Constant propagation analysis by abstract interpretation

Adrián Enríquez Ballester

February 22, 2022

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;
end
```

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.

We are going to define an abstract interpreter for knowing whether the result of an expression *must* be constant at runtime and prove its correctness in terms of an interval analysis that has been previously studied.

Definition 1 (The **Interval** lattice)**.** *The **Interval** lattice is defined as*

$$\textit{\textbf{Interval}} = \{\bot\} \cup \{[a,b] \mid a \in \mathbb{Z} \cup \{-\infty\},\ b \in \mathbb{Z} \cup \{+\infty\},\ a \le b\}$$

with the order relation

$$\begin{gathered} \bot \sqsubseteq int \quad \forall int \in \textit{\textbf{Interval}} \\ [a_1,b_1] \sqsubseteq [a_2,b_2] \iff a_2 \le a_1 \wedge b_1 \le b_2 \\ \forall a_1,a_2 \in \mathbb{Z} \cup \{-\infty\},\ \forall b_1,b_2 \in \mathbb{Z} \cup \{+\infty\} \end{gathered}$$

We already know that **Interval** is a lattice where $\bot$ represents the empty set, $\top$ is $[-\infty, +\infty]$ and, for every pair of intervals, its least upper bound is the smallest interval containing both and its greatest lower bound is their intersection.

It can be used to represent the possible values that a variable may take at each program point and, although we already know an abstract analysis which relates this lattice as an abstract domain with the collecting semantics as a concrete domain, in this case it will be used as our concrete domain.

Definition 2 (The $\mathbb{Z}_\bot^\top$ lattice)**.** *The set $\mathbb{Z}_\bot^\top = \mathbb{Z} \cup \{\top, \bot\}$ is a lattice with the order relation*

$$\begin{aligned} t &\sqsubseteq \top \quad \forall t \in \mathbb{Z}_\bot^\top \\ \bot &\sqsubseteq t \quad \forall t \in \mathbb{Z}_\bot^\top \\ t &\sqsubseteq t \quad \forall t \in \mathbb{Z}_\bot^\top \end{aligned}$$

where the respective least upper bound and greatest lower bound of every pair $t_1, t_2 \in \mathbb{Z}_\bot^\top$ are

$$t_1 \sqcup t_2 = \begin{cases} t_1 & \text{if } t_1 = t_2 \\ \top & \text{otherwise} \end{cases}$$

$$t_1 \sqcap t_2 = \begin{cases} t_1 & \text{if } t_1 = t_2 \\ \bot & \text{otherwise} \end{cases}$$

This will be our abstract domain, which represents the constant value that a variable contains if it is known and $\top$ otherwise. $\bot$ is included in order to construct a lattice, but we are not going to attach any special meaning to it.

Proposition 1 (A Galois connection between **Interval** and $\mathbb{Z}_\bot^\top$)**.** *The quadruple (**Interval**, $\alpha, \gamma, \mathbb{Z}_\bot^\top$) is a Galois connection with the abstraction function*

$$\begin{aligned} \alpha(\bot) &= \bot \\ \alpha([a, b]) &= \begin{cases} a & \text{if } a = b \\ \top & \text{otherwise} \end{cases} \quad \forall a \in \mathbb{Z} \cup \{-\infty\},\ \forall b \in \mathbb{Z} \cup \{+\infty\} \end{aligned}$$

and the concretization function

$$\begin{aligned} \gamma(\bot) &= \bot \\ \gamma(\top) &= \top \\ \gamma(k) &= [k, k] \quad \forall k \in \mathbb{Z} \end{aligned}$$

Proof. We have that **Interval** and $\mathbb{Z}_{\perp}^{\top}$ are lattices with the respective order relations stated in the Definitions 1 and 2. Let us check the required properties for α and γ (i.e. both functions are monotonically increasing, $\gamma \circ \alpha \sqsupseteq id$ and $\alpha \circ \gamma \sqsubseteq id$).

The monotonicity of α can be studied by case distinction on its domain:

- $\perp \sqsubseteq int \ \forall int \in$ **Interval** and, as $\alpha(\perp) = \perp$, then $\alpha(\perp) \sqsubseteq \alpha(int)$.
