Assignment 4
Constant propagation analysis by abstract interpretation

Static Program Analysis and Constraint Solving
Master's Degree in Formal Methods in Computer Science
Year 2021/22

Submission deadline: November, 9th

Submission instructions: Students are required to submit a `.pdf` file with the proofs and definitions corresponding to this assignment. Submissions can be written in English or Spanish. Handwritten submissions are accepted (assuming clear handwriting!).

A *constant propagation analysis* tries to determine, at each program point, whether a variable will always have the same value at that point and, in case it does, it tells us which value. For example, given the following program:

```
z := 4;
n := 0;
x := 0;
while x <= 10 do
    v := z * z;
    n := x + v;
    x := x + 1
```

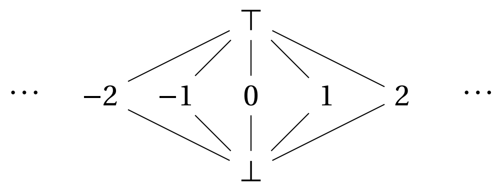
We know that, after the assignment `v := z * z`, the variable `z` always contains the value 4, since once `z` is assigned to at the beginning of the program, it is no longer modified. We also know that `v` always contains 16 after that assignment, since it depends only on `z`. Neither `n` nor `x` are constant after that assignment, since their value changes at each iteration of the loop.

Your goal is to prove the correctness of constant propagation analysis in terms of the interval analysis explained in this lesson.

Let us define the set of abstract values $\mathbb{Z}_\bot^\top = \mathbb{Z} \cup \top, \bot$, with the $\sqsubseteq$ order relation defined as follows:

$$\forall t_1, t_2 \in \mathbb{Z}_\bot^\top. \quad t_1 \sqsubseteq t_2 \iff t_1 = \bot \lor t_2 = \top \lor t_1 = t_2$$

This relation can be depicted by the following Hasse diagram:



An abstract state $\sigma^\sharp$ is a function $\mathbf{Var} \to \mathbb{Z}_\perp^\top$, where **Var** is the set of variables occurring in the program. A constant propagation analysis infers an abstract state for each program point. This abstract state maps a variable to $\top$ if that variable is not guaranteed to have the same value at that point, and to a given integer k if that variable will always have the value k at that point. In the example above, a constant propagation analysis would infer the abstract state $[x \mapsto \top, n \mapsto \top, z \mapsto 4, v \mapsto 16]$ right after the statement `v := z * z`.

In this assignment you have to develop a simple analysis that determines, given an expression e and an input abstract state $\sigma^\sharp$, whether the expression will be always evaluated to the same value at runtime. For example, given $\sigma^\sharp = [x \mapsto 2, v \mapsto \top]$, the expression $x + 4$ will always be evaluated to 6, whereas the expression $x * v$ is not guaranteed to be constant, since neither is v.

It is not difficult to become convinced that a constant propagation analysis is a particularization of an interval analysis. Indeed, if our interval analysis determines at a given point that the value of a variable x lies within an interval $[k, k]$ for some $k \in \mathbb{Z}$, we know that x is constant at that point. Therefore, in this assignment we shall consider that the **Interval** set (see slide 178) is our concrete domain, whereas $\mathbb{Z}_\perp^\top$ is our abstract domain.

1. Define an abstraction function and a concretization function between **Interval** and $\mathbb{Z}_\perp^\top$,

$$\begin{array}{rl} \alpha : & \mathbf{Interval} \to \mathbb{Z}_\perp^\top \\ \gamma : & \mathbb{Z}_\perp^\top \to \mathbf{Interval} \end{array}$$

   and prove that $(\mathbf{Interval}, \alpha, \gamma, \mathbb{Z}_\perp^\top)$ is a Galois connection.

2. Extend this Galois connection to mappings on program variables,

$$(\underbrace{\mathbf{Var} \to \mathbf{Interval}}_{\mathbf{State}}, \alpha', \gamma', \underbrace{\mathbf{Var} \to \mathbb{Z}_\perp^\top}_{\mathbf{State}^\sharp})$$

   and define α' and γ' in the standard way.

3. Define an abstract interpreter $[\![e]\!]^\sharp : \mathbf{State}^\sharp \to \mathbb{Z}_\perp^\top$ that determines whether the result of an expression *must* be constant at runtime.

4. Finally, show that the interpreter is correct by proving that $[\![e]\!]^\sharp \sqsupseteq \alpha \circ [\![e]\!] \circ \gamma'$, where $[\![e]\!]$ is the abstract interpretation in the lattice of intervals (defined in slide 198[1]).

[1] In that slide, the interpretation is lattice is denoted by $[\![e]\!]^\sharp$ instead of $[\![e]\!]$, but we shall use here the latter notation, since **Interval** is our concrete domain in this assignment.