A Concoction of Zonotope Abstraction and Constraint Programming for finding an Invariant

Bibek Kabi – Éric Goubault – Sylvie Putot

Cosynus team, LIX, Ecole Polytechnique

SWIM 2016
9th Small Workshop on Interval Methods
École Normale Supérieure de Lyon in Lyon, France
http://swim2016.sciencesconf.org/

A Quick Introduction to Invariants

Definition

A logical assertion at a location over the program variables that remain same whenever the location is reached

What are they used for?

Synthesizing an invariant is a key concept in formal verification to ensure correctness of programs.

e.g. the program variables stay within some bounds

How to find Invariant?

By inferring a stronger form of the invariant i.e. inductive invariant

Example

Octagon example [Miné *et al.* (2015)]

$x := \text{input } [-1, 1]$
$y := \text{input } [-1, 1]$
while true do
$\quad x' := \frac{\sqrt{2}}{2} * (x - y)$
$\quad y' := \frac{\sqrt{2}}{2} * (x + y)$
$\quad x := x' \quad y := y'$
done

$$E := [-1, 1] \times [-1, 1]$$

(entry states)

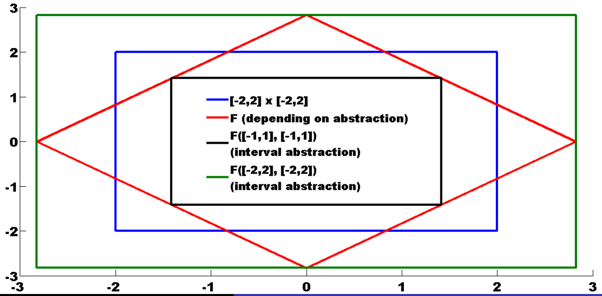
$$F(Y) := \{(\frac{\sqrt{2}}{2}(x - y), \frac{\sqrt{2}}{2}(x + y)) | (x, y) \in Y\}$$

(loop body)

Difference between invariant and inductive invariant using previous example

Invariant versus inductive invariant

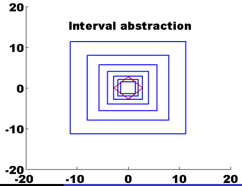
- Let us take a box $I := [-2, 2] \times [-2, 2]$
- All program states i.e. (x, y) lie inside I every time the execution reaches the loop head $\therefore I$ is invariant.
- I is not an inductive invariant because $F(I) \nsubseteq I$ (also in next slide)
- Thus, invariants are not always inductive invariants



Method to Infer Inductive Invariant

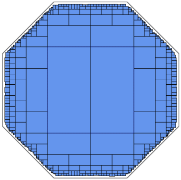
Abstract Interpretation

- $\mathrm{lfp}\, F = \inf\{F(G) \subseteq G\}$ [Tarski (1955)] (smallest inductive invariant)
- The well-known and dominant approach for finding inductive invariant for program verification is *abstract interpretation* [Cousot *et al.* (1977)]
- A classical way to find $\mathrm{lfp}\, F$ is to use Kleene iterations (F is continous and semantic domain is complete partial order)
  $X^0 = E, X^1 = F(X^0), \ldots, X^{k+1} = X^k \cup F(X^k)$
- Kleene iterations on the example do converge but to a useless one $[-\infty, \infty] \times [-\infty, \infty]$ i.e. $(\top)$



Inferences Drawn

- $I := [-2,2] \times [-2,2]$ is not an inductive invariant $\because F(I) \not\subseteq I$
- In fact, no box is inductive
- If we have a union of boxes G with the spiky edges removed then the set is inductive (shown in figure [Miné COVERIF meeting])



- We can apply *constraint solving* with splitting
- Also, abstract domain to be chosen is very crucial for F

Miné *et al.* (2015)

Algorithm: in Brief

- Replacing each operation in the program with associated operation of the chosen abstract domain
- Forming set of constraints and solving them using constraint solver based on propagation and splitting
- The algorithm starts with a single box S (here $S=[-2,2] \times [-2,2]$) and iteratively split [Pelleau *et al.* (2013)] or discard until the inductive invariant properties are satisfied or may stop if the size of the boxes is small enough

Properties to infer inductive invariant

- $E \subseteq \cup_i S_i$ (set of boxes contains the entry)
- $S_i \subseteq I$ (set of boxes entails the invariant)

  this particular property makes possible for the *constraint solving* to be applied
- $F^\sharp(S_i) \subseteq \cup_i S_i$ (set of boxes is inductive)

Box Classification

Based on the properties discussed earlier, boxes are classified into various categories

