TracerX: Dynamic Symbolic Execution with Interpolation

Joxan JAFFAR*, **Rasool MAGHAREH**⋆,
Sangharatna Godboley†, Arpita Dutta*

*National University of Singapore, Singapore
{joxan,arpita}@comp.nus.edu.sg
⋆Huawei Canada Research Centre, Canada
rasool.maghareh@huawei.com
†National Institute of Technology Warangal, India
sanghu@nitw.ac.in

KLEE Workshop - June 2021

TracerX

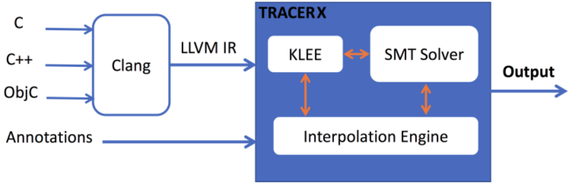
- Introducing *TracerX* symbolic execution approach
  - Based on the KLEE symbolic virtual machine
  - Perform *Interpolation* (information from already traversed (symbolic execution) subtree) to prune other subtrees

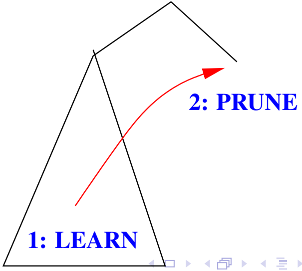
- Second place in RERS 2020 Challenge (+ Frama-C for unbounded prog.)
- Six place in Test-comp 2021 & 2020

- Website: `https://tracer-x.github.io/`
- Github: `https://github.com/tracer-x/`

From KLEE TO TracerX

- DFS Forward Symbolic Execution to find feasible paths (Similar to KLEE)
- Intermediate execution states preserved (Unlike KLEE)
- Path interpolants are generated for each path during backward tracking
- Tree interpolants are generated as conjunction of path interpolants
- Tree interpolants then used for subsumption at similar program points



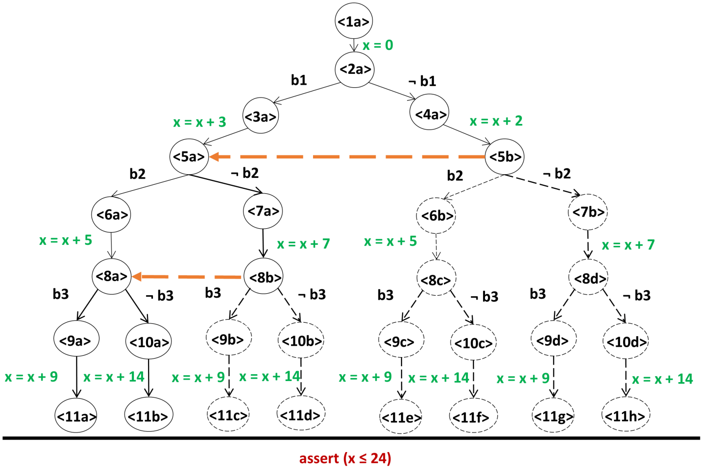


Symbolic Execution Tree with Interpolation

```
x = 0;
if (b1) x += 3 else x += 2
if (b2) x += 5 else x += 7
if (b3) x += 9 else x += 14
{x <= 24}
```

- Can we subsume (prune) ⟨5*b*⟩ with the tree interpolant generated at ⟨5*a*⟩?

- Similarly subsume ⟨8*b*⟩ with the interpolant at ⟨8*a*⟩?



Interpolation: Weakest Precondition

1. PATH Interpolant
   Path-based "weakest precondition"
2. TREE Interpolant
   Tree Interpolant are computed as conjunction of PATH interpolants

- Ideal interpolant is the weakest precondition (WP) of the target. Unfortunately, WP is intractable to compute

  Assume $(b1 \wedge \neg b2 \wedge \neg b3)$ is UNSAT.

  WP is:
  $b1 \longrightarrow (\neg b2 \wedge b3 \wedge x \leq 7) \vee (b2 \wedge x \leq 4)$
  $\neg b1 \longrightarrow x < 3$

- Essentially, WP is exponentially disjunctive
- Challenge is to obtain a conjunctive approximation

Interpolation: Approximation of Weakest Precondition

A Path is a sequence of assignment and assume instructions:

1. Interpolant of Assignment instruction:
   - $\textsc{wp}(inst, \omega) = \cdots$ inverse transition of *inst* over ω
   - Implemented at LLVM IR level: LD/ST, add, sub, cmp, cast, GEP, etc.
   - e.g. $\omega : x \leq 15$ and $inst : x = z + 2$, then $\textsc{wp}(inst, \omega) : z \leq 13$
2. Interpolant of Assume instruction (C is incoming Context):

   $\{C\}$
   assume(B)
   $\{\omega\}$
   - WP Approximation: find $\bar{C}$ to replace C
   - ABDUCTION PROBLEM !!!

