Termination-as-Safety Analysis in CPAchecker

Marek Jankola

September 9, 2024

LMU Munich, SoSy-Lab

Motivation

- Before SV-COMP 24 there was only one termination analysis in CPAchecker.
  - The analysis is based on LassoRanker [1].
  - It decomposes CFA into lassos and tries to find ranking function as terminating argument for each of them.
- Computing ranking functions can be expensive and slows the analysis if the program is non-terminating.

Motivation

- There exists work by A. Biere and V. Schuppan about transformation of liveness to safety properties [2].
- The new analysis utilizes this approach and transforms termination to reachability.
- It can explore the state space faster then the previous analysis and find non-terminating loops faster.

The Idea

```
int main() {
  int n = 0;
  n = n + 1;
  int z = nondet();
  while (n <= z) {
    n = (n - 1) \% 3;
    z = (z + 1) \% 3;
  }
  return 0;
}
```

Find an execution that will visit the same state twice:

(n,z) : $(1,2) \rightarrow (0,0) \rightarrow (-1,1) \rightarrow (-2,2) \rightarrow (0,0)$

- firstly occurred in 2. iteration
- secondly occurred in 5. iteration

BMC-like Approach

- Implemented new CPA that runs with Location and Predicate CPA:
  - Maps the loop-head locations to the number of visits,
  - creates a new variables that represent saving a state and maps them to the variables with SSA indices,
  - constructs a non-termination formula and checks satisfiability.

Example - 1. step

```
int main() {
  int n = 0;
  n = n + 1;
  int z = nondet();
  while (n <= z) {
    n = (n - 1) \% 3;
    z = (z + 1) \% 3;
  }
  return 0;
}
```

Let l_w be the location of the while loop.

The analysis starts with the initial maps $\{l_w \rightarrow -1\}, \varphi_{save} \equiv true$

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \quad SAT$$

$\{l_w \rightarrow -1\} \Rightarrow \{l_w \rightarrow 0\}$

$\varphi_{save} \equiv n@0 = n_1 \wedge z@0 = z_0$

Example - 1. step

Let $\varphi_i \equiv (n_{i+1} = (n_i - 1)\%3 \wedge z_i = (z_{i-1} + 1)\%3)$ express the body of the loop

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \ \wedge$$

$$\varphi_{save} \wedge \varphi_1 \wedge (n@0 = n_2 \wedge z@0 = z_1) \quad \textit{UNSAT}$$

Example - 1. step

Let $\varphi_i \equiv (n_{i+1} = (n_i - 1)\%3 \wedge z_i = (z_{i-1} + 1)\%3)$ express the body of the loop

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \ \wedge$$

$$\varphi_{save} \wedge \varphi_1 \wedge (n@0 = n_2 \wedge z@0 = z_1) \quad \mathit{UNSAT}$$

$$\{l_w \rightarrow 0\} \Rightarrow \{l_w \rightarrow 1\}$$

$$\varphi_{save} \equiv n@0 = n_1 \wedge z@0 = z_0 \wedge n@1 = n_2 \wedge z@1 = z_1$$

Example - 2. step

Let $\varphi_i \equiv (n_{i+1} = (n_i - 1)\%3 \wedge z_i = (z_{i-1} + 1)\%3)$ express the body of the loop

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \ \wedge$$

$$\varphi_{save} \wedge \varphi_1 \wedge \varphi_2 \wedge \big((n@0 = n_3 \wedge z@0 = z_2) \vee (n@1 = n_3 \wedge z@1 = z_2)\big) \quad \textit{UNSAT}$$

Example - 2. step

Let $\varphi_i \equiv (n_{i+1} = (n_i - 1)\%3 \wedge z_i = (z_{i-1} + 1)\%3)$ express the body of the loop

