CS615: Formal Specification and Verification of Programs 2019

Lecture 14: Abstract domains - Box domain and octagonal domain

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2019-10-07

Abstract domain

An abstract domain consists of

- a lattice $(D, \sqsubseteq, \sqcup, \sqcap)$,
- a abstraction function $\alpha : C \to D$ and a concretization function $\gamma : D \to C$ such that
$$(D, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (C, \subseteq),$$
- a widening operator $\triangledown : D \times D \to D$, and
- a narrowing operator $\triangle : D \times D \to D$.

Implementation

In order to compute the fixed-points in abstract domain, we need to provide implementation on the following operators.

- $\sqsubseteq$
- $\sqcup$
- $\triangledown$
- $\triangle$ (some times not implemented!)
- $sp^{\#}$ over update statement
- $sp^{\#}$ over conditional statement

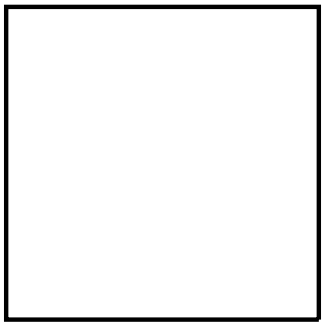
We need not do $\alpha \circ sp \circ \gamma$.

Topic 14.1

Box domain

Box domain

If the program has multiple variables, the product of interval domains for each variable is the box domain.



Example 14.1

Let $V = [x, y]$.

The abstract domain is $D = (\mathbb{Z} \times \mathbb{Z})^2$

We write $([2, 3]_x, [4, +\infty]_y) \in D$.

Operationtion in box domain

We have already seen implementation of the following operations over intervals.

- $\sqsubseteq$
- $\sqcup$
- $\triangledown$
- $\triangle$

We can pointwise extend the operations for the box domain.

$$([a_1, b_1], ..., [a_n, b_n]) \sqsubseteq ([c_1, d_1], ..., [c_n, d_n]) \quad \text{iff} \quad \wedge_{i=0}^{n} ([a_i, b_i] \sqsubseteq [c_i, d_i])$$

Exercise 14.1

Define operations $\sqcup$, $\triangledown$, and $\triangle$ operations in box domain

$sp^{\#}$ over assignments for Box domain

We may use interval arithmetic to compute $sp^{\#}$.

Example 14.2

$$sp^{\#}(x := y + x, ([2,3]_x, [1,4]_y))$$
$$= (([2,3] + [1,4])_x, [1,4]_y) = ([3,7]_x, [1,4]_y)$$

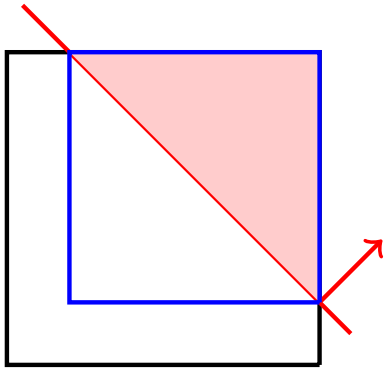
For each operation in the program, we need interval arithmetic.

Exercise 14.2

Define interval arithmetic for multiplication.

$sp^{\#}$ over conditions for Box domain

Consider $d' = sp^{\#}(d, \texttt{assume}(\texttt{aX} \leq \texttt{c}))$.



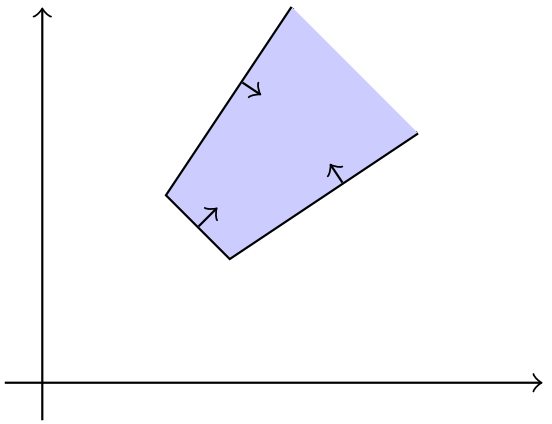
Exercise 14.3

Suggest an algorithm to compute the d'

