Using Dynamic Analysis to Generate Disjunctive Invariants

ThanhVu Nguyen et al.

May 19, 2020

Introduction

- Invariants: defect detection, program verification and program repair.
- Find the invariants: static or dynamic, and their pros and cons.
- Conjunctive, polynomial and convex invariants $\Rightarrow$ Disjunctive program properties

Disjunctive Program

Example

$$\texttt{if } (p)\{a = 1;\} \texttt{ else } \{a = 2;\}$$

Neither $a = 1$ nor $a = 2$ is an invariant, but $(p \wedge a = 1) \vee (\neg p \wedge a = 2)$ is an invariant.

Overview

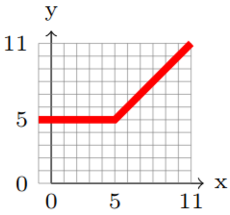
- Existing invariant inference algorithm.
- Max-Plus invariants and a way to infer them.
- Using k-induction to verifying candidate invariants.
- Experiment results.

Motivating Example

Example (1)

```
void ex1(int x){
  int y=5;
  if (x>y) x=y;
  while[L](x ≤ 10){
    if(x ≥ 5)
      y=y+1;
    x=x+1;
  }
  assert(y==11);
}
```

x	y
-1	5
$\vdots$	$\vdots$
5	5
6	6
$\vdots$	$\vdots$
11	11



A program with branch and a executing trace starting from $x = -1$. Invariant at location L is

$$(x < 5 \wedge y = 5) \vee (5 \leq x \leq 11 \wedge x = y)$$

Conjunctive Invariant

From the given trace, existing tools loke Daikon and DIG can generate conjunctive invariant below:

$$11 \geq x$$

$$11 \geq y \geq 5$$

$$y \geq x$$

which is over-approximating and cannot express the disjunctive behavior.

Algorithm Overview

Max-plus inequality: e.g. $\max(x, y+1) \geq 1$
By constructing a max-plus polyhedra over the trace points, we obtain relations simplified to

$$\begin{array}{ccccc} 11 & \geq & x & \geq & -1 \\ 11 & \geq & y & \geq & 5 \\ 0 & \geq & x-y & \geq & -6 \end{array}$$

$$(x < 5 \wedge 5 \geq y) \ \vee \ (x \geq 5 \wedge x \geq y)$$

Then use *k*-**induction** to remove the spurious relations $x - y \geq -6, x \geq -1$. Further, the prover proves that $11 \geq x$ is redundant.

$$11 \geq y \geq 5$$

$$0 \geq x - y$$

$$(x < 5 \wedge 5 \geq y) \vee (x \geq 5 \wedge x \geq y)$$

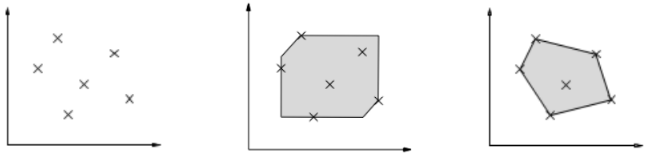
Invariant Inference Algorithm

DIG: A Dynamic Invariant Generator for Polynomial and Array Invariants
General polyhedra inequality:

$$c_1 t_1 + \ldots + c_n t_n \geq 0$$

Octagonal Inequalities:

$$c_1 t_1 + c_2 t_2 \geq k$$



Max-Plus Invariant

To model disjunctive invariant, we use formulas representing max-plus polyhedra. i.e. a non-convex hull which is convex in the sense of max-plus algebra. Formally, max-plus relation can be expressed as:

$$\max(c_0, c_1 + v_1, \ldots, c_n + v_n) \geq \max(d_0, d_1 + v_1, \ldots, d_n + v_n)$$

$c_i, d_i \in \mathbb{R} \cup \{-\infty\}$
It is obvious that $\max(x, y) > n$ is equivalent to $(x \geq y \wedge x > n) \vee (y > x \wedge y > n)$. Hence,...