- If $[a, b] \sqsubseteq int$ for some $int \in$ **Interval**, $a \in \mathbb{Z} \cup \{-\infty\}$ and $b \in \mathbb{Z} \cup \{+\infty\}$ with $a \neq b$, then int must be $[c, d]$ for some $c \in \mathbb{Z} \cup \{-\infty\}$ and $d \in \mathbb{Z} \cup \{+\infty\}$ with $c \neq d$. As $\alpha([a, b]) = \top$ and $\alpha(int) = \alpha([c, d]) = \top$, it is satisfied that $\alpha([a, b]) \sqsubseteq \alpha(int)$.
- If $[a, a] \sqsubseteq int$ for some $int \in$ **Interval** and $a \in \mathbb{Z}$, then int by must be as one of the following cases:
  - $int = [a, a]$, so $\alpha([a, a]) = \alpha(int) = a$
  - $int = [b, c]$ for some $b \in \mathbb{Z} \cup \{-\infty\}$ and $c \in \mathbb{Z} \cup \{+\infty\}$ with $b \neq c$. As $\alpha([a, a]) = a$, $\alpha([b, c]) = \top$ and $a \sqsubseteq \top$, it follows that $\alpha([a, a]) \sqsubseteq \alpha(int) = \top$.

The monotonicity of γ can likewise be studied by case distinction:

- $\perp \sqsubseteq k \ \forall k \in \mathbb{Z}_{\perp}^{\top}$ and, as $\gamma(\perp) = \perp$, then $\gamma(\perp) \sqsubseteq \gamma(k)$.
- If $\top \sqsubseteq k$ for some $k \in \mathbb{Z}_{\perp}^{\top}$, then k must also be $\top$, so $\gamma(\top) = \gamma(k) = \top$.
- If $a \sqsubseteq k$ for some $a \in \mathbb{Z}$ and $k \in \mathbb{Z}_{\perp}^{\top}$, then k by must be as one of the following cases:
  - $k = a$, so $\gamma(a) = \gamma(k) = [a, a]$.
  - $k = \top$, so $\gamma(a) \sqsubseteq \gamma(k) = \top$.

With the same explicit technique, we can check that $\gamma \circ \alpha \sqsupseteq id$:

- $\alpha(\gamma(\perp)) = \alpha(\perp) = \perp$.
- $\alpha(\gamma([a, a])) = \alpha(a) = [a, a] \ \forall a \in \mathbb{Z}$.
- $\alpha(\gamma([b, c])) = \alpha(\top) = \top \ \forall a \in \mathbb{Z} \ \forall b \in \mathbb{Z} \cup \{-\infty\}$ and $\forall c \in \mathbb{Z} \cup \{+\infty\}$ with $b \neq c$. This satisfies the property because $id([b, c]) = [b, c] \sqsubseteq \top$.

And, finally, it remains to be checked that $\alpha \circ \gamma \sqsubseteq id$:

- $\gamma(\alpha(\perp)) = \gamma(\perp) = \perp$.
- $\gamma(\alpha(\top)) = \gamma(\top) = \top$.
- $\gamma(\alpha(a)) = \gamma([a, a]) = a \ \forall a \in \mathbb{Z}$.

□

Note that, as $\alpha \circ \gamma = id$, this defined Galois connection is also a Galois insertion.

These lattices and their connection would be useful if we were only interested in the analysis of a single program variable, so we are going to extend it for being able to keep track of multiple variables at once in the same analysis.

Definition 3 (The **State** and **State**$^{\#}$ lattices)**.** *We define the following lattices as the corresponding lattice extension to funcions of **Interval** and* $\mathbb{Z}_{\perp}^{\top}$*:*

$$\begin{aligned} \textbf{\textit{State}} &= \textbf{\textit{Var}} \rightarrow \textbf{\textit{Interval}} \\ \textbf{\textit{State}}^{\#} &= \textbf{\textit{Var}} \rightarrow \mathbb{Z}_{\perp}^{\top} \end{aligned}$$

*where **Var** is the set of variable names.*

As we know, this kind of extension of a lattice to functions yields also a lattice with the order relation $f_1 \sqsubseteq f_2 \iff f_1(s) \sqsubseteq f_2(s)\ \forall s$ in the former lattice.

Proposition 2 (A Galois connection between **State** and **State**$^{\#}$)**.** *The quadruple* $(\textbf{\textit{State}}, \alpha', \gamma', \textbf{\textit{State}}^{\#})$ *is a Galois connection with the abstraction function*

$$\alpha'(f) = \alpha \circ f \quad \forall f \in \textbf{\textit{State}}$$

and the concretization function

$$\gamma'(f) = \gamma \circ f \quad \forall f \in \textbf{\textit{State}}^{\#}$$

Proof. We have seen the Galois connection between **Interval** and $\mathbb{Z}_{\perp}^{\top}$ in the Proposition 1 and this is its extension to funcions with domain **Var**, which we know that is a systematic method for deriving a new Galois connection. □

Given this relation between the two lattices from the Definition 3, it can be derived an abstract interpreter for the abstract domain in terms of another interpreter for the concrete domain which can be used as a reference to guarantee the correctness of new interpreters.

