Automatic Inference of Recursive Invariants Based on Catamorphisms

Oleynikov Andrey
MENTOR: Y. O. Kostyukov

August 30, 2023

Problem

Common problem:

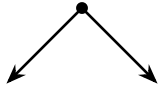
recursion in code exponentially increases the number of execution paths

Even harder problem:

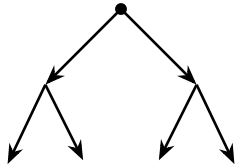
recursion in data structures exponentially increases execution states

Problem

Problem

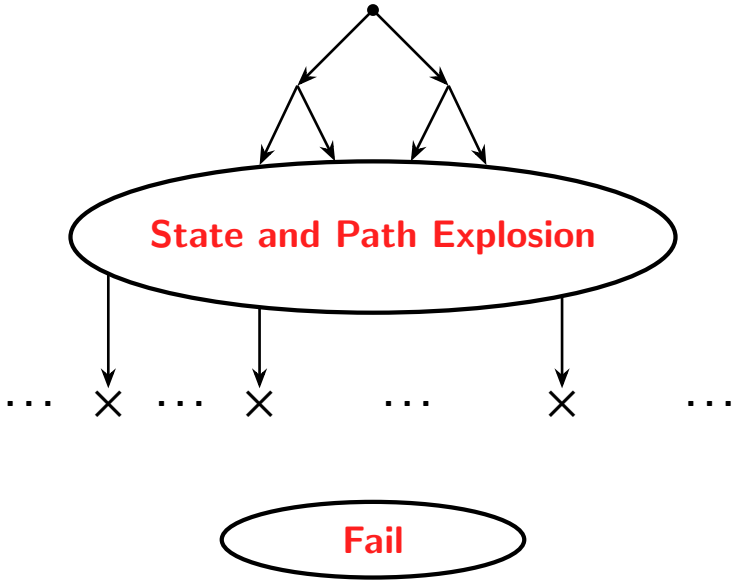


Problem

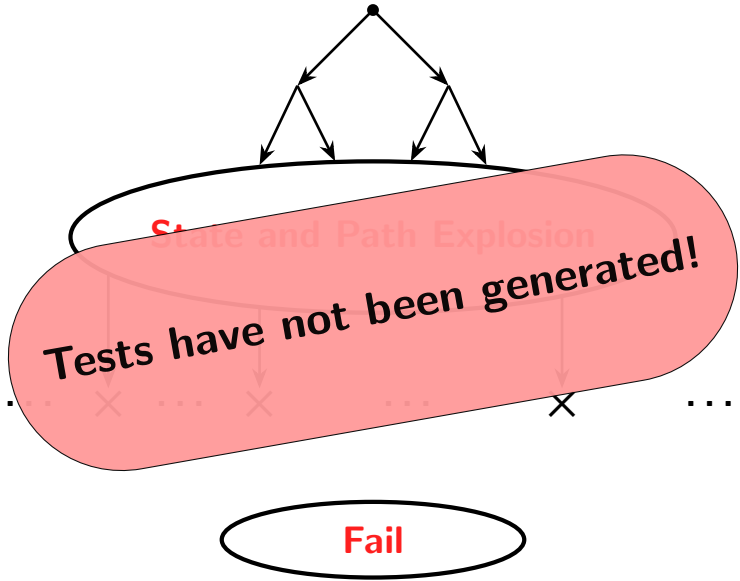


Fail

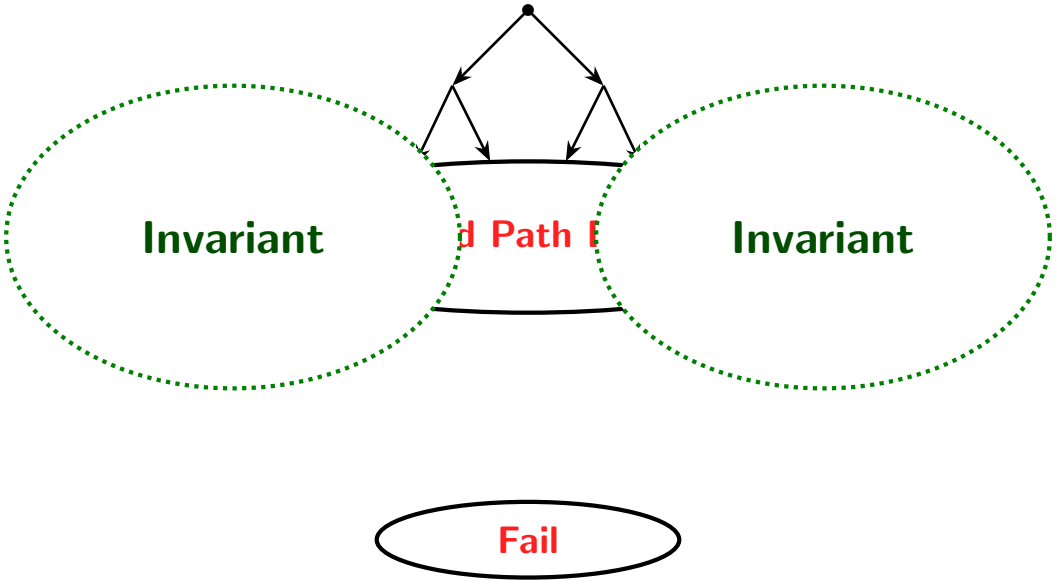
Problem



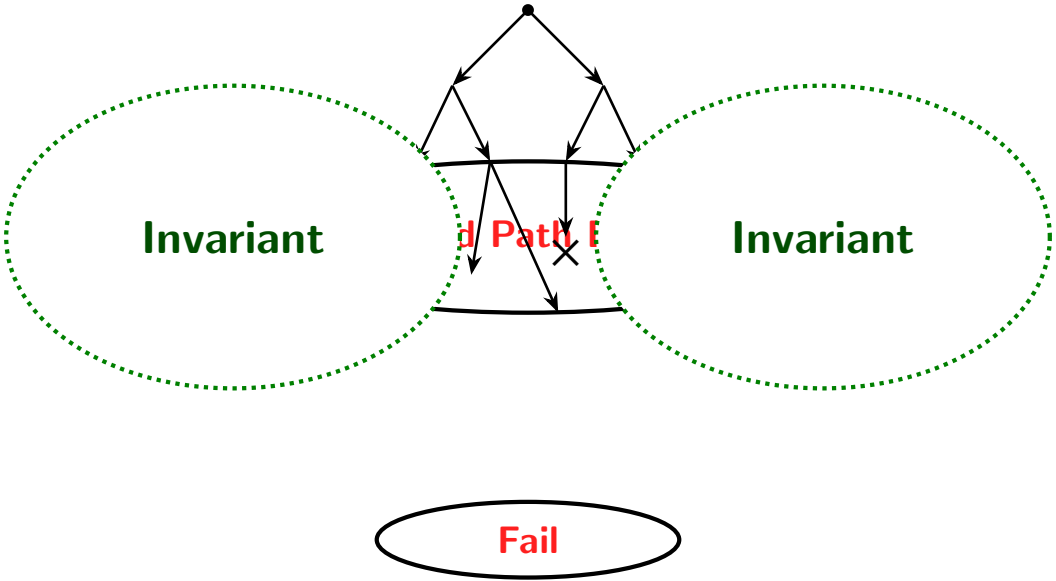
Problem


State and Path Explosion