Geometric Shape of Max-Plus Invariant

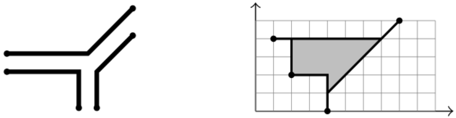


Figure 3: (a) Three possible shapes of a max-plus line segment: $\max(x+a,b) \geq y$ **(top),** $\max(y+a,b) \geq x$ **(right),** $\max(x+a,y+b) \geq 0$ **(left) and (b) a max-plus convex hull built over four points using these line segments.**

Convex on max-plus algebra: for any two points of the max-plus polyhedra, there is a max-plus line segment connecting these two points.

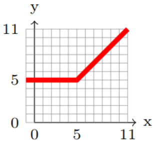
Dynamically Infer Max-Plus Invariants

Input: set of variables V, set of traces X, max degree d
Output: A set S of polynomial inequalities.
$T \leftarrow \texttt{genTerms}(V, d)$
$P \leftarrow \texttt{genPoints}(T, X)$
$H \leftarrow \texttt{createMaxPlusPolyhedron}(P)$
$S \leftarrow \texttt{extractFacets}(H)$
return S.

Inferring Example

```
void ex1(int x){
  int y=5;
  if (x>y) x=y;
  while[L](x ≤ 10){
    if(x ≥ 5)
      y=y+1;
    x=x+1;
  }
  assert(y==11);
}
```

x	y
-1	5
$\vdots$	$\vdots$
5	5
6	6
$\vdots$	$\vdots$
11	11



$$
\begin{array}{cccll}
11 & \geq & x & \geq & -1 \\
11 & \geq & y & \geq & 5 \\
0 & \geq & x-y & \geq & -6 \\
\max(0, x-5) & \geq & y-5 & &
\end{array}
$$

Conjunctions of formulas above is equivalent to

$$(x < 5 \wedge 5 = y) \vee (5 \leq x \leq 11 \wedge x = y)$$

Weak Max-Plus Invariants

Traditional min max invariant complexity: $O(n^k)$[1].
We define a weak max relation to be of the form:

$$\max(c_0, c_1 + v_1, \ldots, c_k + v_k) \geq v_j + d$$

$$v_j + d \geq \max(c_0, c_1 + v_1, \ldots, c_k + v_k)$$

where $c_i \in \{0, -\infty\}$
Difference of weak version: Only performs in the form $\max(x, b) \geq y$ and $\max(y, b) \geq x$

[1] Inferring Min and Max Invariants Using Max-plus Polyhedra

Example of Finding Weak Max-Plus Invariant

Points: $\{(x-1, y_1), \ldots, (x_n, y_n)\}$ in 2D.
For the form $\max(c_0, x + c_1, y + c_2) \geq x + d$, there are 8 versions depending on the value of c_i. Same for the other direction and different variable.
Then compute parameter d by using the points. e.g.

$$\max(y, 0) \geq x + d$$

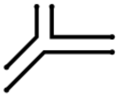
then, $d = \min(\max(y_i, 0) - x_i)$
Number of inequalities: $O(k2^{k+2})$
Time complexity: $O(n2^k)$

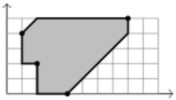
Min-Plus Invariant

Min relation is of the form

$$\min(c_0, c_1 + v_1, \ldots, c_n + v_n) \geq \min(d_0, d_1 + v_1, \ldots, d_n + v_n)$$

Shapes:





