VIAP 1.1

Pritom Rajkhowa and Fangzhen Lin

Presenter: Pritom Rajkhowa

Hong Kong University of Science and Technology

April 5, 2019

Aim and Motivation

- VIAP (Verifier for Integer Assignment Programs) is an automated system for verifying safety properties of procedural programs with integer assignments and loops.
- It is based on a translation from of a program to a set of first-order axioms with quantification over natural numbers, and currently makes use of SymPy as the algebraic simplifier and the SMT solver Z3 as the theorem prover.
- Here we describes VIAP 1.1, a new version that makes use of our newly developed recurrence solver. As a result, VIAP 1.1. is able to verify many programs that were out of reach for the older version VIAP 1.0. An earlier version of VIAP competed at SV-COMP 2018, and is described in [RL17, RL18].

Work Flow of VIAP

Input Program → Translation → Recurrence Solver → Z3 → Output

Input Program

```
int x = 0, y = 0;
while (x < 100){
        if (x < 50) {
                y++;
        } else {
                y--;
        }
x++;
}
assert(y == 0);
```

Output of Translator

$$x_1 = x_2(N), y_1 = y_2(N),$$
$$\forall n. x_2(n+1) = x_2(n)+1,$$
$$x_2(0) = 0,$$
$$\forall n. y_2(n+1) = ite(x_2(n) < 50,$$
$$y_2(n)+1, y_2(n)-1),$$
$$y_2(0) = 0,$$
$$\neg(x_2(N) < 100),$$
$$\forall n. n < N \rightarrow x_2(n) < 100.$$

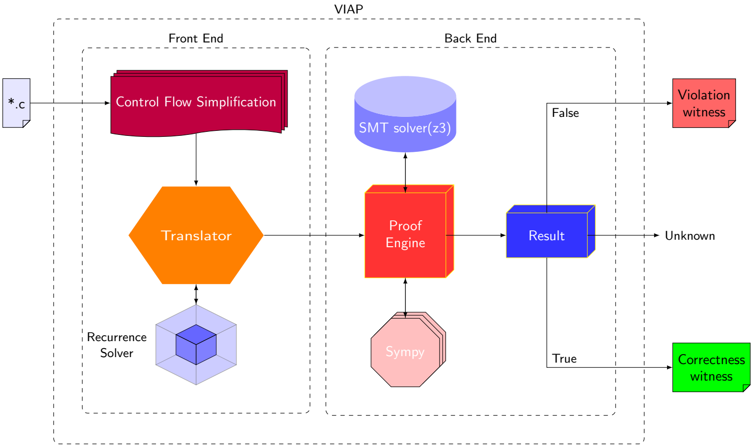
Output of Recurrence Solver

$$x_1 = N,$$
$$y_1 = ite(0 \leq N < 50, N, 100-N),$$
$$N \geq 100),$$
$$\forall n. n < N \rightarrow n < 100.$$

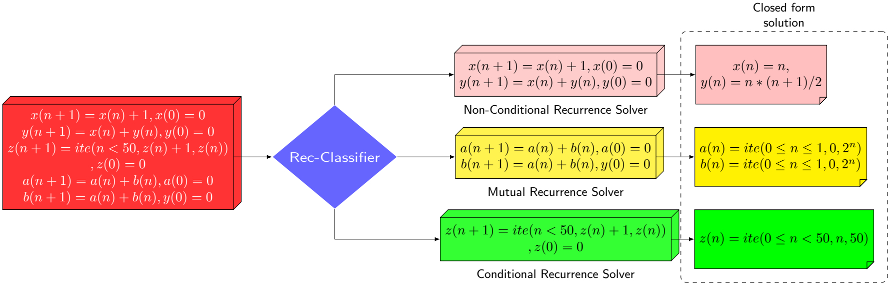
Input to Z3

```
s.add(x1 === N)
s.add(y1 ==
If(And(0≤ N, N < 50)
,N,100-N))
s.add(N ≥ 100)
s.add(Forall([n],Implies(And(n≥
0, n < N), n < 100)
```

VIAP Architecture



Recurrence Solver (RS)



Divergent from other Approaches

- Existing work based on recurrences analysis either focuses on computing accurate information about syntactically restricted loops, or focuses on over-approximate analysis of general loops. In contrast, our recurrences precisely summarize the semantics of general loops. In other words, we aim to analyze the accurate semantics of general loops.
- A major disadvantage of existing methods is that if they fail to find a closed form solution, then they are unable to find suitable invariants as a result, they fail to complete the proof. Whereas even if our system is unable to solve recurrence equations, our system can still complete the proof.
- Some existing methods try to use recurrence solver for inferring all invariants they can find. However our approach is property-guided (or goal-directed) and we only aim to verify the condition.

SV-COMP 2019 Result

Dataset	Total Task	correct		correct-unconfimed		incorrect		Score
		True	False	True	False	True	False	
Array	231	175	28	0	3	0	0	378
Loop	208	62	28	26	8	0	0	178
Recursive	96	37	42	8	2	0	0	124

Strength and Weakness

- Strength
  - This approach comes with a clean separation between the translation (semantics) and the use of the translation in proving the properties (computation).
  - This approach proves an assertion without explicitly generating loop invariants.
  - VIAP can effectively verify a number C programs from Arrays, Loops and Recursive sub-categories of ReachSafety category.
- Weakness
  - VIAP provides little or no support for translation and reasoning about dynamic linked data structures or programs with floating points.
  - It consumes more CPU time. The main overhead of VIAP comes from solving recurrences and in some cases applying the proof strategy.

Thanks

References I

Pritom Rajkhowa and Fangzhen Lin, *VIAP - automated system for verifying integer assignment programs with loops*, 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Timisoara, Romania, September 21-24, 2017 (Tudor Jebelean, Viorel Negru, Dana Petcu, Daniela Zaharie, Tetsuo Ida, and Stephen M. Watt, eds.), IEEE Computer Society, 2017, pp. 137–144.

______, *Extending viap to handle array programs*, 10th Working Conference on Verified Software: Theories, Tools, and Experiments, VSTTE 2018, Oxford, UK, July 18-19, 2018.