Tests have not been generated!
Fail

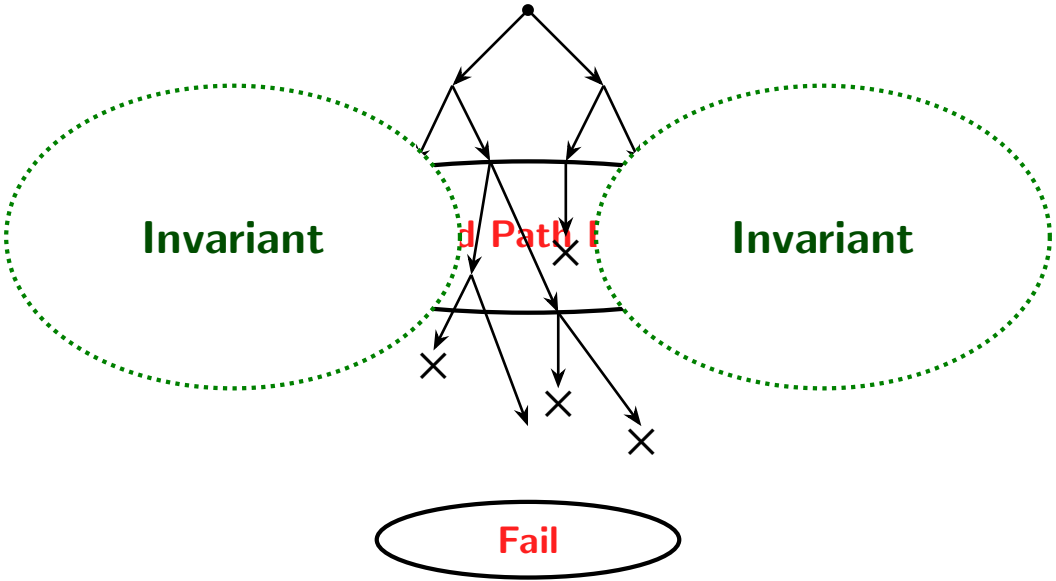
Problem



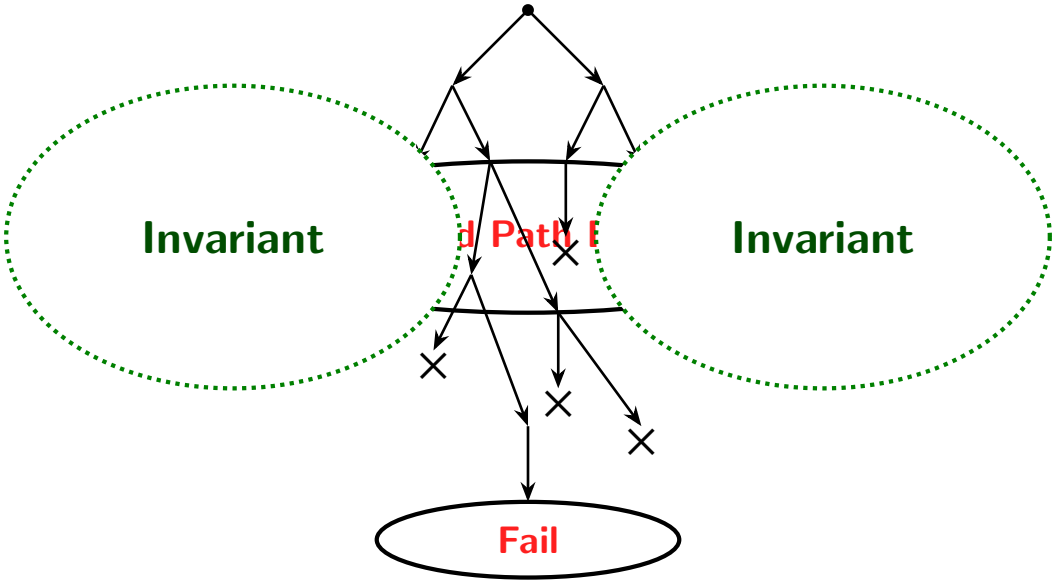
Problem



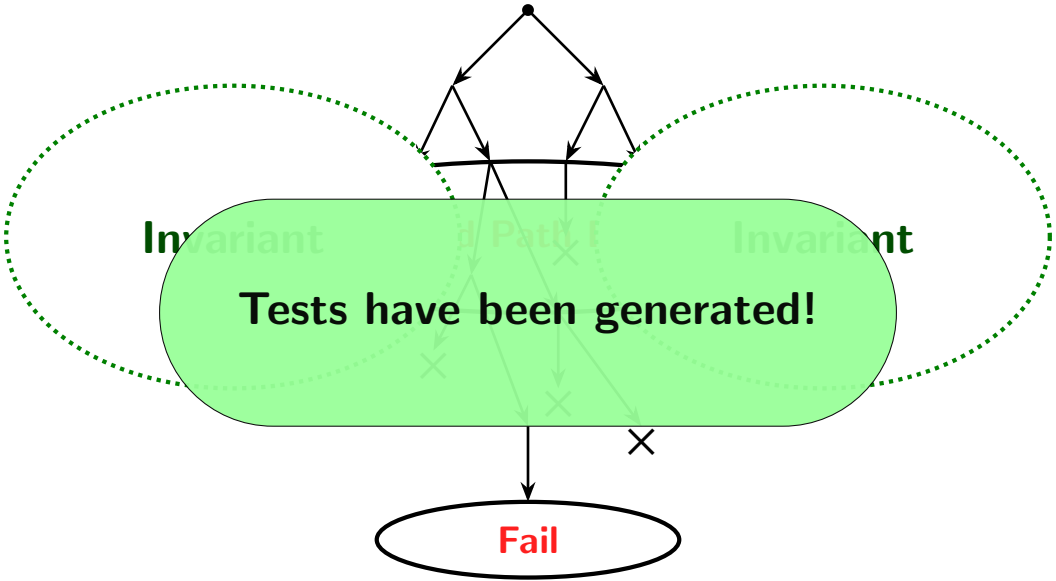
Problem



Problem



Problem



Goal

Goal: to infer invariants of programs with complex data structures
Requirements:

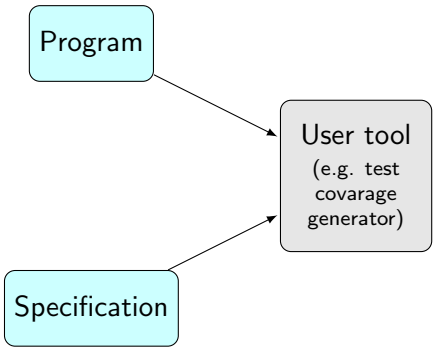
- Fully automatic
- Support data structures
- Return invariant
- Support SMT theory combination

Users

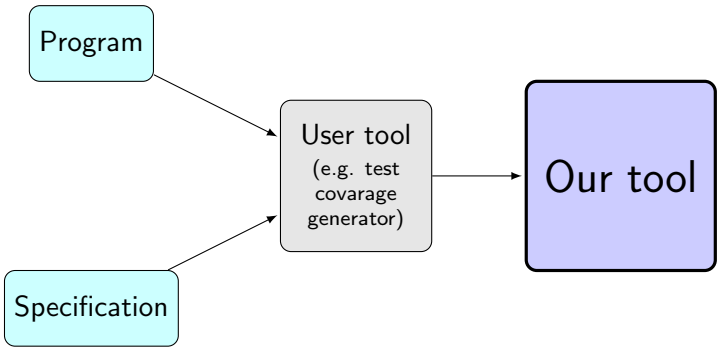
Software engineers who develop

- symbolic execution based tools
- static analyzers
- programs with complex data structures
- smart contracts