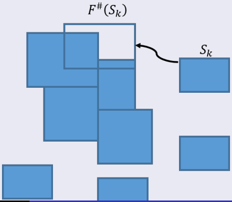
Box classes

- *doomed*, if $F^\sharp(S_k) \cap (\cup_i S_i) = \emptyset$

  Even splitting cannot help it; are only discarded

- *benign*, if $F^\sharp(S_k) \subseteq \cup_i S_i$

  Measure of Benign:coverage(S_k):= $\frac{\sum_i vol(F^\sharp(S_k) \cap S_i)}{vol(F^\sharp(S_k))}$ (The ultimate aim is to have $\forall k$: coverage(S_k)= 1 which implies the algorithm returns when the box has a coverage of 1)



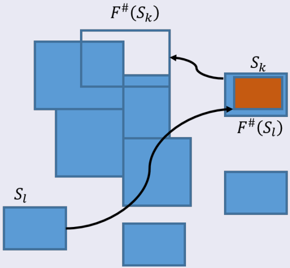
Box Classification

necessary, if $S_k \cap E \neq \emptyset$

- if *necessary*, we split
- else check for usefulness (next slide) or size or coverage and decide to discard or split until ϵ_s (here it is $0.01 \times size([-2, 2] \times [-2, 2])$

Box Classification

- *useful*, if $S_k \cap (\cup_i F^\sharp(S_i)) \neq \emptyset$
- In the figure, the box S_k intersects the image of a box S_l under $F^\sharp$ which implies that $F^\sharp(S_l) \subseteq \cup_i S_i$ i.e. S_k helps make S_l benign
- if S_k is discarded it leaves S_l non-benign i.e. $F^\sharp(S_l) \nsubseteq \cup_{i \neq k} S_i$ and eventually failure



Motivation for using zonotopic abstraction

- To have a more precise $F^\sharp$ and also tractable
- To have less splitting
- To sum up, we are trying to see how much refinement is possible with such an abstraction when combined with *constraint solving*

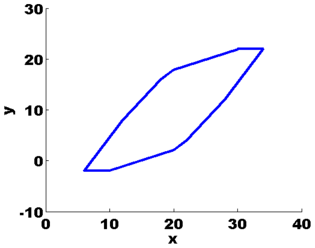
A Brief Introduction on Zonotopes

An affine form of a variable x

$$\hat{x} := {\alpha_0}^x + \sum_{i=1}^{n} {\alpha_i}^x \varepsilon_i$$

Zonotope is the geometric concretization of sets of values taken by the affine form

$$\hat{x} = 20 - 3\varepsilon_1 + 5\varepsilon_2 + 2\varepsilon_3 + 1\varepsilon_4 + 3\varepsilon_5$$
$$\hat{y} = 10 - 4\varepsilon_1 + 2\varepsilon_2 + 1\varepsilon_4 + 5\varepsilon_5$$



A Brief Introduction on Zonotopes

Zonotope is represented by a center $c \in \mathbb{R}^n$ and generators $g^{(i)} \in \mathbb{R}^n$

$$Z = \{x \in \mathbb{R}^n | x = c + \sum_{i=1}^{p} \varepsilon_i g^i\}$$

$$\hat{x} = 20 - 3\varepsilon_1 + 5\varepsilon_2 + 2\varepsilon_3 + 1\varepsilon_4 + 3\varepsilon_5$$

$$\hat{y} = 10 - 4\varepsilon_1 + 2\varepsilon_2 + 1\varepsilon_4 + 5\varepsilon_5$$

$$c = \begin{bmatrix} 20 \\ 10 \end{bmatrix}$$

and

$$g^{(i)} = \begin{bmatrix} -3 & 5 & 2 & 1 & 3 \\ 4 & 2 & 0 & 1 & 5 \end{bmatrix}$$

Challenges while combining zonotope *abstraction* and *contraint solving*

Challenges

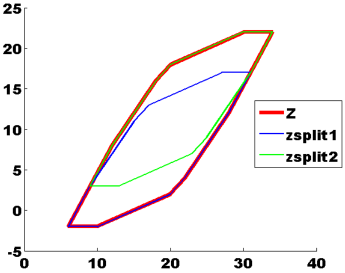
- Calculating coverage(S_k) needs volume computation
  - Volume of zonotopes can be computed by the method in [Gover *et al.* (2010)]
  - However, this is costly
  - We change this into checking *coverage* $= 1$ (just an inclusion of zonotopes) and otherwise the coverage is a heuristic, we can think of estimating the size of the intersection by computing the size of the resulting ε_i in intervals after using the CAV 2010 intersection
- The size computation

  one way to characterize size would be to compute sum (or weighted sum) of the norm of the columns of generator matrix [Combastel *et al.* (2015)] and [Le *et al.* (2013)]