Representing polyhendrons

- Constraints

$$2x - 3y \leq 0 \wedge -3x + 2y \leq 0 \wedge x + y \geq 25$$



- Vertices and rays

$$(Q, R) = (\underbrace{\{(15, 10), (10, 15)\}}_{\text{vertices}}, \underbrace{\{(3, 2), (2, 3)\}}_{\text{rays}})$$

Exercise 14.4

If you have vertices and rays, do you have an algorithm?

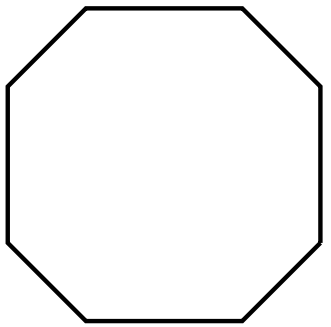
Topic 14.2

Octagon domain

Octagon domain

Let us assume $V = \{x_1, \ldots, x_n\}$.

$O = \{\pm \mathtt{x} \pm \mathtt{y} \leq c | \mathtt{y}, \mathtt{x} \in V, c \in \mathbb{I}\}$
where $\mathbb{I} \in \{\mathbb{Q}, \mathbb{Z}, \mathbb{R}\}$



$D = \{o_1 \wedge \cdots \wedge o_k | \forall i.\ o_i \in O\}$

Domain representation

- Since a tight octagonal difference bound matrix (ODBM) represents $F \in D$ canonically, we say D is a set of tight ODBMs .
- Tight ODBMs do not represent unsat formulas therefore we need to add a special element $\perp$ to represent the least element.

Switch lecture.
Let us learn some algorithms for solving octagonal constraints.

Lattice operators in Octagonal domain

Let A^1 and A^2 be $2n \times 2n$ tight ODBMs.

- $is\bot?$ due to canonical representation trivial
- $A^1 \sqsubseteq A^2 \triangleq A^1 \dot{\leq} A^2$
- $A^1 \sqcap A^2 \triangleq (\dot{\min}(A^1, A^2))^\bullet$
- $A^1 \sqcup A^2 \triangleq (\dot{\max}(A^1, A^2))^\bullet$
- $A^1 \triangledown A^2 = A^\bullet$, where $A_{ij} = (A^2_{ij} > A^1_{ij} ? \infty : A^1_{ij})$
- $sp^\#(\rho, A^1) = \alpha((\exists V' F[A^1] \wedge \rho)[V'/V])$

ρ is a polyhedron then the param to α is also polyhedron.

$\dot{\leq}$ is pointwise $\leq$

Octagonal abstraction of polyhedron

Let an polyhedron is given to us a pair of vertices Q and rays R.

$\alpha((Q,R)) = A$ is defined as follows

- $A_{(2i)(2i-1)} = (\ \exists r \in R.\ r_i > 0\ ?\ \infty :\ 2\max\{v_i | v \in Q\})$
- $A_{(2i-1)(2i)} = (\ \exists r \in R.\ r_i < 0\ ?\ \infty :\ -2\min\{v_i | v \in Q\})$
- $A_{(2i-1)(2j-1)} = A_{(2i)(2j)} = (\ \exists r \in R.\ r_i > r_j\ ?\ \infty :\ \max\{v_i - v_j | v \in Q\})$
- $A_{(2i-1)(2j)} = (\ \exists r \in R.\ r_i + r_j > 0\ ?\ \infty :\ -\min\{v_i + v_j | v \in Q\})$
- $A_{(2i)(2j-1)} = (\ \exists r \in R.\ 0 > r_i + r_j\ ?\ \infty :\ \max\{v_i + v_j | v \in Q\})$
- $A_{(i)(i)} = 0$

End of Lecture 14