A Data Driven Approach for Algebraic Loop Invariants

Paper by Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken, Percy Liang, and Aditya V. Nori
In ESOP 2013

Vanya Dancheva

Seminar: Research Topics in Software Engineering

22.04.2013

Motivation

- Generating loop invariants is crucial for program verification
- Major drawbacks with previous techniques for algebraic invariants
  - Restrict predicates on branches to either equalities or inequalities
  - Cannot handle nested loops
  - Interpret program variables as real numbers

Guess-and-check algorithm

- Finds algebraic invariant of the form

$$\wedge_i \; f_i(x_1, \ldots, x_n) = 0$$

- Guess phase – suggests a candidate invariant
- Check phase – checks whether the candidate invariant is an invariant

- Advantages
  - Uses a decision procedure to check the candidate invariant
  - The guess phase operates over data

Example

```
1: assume(x=0 && y=0);
2: while(*) do
3:     writelog(x, y);
4:     y := y + 1;
5:     x := x + y;
6: done
```

- First, run the program and accumulate the resulting data
- Assume the loop is exercised once
- Assume an upper bound on the degree of the polynomials –

  $d = 2$

Example

- Enumerate all monomials up to the chosen degree $\vec{\alpha} = \{1, x, y, y^2, x^2, xy\}$

- Construct a data matrix A

1	x	y	y^2	x^2	xy
1	0	0	0	0	0

Example

- Employ the null space of A to compute a candidate invariant

$$I \equiv \bigwedge_{i=1}^{k} (b_i^T \begin{bmatrix} 1 \\ x \\ y \\ y^2 \\ x^2 \\ xy \end{bmatrix} = 0)$$

$\{b_1, b_2, \ldots, b_k\}$ is a basis for the null space of A

Example

- The basis for the null space of A is

$$\left\{\begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\1\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}\right\}$$

- Candidate invariant is

$$I \equiv x = 0 \wedge y = 0 \wedge x^2 = 0 \wedge y^2 = 0 \wedge xy = 0$$

Example

- Check the candidate invariant I

  $L \equiv \texttt{while } B \texttt{ do } S$

1. If φ is a precondition then $\varphi \Rightarrow I$
2. Executing the loop body S with a state satisfying $I \wedge B$, results in a state satisfying I

Example

1. $(x = 0 \wedge y = 0) \Rightarrow (x = 0 \wedge y = 0 \wedge x^2 = 0 \wedge y^2 = 0 \wedge xy = 0)$

2. $(x = 0 \wedge y = 0 \wedge x^2 = 0 \wedge y^2 = 0 \wedge xy = 0 \wedge y' = y + 1 \wedge x' = x + y')$
   $\Rightarrow (x' = 0 \wedge y' = 0 \wedge x'^2 = 0 \wedge y'^2 = 0 \wedge x'y' = 0)$

A counter example for 2. $x' = 1, y' = 1$

Example

- Lets generate more program states

A=

1	x	y	y^2	x^2	xy
1	0	0	0	0	0
1	1	1	1	1	1
1	3	2	4	9	36
1	6	3	9	36	18
1	10	4	16	100	40

Example

- Basis for the null space of A is $\left\{\begin{bmatrix} 0 \\ 2 \\ -1 \\ -1 \\ 0 \\ 0 \end{bmatrix}\right\}$
- New candidate invariant is $I \equiv 2x - y - y^2 = 0$
- Both conditions 1. and 2. are valid and this is the desired loop invariant

1. $(x = 0 \wedge y = 0) \Rightarrow y + y^2 = 2x$
2. $(y + y^2 = 2x \wedge y' = y + 1 \wedge x' = x + y') \Rightarrow (y' + y'^2 = 2x')$

The algorithm

Guess-And-Check(L,ϕ,d)
Returns: A loop invariant I for L

```
1:  x := vars (L)
2:  Tests := TestGen(φ,L)
3:  logfile := {}
4:  for t in Tests do
5:      logfile := logfile :: Execute(L, x = t)
6:  end for
7:  repeat
8:      I := Guess(logfile, d)
9:      (done, t) := Check(I, L, φ)
10:     if ¬done then
11:         logfile := logfile :: t
12:     end if
13: until done
14: return I
```

Guess(logfile, d)
Returns: A candidate invariant

```
1:  if logfile = {} then
2:      return false
3:  end if
4:  A := DataMatrix(logfile, d)
5:  B := Basis(NullSpace(A))
6:  if B = 0 then
7:      // No non-trivial invariant
8:      return true
9:  end if
10: return CandidateInvariant(B)
```

The algorithm

- The Guess-and-check algorithm terminates after at most n iterations, if the Check procedure is sound and complete
  - n is the total number of monomials with degree bounded by d
- If the algorithm Guess-and-check terminates and the Check procedure is sound it returns an invariant

Extensions and Evaluation

- Guess-and-check easily extends to nested loops
- Linear invariants

- Evaluated on benchmarks from the literature
- Terminated on all benchmarks in one iteration

References

- de Moura, L.M., Bjorner. “Z3: An efficient SMT solver”. In TACAS. pp. 337-340 (2008)
- Sharma, R., Gupta, S., Hariharan, B., Aiken, A., Nori. “A data driven approach for algebraic loop invariants”. Tech. Report MSR-TR-2012-97, Microsoft Research (2012)