Challenges while combining zonotope *abstraction* and *contraint solving*

Splitting with overlap

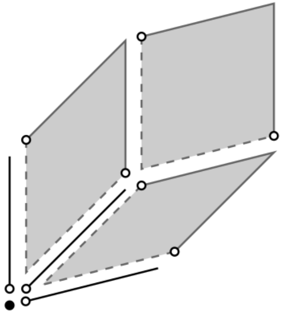
- A zonotope can be split into two zonotopes with overlap by splitting the j^{th} generator [Althoff *et al.* (2008)]
- We split the generator with longest length ($\|g^{(i)}\|_1$) to ensure less overlap



Challenges while combining zonotope *abstraction* and *contraint solving*

Splitting without overlap

A zonotope can be split into disjoint union of parallelotopes

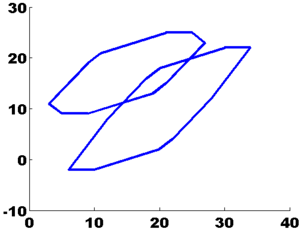


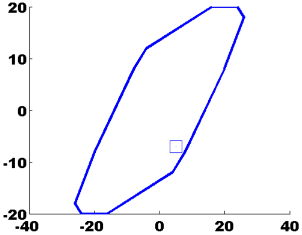
Challenges while combining zonotope *abstraction* and *contraint solving*

Checking for intersection

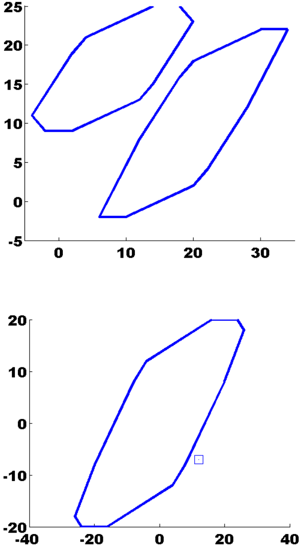
- Let $Z_1 = (c1, < g_1, \cdots, g_k >)$ and $Z_2 = (c2, < h_1, \cdots, h_m >)$
- $Z_1 \cap Z_2 \neq \emptyset$ if $c1 - c2$ is entailed in $(0, < g_1, \cdots, g_k, h_1, \cdots, h_m >)$
- We use LP solver for checking point inclusion inside zonotope

Intersecting case





Non-intersecting case



Future Scope

- Use CORA Toolbox [Althoff (2015)] to develop the complete algorithm
- Using of efficient data structures (hierarchical representation of sets of zonotopes) inorder to make it faster

References I

Cousot, P., & Cousot, R. (1977, January). Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (pp. 238-252). ACM.

Tarski, A. (1955). A lattice-theoretical fixpoint theorem and its applications. Pacific journal of Mathematics, 5(2), 285-309.

Miné, A., Breck, J., & Reps, T. (2015). An algorithm inspired by constraint solvers to infer inductive invariants in numeric programs. Submitted for publication.

Pelleau, M., Miné, A., Truchet, C., & Benhamou, F. (2013, January). A constraint solver based on abstract domains. In Verification, Model Checking, and Abstract Interpretation (pp. 434-454). Springer Berlin Heidelberg.

Gover, E., & Krikorian, N. (2010). Determinants and the volumes of parallelotopes and zonotopes. Linear Algebra and Its Applications, 433(1), 28-40.

Althoff, M. (2015). An introduction to CORA 2015. In Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems.

Combastel, C. (2015). Zonotopes and Kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence. Automatica, 55, 265-273.

References II

Althoff, M., Stursberg, O., & Buss, M. (2008, December). Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In Decision and Control, 2008. CDC 2008. 47th IEEE Conference on (pp. 4042-4048). IEEE.

Guibas, L. J., Nguyen, A., & Zhang, L. (2003, January). Zonotopes as bounding volumes. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (pp. 803-812). Society for Industrial and Applied Mathematics.

Le, V. T. H., Stoica, C., Alamo, T., Camacho, E. F., & Dumur, D. (2013). Zonotopes: From Guaranteed State-estimation to Control. John Wiley & Sons.

Goubault, E., & Putot, S. (2015). A zonotopic framework for functional abstractions. Formal Methods in System Design, 47(3), 302-360.

Ghorbal, K., Goubault, E., & Putot, S. (2010, July). A logical product approach to zonotope intersection. In International Conference on Computer Aided Verification (pp. 212-226). Springer Berlin Heidelberg.