Min-plus:

$$\min(c_0, c_1 + v_1, \ldots, c_k + v_k) \geq v_j + d$$

$$v_j + d \geq \min(c_0, c_1 + v_1, \ldots, c_k + v_k)$$

where $c_i \in \{0, +\infty\}$

Combine Max-Plus and Min-Plus

```
int ex2(int x){
  int y, b;
  if (x>=0) {y=x+1;}
  else {y=x-1;}
  b=(y>10);
  [L]
  return b;
}
```

x	y	b
-50	-51	0
-33	-34	0
9	10	0
10	11	1
12	13	1
40	41	1

The invariant of this program at location L is

$$y \leq 10 \Leftrightarrow b = 0$$

can be described equivalently by

$$\max(y - 10, 0) \geq b \text{ and } b + 10 \geq \min(y, 11)$$

Verifying Candidate Invariant

In the algorithm we use k-induction to verify the invriant and remove spurious invariant.
k-Induction: k base cases are specified, and k previus instance are availble to prove the inductive step.
$M = (I, T)$

$$\begin{aligned} I \wedge T_1 \wedge \cdots \wedge T_k &\Rightarrow p_0 \wedge \cdots \wedge p_k \\ p_n \wedge T_{n+1} \wedge \cdots \wedge p_{n+k} \wedge T_{n+k+1} &\Rightarrow p_{n+k+1} \end{aligned}$$

Sound, not complete.

k-Induction Example

Example (2)

$M = (I, T)$
$I : x_0 = 0 \wedge y_0 = 1 \wedge z_0 = 2.$
$T_n : x_n = y_{n-1} \wedge y_n = z_{n-1} \wedge z_n = x_{n-1}$
Standard induction: $I \Rightarrow p_0$, $p_i \wedge T_{i+1} \not\Rightarrow p_{i+1}$.
3-Inductive: $I \wedge T_1 \wedge T_2 \wedge T_3 \Rightarrow p_0 \wedge p_1 \wedge p_2 \wedge p_3$.
$p_i \wedge T_{i+1} \wedge p_{i+1} \wedge T_{i+2} \wedge p_{i+2} \wedge T_{i+3} \Rightarrow p_{i+3}$

Code: k-Induction

input : I, T, p
output: $\{proved, disproved, unproved\}$

```
for k = 0 to maxK do
    // base case
    if k = 0 then S_b.assert(I) else S_b.assert(T_k)
    if ¬S_b.entail(p_k) then return (disproved, S_b.cex)

    // induction step
    S_s.assert(p_k, T_{k+1})
    if ¬S_s.entail(p_{k+1}) then return proved

return unproved
```

Code: KIP

input : S, L, P
output: P_i, P_r, P_d, P_u

```
I, T ← vcgen(S, L)
P_p ← ∅; P_d ← ∅; P_u ← ∅
repeat
    New_p ← ∅; New_u ← ∅
    foreach p ∈ P do
        r ← kprove(I, T, p)
        if r = proved then
            P_p.add(p); New_p.add(p)
        else if r = unproved then New_u.add(p)
        else P_d.add(p)
    KIP.addLemmas(New_p)
    P ← New_u
until New_p = ∅ ∨ New_u = ∅
P_u ← P
P_i, P_r = check_redundancy(P_p)
return P_i, P_r, P_d, P_u
```

Experiment Result

Prog	Loc	Var	Gen	T_{Gen}	Val	T_{Val}	Hoare
ex1	1	2	15	0.2	4	1.5	✓
strncpy	1	3	69	1.1	4	7.7	✓
oddeven3	1	6	286	3.7	8	16.0	✓
oddeven4	1	8	867	12.7	22	46.0	✓
oddeven5	1	10	2334	56.8	52	1319.4	✓
bubble3	1	6	249	4.1	8	4.9	✓
bubble4	1	8	832	11.7	22	47.6	✓
bubble5	1	10	2198	53.9	52	938.2	✓
partd3	4	5	479	10.5	10	50.8	✓
partd4	5	6	1217	23.3	15	181.1	✓
partd5	6	7	2943	53.3	21	418.1	✓
parti3	4	5	464	10.3	10	45.5	✓
parti4	5	6	1148	22.4	15	165.1	✓
parti5	6	7	2954	53.6	21	405.6	✓
total			16055	317.6	264	3647.5	14/14