Definition 4 (An abstract interpreter for expressions in the **State** lattice)**.** *Let* $[\![e]\!]\textbf{\textit{State}} \rightarrow \textbf{\textit{Interval}}$ *be the function defined as follows:*

$$
\begin{aligned}
[\![n]\!] &= \lambda\sigma.[n,n] \\
[\![x]\!] &= \lambda\sigma.\sigma(x) \\
[\![e_1 + e_2]\!] &= \lambda\sigma.\Big([\![e_1]\!]\sigma\Big) \oplus \Big([\![e_2]\!]\sigma\Big) \\
[\![e_1 - e_2]\!] &= \lambda\sigma.\Big([\![e_1]\!]\sigma\Big) \ominus \Big([\![e_2]\!]\sigma\Big) \\
[\![e_1 * e_2]\!] &= \lambda\sigma.\Big([\![e_1]\!]\sigma\Big) \otimes \Big([\![e_2]\!]\sigma\Big)
\end{aligned}
$$

*where $n \in \mathbb{Z}$, $x \in$ **Var**, $e_1, e_2 \in$ **Exp** and the definitions of $\oplus, \ominus$ and $\otimes$ are respectively*

$$
\begin{aligned}
\bot \oplus int &= \bot \quad \forall int \in \textbf{\textit{Interval}} \\
int \oplus \bot &= \bot \quad \forall int \in \textbf{\textit{Interval}} \\
[a_1, b_1] \oplus [a_2, b_2] &= [a_1 + a_2, b_1 + b_2] \\
&\qquad \forall a_1, a_2 \in \mathbb{Z} \cup \{-\infty\},\ \forall b_1, b_2 \in \mathbb{Z} \cup \{+\infty\}
\end{aligned}
$$

$$
\begin{aligned}
\bot \ominus int &= \bot \quad \forall int \in \textbf{\textit{Interval}} \\
int \ominus \bot &= \bot \quad \forall int \in \textbf{\textit{Interval}} \\
[a_1, b_1] \ominus [a_2, b_2] &= [a_1 - b_2, b_1 - a_2] \\
&\qquad \forall a_1, a_2 \in \mathbb{Z} \cup \{-\infty\},\ \forall b_1, b_2 \in \mathbb{Z} \cup \{+\infty\}
\end{aligned}
$$

$$
\begin{aligned}
\bot \otimes int &= \bot \quad \forall int \in \textbf{\textit{Interval}} \\
int \otimes \bot &= \bot \quad \forall int \in \textbf{\textit{Interval}} \\
[a_1, b_1] \otimes [a_2, b_2] &= [min(V), max(V)] \\
&\qquad \forall a_1, a_2 \in \mathbb{Z} \cup \{-\infty\},\ \forall b_1, b_2 \in \mathbb{Z} \cup \{+\infty\} \\
&\qquad \text{and } V = \{a_1 \cdot a_2, a_1 \cdot b_2, b_1 \cdot a_2, b_1 \cdot b_2\}
\end{aligned}
$$

$[\![e]\!]$ *is an abstract interpreter for expressions in the **State** lattice.*

We assume to be known that this interpreter is correct and, although $\alpha \circ [\![e]\!] \circ \gamma'$ is an interpreter in the **State**$^{\#}$ lattice, it is in general not computable. We are going to directly define a candidate for this latter.