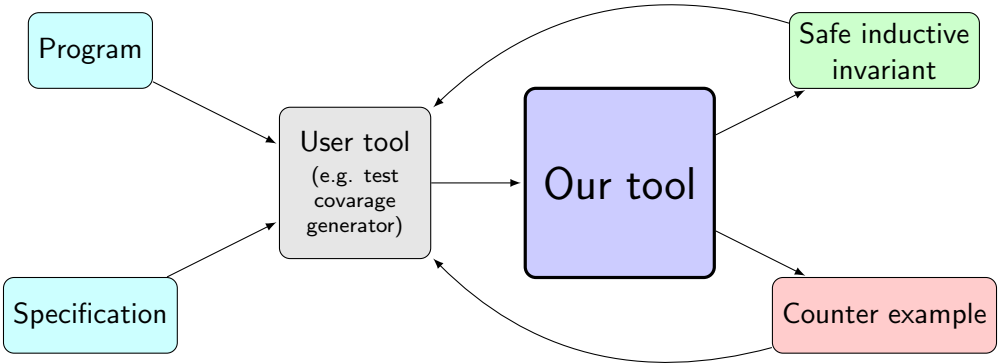
User story


Program
User tool
(e.g. test covarage generator)
Specification

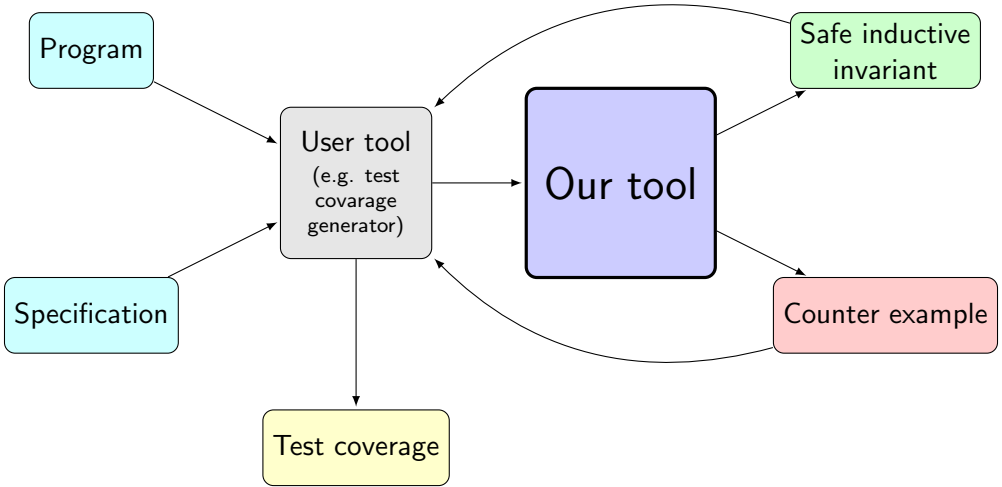
User story


Program
User tool
(e.g. test covarage generator)
Our tool
Specification

User story


Program
Specification
User tool
(e.g. test covarage generator)
Our tool
Safe inductive invariant
Counter example

User story


Program
Specification
User tool
(e.g. test covarage generator)
Test coverage
Our tool
Safe inductive invariant
Counter example

Proposed solution

Tool	SPACER	RACER	ELDARICA	HOICE	RCHC	RINGEN	OUR
Fully automatic	✔	✘	✔	✔	✔	✔	✔
Supports data structures	✘	✔	✔/✘	✘	✔	✔	✔
Returns invariant	✔	✘	✔	✔	✔	✔	✔
Supports SMT theory combination	✔	✔	✔	✔	✘	✘	✔

Proposed solution

Tool	SPACER	RACER	ELDARICA	HOICE	RCHC	RINGEN	OUR
Fully automatic	✔	✘	✔	✔	✔	✔	✔
Supports data structures	✘	✔	✔/✘	✘	✔	✔	✔
Returns invariant	✔	✘	✔	✔	✔	✔	✔
Supports SMT theory combination	✔	✔	✔	✔	✘	✘	✔

Idea: approximate data structures with simple schema

Results

https://github.com/ndreuu/adt-solver

Benchmark:

- recursive programs with complex data structures,
- i.e., **lists**, **trees**, **regular expressions**, **ASTs**, **maps**, **states**, **queues** etc.

ELDARICA

- inferred invariants: **13**
- found counterexamples: **18**

OUR SOLUTION

- inferred invariants: **24**
- found counterexamples: **7**

Results

https://github.com/ndreuu/adt-solver

Benchmark:

- recursive programs with complex data structures,
- i.e., **lists**, **trees**, **regular expressions**, **ASTs**, **maps**, **states**, **queues** etc.

ELDARICA

- inferred invariants: **13**
- found counterexamples: **18**

OUR SOLUTION

- inferred invariants: **24**
- found counterexamples: **7**

The theorem prover being used has bugs, which we have reported

Future work

- Investigate methods to help SMT solvers to handle quantifiers
- Improve counterexample rate
- Submit to CHC-COMP competition

Fin

Andrey Oleynikov
a.oleyn1kov@outlook.com

https://github.com/ndreuu/adt-solver