Progess Report 8

Xie Li

December 29, 2020

Overview of the Progress

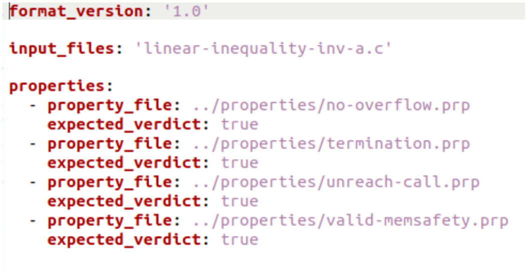
- After last survey of SV-COMP, implemented some utils help parse the information of SV-COMP cases.
- Debugging the tool with Weizhi.
- Other things...

Checking $\mathbf{G}(\neg$`reachError()`$)$

Since the uniqueness of function `reachError()`, checking can be done by

- Sample the paths from automaton. Sampler return the path once `reachError()` is met.
- Check the feasibility of the `reachError()` path.

Parsing



```yaml
format_version: '1.0'

input_files: 'linear-inequality-inv-a.c'

properties:
  - property_file: ../properties/no-overflow.prp
    expected_verdict: true
  - property_file: ../properties/termination.prp
    expected_verdict: true
  - property_file: ../properties/unreach-call.prp
    expected_verdict: true
  - property_file: ../properties/valid-memsafety.prp
    expected_verdict: true
```

Use C++ tool: yaml-cpp to do the parsing and extract:

- path of input file
- path of property file
- expected verdict

Current Problem Encountered

Interprocedural Analysis

- `void checkAssertion(bool condition)`
- In the translation, for every function we construct an automaton for the function.
- When sampling the path in the automaton. Merge the automaton of called procedure into the main automaton?

Checking Path Feasibility

After converting the CFG to an automaton e.g. DFA. We are able to sample a path of the automaton and do the path feasibility checking.
Current path feasibility check:

- Frontend: C$\rightarrow$ IR $\rightarrow$ CFG.
- Modelling: CFG$\rightarrow$ DFA, where alphabet of DFA is the set of instructions.
- Sampling a path and conjuncts condition induced by the instructions result in the path condition.

We are now able to generate path conditions for programs with only arithmetic operations.