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \ \wedge$$

$$\varphi_{save} \wedge \varphi_1 \wedge \varphi_2 \wedge \big((n@0 = n_3 \wedge z@0 = z_2) \vee (n@1 = n_3 \wedge z@1 = z_2)\big) \quad \mathit{UNSAT}$$

$$\{l_w \rightarrow 1\} \Rightarrow \{l_w \rightarrow 2\}$$

$$\varphi_{save} \equiv n@0 = n_1 \wedge z@0 = z_0 \wedge n@1 = n_2 \wedge z@1 = z_1 \wedge n@2 = n_3 \wedge z@2 = z_2$$

Example - 4. step

In 3. step we again get UNSAT, but in 4. step:

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \ \wedge$$

$$\varphi_{save} \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \big((n@0 = n_4 \wedge z@0 = z_3) \vee (n@1 = n_4 \wedge z@1 = z_3) \vee$$

$$(n@2 = n_4 \wedge z@2 = z_3) \vee (n@3 = n_5 \wedge z@3 = z_4)\big) \quad SAT$$

Example - 4. step

In 3. step we again get UNSAT, but in 4. step:

$$n_0 = 0 \wedge n_1 = n_0 + 1 \wedge z_0 = r_0 \ \wedge$$

$$\varphi_{save} \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \big((n@0 = n_4 \wedge z@0 = z_3) \vee (n@1 = n_4 \wedge z@1 = z_3) \vee$$

$$(n@2 = n_4 \wedge z@2 = z_3) \vee (n@3 = n_5 \wedge z@3 = z_4)\big) \quad SAT$$

With the counterexample:
$(n@0 = 1, z@0 = 2), (n@1 = 0, z@1 = 0), (n@2 = -1, z@2 = 1),$
$(n@3 = -2, z@3 = 2), (n_5 = 0, z_4 = 0)$

Experiments

	Termination-as-Safety		LassoRankerAnalysis		Combination	
	status	cputime (s)	status	cputime (s)	status	cputime(s)
all results	2191	454000	2191	529000	2191	283000
correct results	1142	35300	983	43700	1342	100000
correct true	494	23700	472	27900	673	80100
correct false	648	11500	511	15900	669	20000
incorrect results	4	19.5	18	822	8	1890
incorrect true	4	19.5	1	7.88	1	307
incorrect false	0	0	17	814	7	1580

Easter Egg

```
int main() {
  int n = 0;
  n = n + 1;
  int z = nondet();
  while (n <= z) {
    n = (n - 1) \% 3;
    z = (z + 1) \% 3;
  }
  return 0;
}
```

```
int main() {
  int saved = 0; int n_instr; int z_instr;
  int n = 0;
  n = n + 1;
  int z = nondet();
  while (n <= z) {
    assert(saved = 0 || (n != n_instr || z
        != z_instr))
    if(saved == 0 && nondet())
      saved = 1; n_instr = n; z_instr = z
    n = (n - 1) \% 3;
    z = (z + 1) \% 3;
  }
  return 0;
}
```

Easter Egg - Experiments

	Transformation + BMC		Termination-as-Safety	
	status	cputime (s)	status	cputime (s)
all results	386	211000	386	255000
correct results	128	2670	99	1130
correct true	62	2060	34	568
correct false	66	604	65	557
incorrect results	0	0	3	21.1
incorrect true	0	0	3	21.1
incorrect false	0	0	0	0

Conclusion

- New termination analysis that aims to be fast in finding non-terminating loops.
- Outperforms the old analysis and the combination can leverage benefits of both.
- **Future work:** Generalization from BMC UNSAT queries to possible termination.

References i

[1] Leike, J., Heizmann, M.: Ranking templates for linear loops. Logical Methods in Computer Science **11**(1) (2015). https://doi.org/10.2168/LMCS-11(1:16)2015

[2] Schuppan, V., Biere, A.: Liveness checking as safety checking for infinite state spaces. Electr. Notes Theor. Comput. Sci. **149**(1), 79–96 (2006). https://doi.org/10.1016/j.entcs.2005.11.018