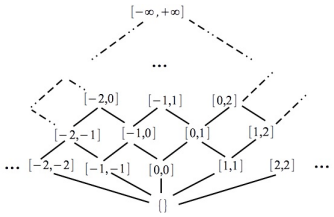
Homework 4

Ahmed Rezine

Problem 1:



We adopt the following widening operator ∇ for the interval domain:

- $[a,b]\nabla\bot = \bot\nabla[a,b] = [a,b]$
- $[a,b]\nabla[c,d] = [l,r]$ with
  - $l = a$ if $a \leq c$ and $l = -\infty$ otherwise
  - $r = b$ if $b \geq d$ and $r = +\infty$ otherwise

- Give a sequence of intervals obtained during a fixpoint computation where you systematically use widening as a join operator.
- (optional) The obtained fixpoint does not establish that `x <= 10` at line `L3`. Describe how such a fact can be established using the interval domain.

```
//x: ⊤ = [−oo, +oo]
L1. x:= 0
//x: ⊥
L2. x:= x + 1
//x: ⊥
L3. nop
//x: ⊥
L4. if x < 10 goto L2
//x: ⊥
L5. end
```

Problem 2:

Check the `README.md` file under the cpachecker folder (see course page) and go through `https://sosy-lab.gitlab.io/research/tutorials/CPAchecker`.

- Follow the tutorial for the two examples (example.c and example_bug.c) with the default configurations and properties. Use the `predicateAnalysis` configuration instead of the default one. Checkout the `output/abstractions.txt` file.
- Write your own version of the lock program from the lecture and verify it using the `predicateAnalysis` for predicate abstraction. Checkout the `output/abstractions.txt` file.
- Choose a programs from `https://github.com/sosy-lab/sv-benchmarks/tree/master/c/loop-crafted` (the sv-benchmarks are used in the software verification competition `https://sv-comp.sosy-lab.org/2025/`). Understand what the program is about and try to verify using some CPAchecker configuration.
- (Optional) Write a program of your own (of the same level of complexity as example.c). Try to verify it with CPAchecker. Introduce a bug and try to get CPAchecker to exhibit an error trace.