Abstraction Domain with Congruence

Kihwan Kim (kimkihwan@kaist.ac.kr)

September 2021

0 Notations

* $\mathcal{P}(S)$ means power set of S
* 'iff' means 'if and only if'
* For two number n and m, $n \mid m$ means $n \equiv 0 \mod m$
* For two number n and m, $gcd(n, m)$ means greatest common divisor and $lcm(n, m)$ means least common multiple
* $n\%m = r$ iff $n \equiv r \mod m$ and $0 \leq r < m$
* $\exists!$ quantifier means uniquely exists.

1 Target Concrete State

This abstraction domain is targeting the program language which values are all integer number (we do not consider overflow or underflow). It doesn't have any kind of high-level structure such as functions or objects. Formally, set of all values $\mathbb{V}$ is equal to $\mathbb{Z}$, and let $\mathbb{S}$ be a set of all possible variable names.
We will consider only memory state. Memory state is a mapping from variable name to its value.

2 Definition of Abstraction Domain

The syntax of abstraction of value is pair of two natural number. Formally, abstraction domain of value $\mathbb{A}_v = \mathbb{Z} \times (\mathbb{N} \cup \{\infty\} \setminus \{0\})$
In textbook (Introduction to Static Analysis) represent singleton $\{n\}$ as $(n, 0)$ unlike my abstraction domain. It is because textbook if focused on concretization, on the other hand, we are focused on abstraction operators. Thinking about gcd or lcm with infinity is more intuitive than zero.
Second element of the pair means divisor and first element means remainder. So, second element cannot be zero, and first element (remainder) is always integer and less than second element (divisor).
We can think concretization function $\gamma : \mathbb{A}_v \to \mathbb{V}$. For ordinary cases,

$$\forall n \in \mathbb{Z}, \forall m \in (\mathbb{N} \setminus \{0\}), \gamma((n, m)) = \{mx + n \mid x \in \mathbb{Z}\}$$

One thing special is that second element can be infinite. Because dividing by infinity is an ambiguous behavior, we will define it formally below.

* For all $n \in \mathbb{Z}, \gamma((n, \infty)) = \{n\}$
* For all $n \in (\mathbb{Z} \cup \{\infty\} \setminus \{0\})$, $gcd(n, \infty) = n$ and $lcm(n, \infty) = \infty$
* For all $n \in \mathbb{Z}, (\exists! m \in \mathbb{Z}, n \equiv m \mod \infty)$ and $n \equiv n \mod \infty$

Now we need to construct poset (partially ordered set) to make our abstraction domain worth it. For interval, we use inclusion for partial ordering, so we will try it again to this abstraction domain.

Definition 1 (Galois Connection) *Galois connection of the abstraction domain is*

$$(\mathbb{C}, \subseteq) \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} (\mathbb{A}, \sqsubseteq)$$

where $\mathbb{C} = \mathcal{P}(\mathbb{S} \times \mathbb{V})$ *and* $\mathbb{A} = \mathbb{S} \mapsto \mathbb{A}_v$.

3 Definition of Functions

3.1 Typical Operators

For this subsection, we will fix two arbitrary abstract elements $a_1 = (r_1, d_1)$ and $a_2 = (r_2, d_2)$.

Definition 2 (Inclusion Test)

$$a_1 \sqsubseteq a_2 \text{ iff } d_1 \mid d_2 \wedge r_1 \equiv r_2 \mod d_2$$

Definition 3 (Abstract Union) *Let* $D = gcd(d_1, d_2)$*, then*

$$a_1 \sqcup^\sharp a_2 = (k, D) \text{ where } k = \arg \max_{-D<k<D} gcd(gcd((r_1 - k)\%D, (r_2 - k)\%D), D)$$

Definition 4 (Abstract Joint) *Let* $D = lcm(d_1, d_2)$*, then*

$$a_1 \sqcap^\sharp a_2 = \begin{cases} (k, D) & \text{if} \quad \exists x, y \in \mathbb{Z}, d_1 x + r_1 \equiv d_2 y + r_2 \equiv k \mod D \\ \bot & \text{otherwise} \end{cases}$$

We need abstract joint because it is used in implementation of 'filtering'. For scalability, while analyzing conditional command, filtering function joint proper mask abstraction with given abstraction.

Definition 5 (Widening)

$$a_1 \nabla a_2 = \begin{cases} a_1 & \text{if} \quad d_2 \mid d_1 \wedge (r_2 - r_1) \mid d_1 \\ \top & \text{otherwise} \end{cases}$$

Widening is a scheme that can analyze iteration even abstraction domain is not a finite lattice. First, set a finite logical constraints, and widening loosen the constraint or terminate, for each iteration. Logical constraint of abstraction domain with Congruence is only one, which is $n \equiv r \mod d$. So we just check that the states, after computing an iteration, still satisfies constraint $n \equiv r \mod d$.

3.2 Transfer Functions

First we will define abstraction of scalar expressions

Definition 6 (Abstraction of Scalar Constant) $[n]^\sharp(M^\sharp) = (n, \infty)$

Definition 7 (Abstraction of variable) $[x]^\sharp(M^\sharp) = M^\sharp(x)$

Definition 8 (Abstraction of Addition and Subtraction) *Let* r_1, r_2, d_1, d_2 *for given* E_1 *and* E_2 *such that* $[E_1]^\sharp(M^\sharp) = (r_1, d_1)$, $[E_2]^\sharp(M^\sharp) = (r_2, d_2)$, *and* $gcd(d_1, d_2) = D$,

$$[E_1 + E_2]^\sharp(M^\sharp) = ((r_1 + r_2)\%D, D)$$
$$[E_1 - E_2]^\sharp(M^\sharp) = ((r_1 - r_2)\%D, D)$$

Second, we will define abstraction of program commands

Definition 9 (Abstraction of Skip) $[skip]^\sharp(M^\sharp) = M^\sharp$

Definition 10 (Abstraction of Sequence) $[C_1; C_2]^\sharp(M^\sharp) = [C_2]^\sharp([C_1]^\sharp(M^\sharp))$

Definition 11 (Abstraction of Assignment and Input) *Recall the abstraction of scalar expressions.*

$$[x := E]^\sharp(M^\sharp) = M^\sharp[x \mapsto [E]^\sharp(M^\sharp)]$$
$$[\text{Input}(x)]^\sharp(M^\sharp) = M^\sharp[x \mapsto \top]$$

Definition 12 (Abstraction of Conditional Statements)

$$[\text{if}(B)\text{then}\{C_t\}\text{else}\{C_e\}]^\sharp(M^\sharp) = [C_t]^\sharp(filter_B(M^\sharp)) \sqcup^\sharp [C_e]^\sharp(filter_{\neg B}(M^\sharp))$$

Definition 13 (Abstraction of Assertion) $[\text{Assert } (x\%d = r)]^\sharp(M^\sharp) = filter_{x\%d=r}(M^\sharp)$

Definition 14 (Abstraction of Loop) *Recall the abstract iteration function* abs_iter *which use widening,*

$$[\text{While}(B)\{C\}]^\sharp(M^\sharp) = filter_{\neg B}(\text{abs\_iter}([C]^\sharp \circ filter_B, M^\sharp))$$

4 Proofs

W.I.P.