Definition 5 (An abstract interpreter candidate for expressions in the **State**$^{\#}$ lattice)**.** *Let $[\![e]\!]^{\#}$**State**$^{\#} \rightarrow \mathbb{Z}_{\bot}^{\top}$ be the function defined compositionally as*

$$\llbracket n \rrbracket^{\#} = \lambda \sigma^{\#}.n$$
$$\llbracket x \rrbracket^{\#} = \lambda \sigma^{\#}.\sigma^{\#}(x)$$
$$\llbracket e_1 \ op \ e_2 \rrbracket^{\#} = \lambda \sigma^{\#}. \begin{cases} \top & \textit{if } \llbracket e_1 \rrbracket^{\#}\sigma^{\#} \in \{\top, \bot\} \\ & \quad \vee \llbracket e_2 \rrbracket^{\#}\sigma^{\#} \in \{\top, \bot\} \\ \llbracket e_1 \rrbracket^{\#}\sigma^{\#} \ op_{\mathbb{Z}} \ \llbracket e_2 \rrbracket^{\#}\sigma^{\#} & \textit{otherwise} \end{cases}$$

*where $n \in \mathbb{Z}$, $x \in$ **Var**, $e_1, e_2 \in$ **Exp**, $op \in \{+, -, *\}$ and $op_{\mathbb{Z}}$ denotes its corresponding counterpart operations in $\mathbb{Z}$.*

This function is an interpreter which determines whether the result of an expression must be constant at runtime given an abstract state.

Finally, we are going to prove the correctness of the defined interpreter in terms of the one from the Definition 4.

Proposition 3 ($\llbracket e \rrbracket^{\#}$ is a correct abstract interpreter)**.** *The abstract interpreter from the Definition 5 is a correct abstract interpreter, thus it can be used in the constant propagation analysis.*

Proof. Recall the abstract interpreter $\llbracket e \rrbracket$ in the lattice of intervals from the definition 4, whose correctness we assume to be known. Due to the Galois connection of the Proposition 2, $\alpha \circ \llbracket e \rrbracket \circ \gamma'$ is also a correct interpreter for the **State**$^{\#}$ lattice, so any other interpreter which always leads to less specific results than it (i.e. greater results) must also be considered to be correct.

We are going to show that $\llbracket e \rrbracket^{\#} \sqsupseteq \alpha \circ \llbracket e \rrbracket \circ \gamma'$ by structural induction on the expression e.

Given a literal expression $n \in \mathbb{Z}$, the defined abstract interpreter yields the same literal

$$\llbracket n \rrbracket^{\#}\sigma^{\#} = n$$

and the derived one produces the same result

$$(\alpha \circ \llbracket n \rrbracket \circ \gamma')\sigma^{\#} = \alpha(\llbracket n \rrbracket(\gamma'(\sigma^{\#}))) = \alpha(\llbracket n \rrbracket(\gamma \circ \sigma^{\#})) = \alpha([n, n]) = n$$

so $\llbracket n \rrbracket^{\#}\sigma^{\#} = (\alpha \circ \llbracket n \rrbracket \circ \gamma')\sigma^{\#}$

When given a variable $x \in$ **Var**, the defined abstract interpreter takes its value from the abstract state

$$\llbracket x \rrbracket^{\#}\sigma^{\#} = \sigma^{\#}(x)$$

and the derived interpreter does the same but going and returning to the concrete domain by means of the concretization and the abstraction functions

$$(\alpha \circ \llbracket x \rrbracket \circ \gamma')\sigma^{\#} = \alpha(\llbracket x \rrbracket(\gamma'(\sigma^{\#}))) = \alpha(\llbracket x \rrbracket(\gamma \circ \sigma^{\#})) = \alpha(\gamma(\sigma^{\#}(x)))$$

As this is a Galois connection, we know that $\alpha \circ \gamma \sqsubseteq id$, so $\alpha(\gamma(\sigma^{\#}(x))) \sqsubseteq \sigma^{\#}(x)$ and then $(\alpha \circ [\![x]\!] \circ \gamma')\sigma^{\#} \sqsubseteq [\![x]\!]^{\#}\sigma^{\#}$.

The base expression cases has been proved, so we proceed to the inductive steps. For an expression $e_1 + e_2$ with $e_1, e_2 \in \mathbf{Exp}$, the defined abstract interpreter yields:

$$[\![e_1 + e_2]\!]^{\#}\sigma^{\#} = \begin{cases} \top & \text{if } [\![e_1]\!]^{\#}\sigma^{\#} \in \{\top, \bot\} \\ & \quad \vee [\![e_2]\!]^{\#}\sigma^{\#} \in \{\top, \bot\} \\ [\![e_1]\!]^{\#}\sigma^{\#} + [\![e_2]\!]^{\#}\sigma^{\#} & \text{otherwise} \end{cases}$$

If $[\![e_1]\!]^{\#}\sigma^{\#} \in \{\top, \bot\}$ or $[\![e_2]\!]^{\#}\sigma^{\#} \in \{\top, \bot\}$, then the property trivially holds.

