jandom analyzes JVM bytecode using **Soot**

Project - Jandom

We've extended Jandom from this repository created by University of Padua' students



Our Contribution

We have:

1. Implemented the Integer Interval Domain
2. Implemented Bounded Box Domain specializing the previous domain, because abstract operators of both domains are very similar

Our Contribution

Abstract **sum** operator algorithm in Bounded Box Domain.

1. Execute sum operator of Interval Domain.
   $[a, b] +_b^{\#} [c, d] = [a + c, b + d] = [e, f]$
2. $[e, f]$ must be represented as an element of Bounded Box Domain

$$[e, f] = \begin{cases} \top^{\#} & e < m \wedge f > n \\ [n, +\infty) & e \geq n \wedge e \neq f \\ [e, +\infty) & e < n \wedge f > n \\ (-\infty, m] & f \leq m \wedge e \neq f \\ (-\infty, f] & f > m \wedge e < m \\ [e, f] & otherwise \end{cases}$$

Our Contribution

Abstract **reminder** operator algorithm in Box Domain

$$[a,b]\%_b^{\#}[c,d] = \begin{cases} \top^{\#} & [c,d] = \top^{\#} \\ \bot^{\#} & [a,b] = \bot^{\#} \vee [c,d] = \bot^{\#} \vee [c,d] = [0,0] \\ [0,0] & [a,b] = [0,0] \\ [0,d-1] & c \geq 0 \\ [c+1,0] & d \leq 0 \\ [c+1,d-1] & \textit{otherwise} \end{cases}$$

Our contribution

We have defined a new type, called *InfInt*, to:

1. model infinity values with Integer type
2. overload operations between integer number
3. simplify further contribution

Example

$(+\infty) + n = +\infty$
$(+\infty) \times (-\infty) = -\infty$
$(+\infty) \div (+\infty) = 0$

First example


```
int i0, i1, i2, i3;

/*[ [ i0 = T , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        i0 = -5;

/*[ [ i0 = [-5,-5] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        i1 = 15;

    label1:
/*[ [ i0 = [-5,6] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        if i0 > 5 goto label2;

/*[ [ i0 = [-5,5] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        i1 = i0 * i1;

/*[ [ i0 = [-5,5] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        i0 = i0 + 1;

/*[ [ i0 = [-4,6] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        goto label1;

    label2:
/*[ [ i0 = [5,6] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        if i0 < 7 goto label3;

/*[ [ empty ]types: int,int,int,int ]*/
        i2 = i0 - 10;

/*[ [ empty ]types: int,int,int,int ]*/
        goto label4;

    label3:
/*[ [ i0 = [5,6] , i1 = T , i2 = T , i3 = T ]types: int,int,int,int ]*/
        i3 = i0 + 10;

    label4:
/*[ [ i0 = [5,6] , i1 = T , i2 = T , i3 = [10,+] ]types: int,int,int,int */
```