We are going to develop the derived abstract interpreter result in order to find more trivial situations:

$$\begin{aligned} (\alpha \circ [\![e_1 + e_2]\!] \circ \gamma')\sigma^{\#} &= \alpha([\![e_1 + e_2]\!](\gamma'(\sigma^{\#}))) \\ &= \alpha([\![e_1 + e_2]\!](\gamma \circ \sigma^{\#})) \\ &= \alpha\Bigg(\Big([\![e_1]\!](\gamma \circ \sigma^{\#})\Big) \oplus \Big([\![e_2]\!](\gamma \circ \sigma^{\#})\Big)\Bigg) \end{aligned}$$

If $\Big([\![e_1]\!](\gamma \circ \sigma^{\#})\Big) = \bot$ or $\Big([\![e_2]\!](\gamma \circ \sigma^{\#})\Big) = \bot$, then $\alpha(\bot) = \bot$ and the property also holds.

If it is not the case of any of the trivial ones, then $[\![e_1]\!]^{\#}\sigma^{\#} = a$ and $[\![e_1]\!]^{\#}\sigma^{\#} = b$ for some $a, b \in \mathbb{Z}$. The induction hypothesis states that

$$\begin{aligned} (\alpha \circ [\![e_1]\!] \circ \gamma')\sigma^{\#} &= \alpha([\![e_1]\!](\gamma \circ \sigma^{\#})) \sqsubseteq [\![e_1]\!]^{\#}\sigma^{\#} = a \\ (\alpha \circ [\![e_2]\!] \circ \gamma')\sigma^{\#} &= \alpha([\![e_2]\!](\gamma \circ \sigma^{\#})) \sqsubseteq [\![e_2]\!]^{\#}\sigma^{\#} = b \end{aligned}$$

As $\alpha([\![e_1]\!](\gamma \circ \sigma^{\#}))$ and $\alpha([\![e_2]\!](\gamma \circ \sigma^{\#}))$ cannot be $\bot$ nor $\top$, they must also be respectively a and b, which requires that $[\![e_1]\!](\gamma \circ \sigma^{\#}) = [a, a]$ and $[\![e_2]\!](\gamma \circ \sigma^{\#}) = [b, b]$. This means that

$$\begin{aligned} \alpha\Bigg(\Big([\![e_1]\!](\gamma \circ \sigma^{\#})\Big) \oplus \Big([\![e_2]\!](\gamma \circ \sigma^{\#})\Big)\Bigg) &= \alpha([a, a] \oplus [b, b]) \\ &= \alpha([a + b, a + b]) \\ &= a + b \end{aligned}$$

which is the same as $[\![e_1 + e_2]\!]^{\#}\sigma^{\#} = [\![e_1]\!]^{\#}\sigma^{\#} + [\![e_2]\!]^{\#}\sigma^{\#} = a + b$.

For an expression $e_1 - e_2$ with $e_1, e_2 \in \mathbf{Exp}$, the reasoning is as the above one until we have to develop the non trivial case, where we have:

$$\begin{aligned}
\alpha\left(\left(\llbracket e_1\rrbracket(\gamma\circ\sigma^{\#})\right)\ominus\left(\llbracket e_2\rrbracket(\gamma\circ\sigma^{\#})\right)\right) &= \alpha([a,a]\ominus[b,b])\\
&= \alpha([a-b,a-b])\\
&= a-b
\end{aligned}$$

which is the same as $\llbracket e_1 - e_2\rrbracket^{\#}\sigma^{\#} = \llbracket e_1\rrbracket^{\#}\sigma^{\#} - \llbracket e_2\rrbracket^{\#}\sigma^{\#} = a-b$.

Finally, for an expression $e_1 * e_2$ with $e_1, e_2 \in \mathbf{Exp}$, the reasoning is again as the composite expressions above until we have to develop the non trivial case

$$\begin{aligned}
\alpha\left(\left(\llbracket e_1\rrbracket(\gamma\circ\sigma^{\#})\right)\otimes\left(\llbracket e_2\rrbracket(\gamma\circ\sigma^{\#})\right)\right) &= \alpha([a,a]\otimes[b,b])\\
&= \alpha([min(\{a\cdot b\}), max(\{a\cdot b\})])\\
&= \alpha([a\cdot b, a\cdot b])\\
&= a\cdot b
\end{aligned}$$

which is the same as $\llbracket e_1 * e_2\rrbracket^{\#}\sigma^{\#} = \llbracket e_1\rrbracket^{\#}\sigma^{\#}\cdot\llbracket e_2\rrbracket^{\#}\sigma^{\#} = a\cdot b$. $\square$