Interpolation: Approximation of Weakest Precondition

This algorithm is the heart of TracerX:

1. We compute finest partition so that $var(C_i) * var(C_j)$ $s.t.$ $i \neq j$:
   $\{C_1 * C_2 * C_3 * ... * C_n\}$ assume(B) $\{\omega_1 * \omega_2 * \omega_3 * ... * \omega_m\}$
   ($*$ is as in separation logic).
2. Bunch C_i into three:
   - **Target independent:** The C_i which are separate from B and ω.
     Action: Replace C_i with *true*, i.e. remove C_i.
   - **Guard independent:** Consider $C_{gi} \equiv C_i$ s.t. $C_i * B$; and, $\omega_{gi} \equiv \omega_j$ s.t. $B * \omega_j$.
     Action: Replace C_{gi} by ω_{gi}.
   - **Remainder of the C_i:** We do not capture exact WP for this group.
     {z == 5} assume(x > z - 2) {x > 0} (e.g. $z > 2$ is the WP)
     Action: No change to C_i, i.e. keep C_i.

Interpolation: Approximation of Weakest Precondition

Note 1: Our algorithm is **fundamentally different** from CDCL in SMT solvers.
Note 2: We use no solver calls in our algorithm.

We have **OPTIONAL** algorithms for the **remainder of the C_i**.

1. **Elimination:** The WP is *true* and $x = 5$ can be eliminated.
   {x = 5}   assume(x < 7)   {x < 8}
2. **Projection:** The WP $z > 2$ can be computed by projection of $(x > z - 2) \wedge x > 0$ over z.
   {z == 5}   assume(x > z - 2)   {x > 0}
3. ...

The OPTIONAL algorithms can be turned on/off by user.

Experiment Setting

47 programs from SV-COMP and Reactive Systems Challenge (RERS)

- Industrial programs or have been used in testing and verification competitions

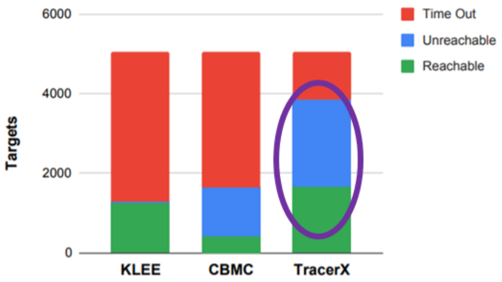
Two Experiments:

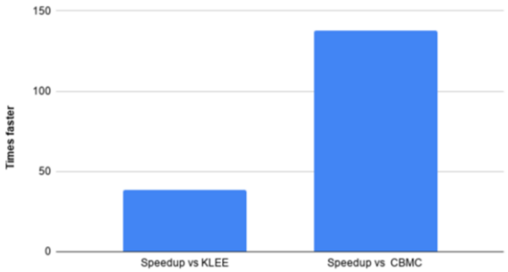
1. All targets: 5058
   - Each tool given 300 seconds for each target
2. Hard Targets: 1470
   - Not detected as reachable by KLEE in 5 minuets (representing testing)
   - Not detected as unreachable by Frama-C (representing static analysis)
   - Each tool given 600 seconds for each target

Presented in: **TracerX: Dynamic Symbolic Execution with Interpolation**
J. Jaffar, R. Maghareh, S. Godboley, X.L. Ha, 2020
`https://arxiv.org/abs/2012.00556`

Experiment - All Targets

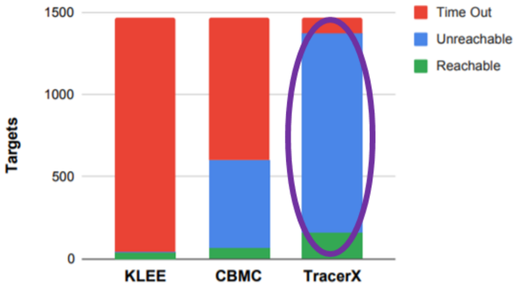
All targets: 5058 (300 seconds timeout)
TracerX wins in 1339 (26.57%) targets, while loses in only 112 (2.21%) targets
TracerX is 38.55x faster than KLEE and 137.56x faster than CBMC

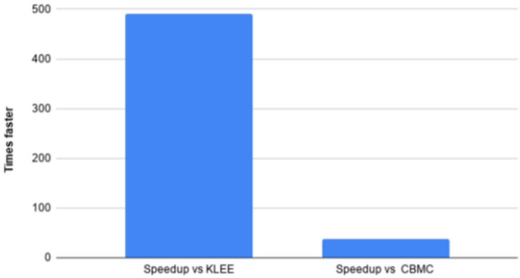




Experiment - Hard Targets

Hard targets: 1470 (600 seconds timeout)
TracerX wins in 796 (54.15%) targets, while loses in only 64 (4.35%) targets
TracerX is 490.26x faster than KLEE and 37.50x faster than CBMC





Future Directions

Testing:

- Modified Condition/Decision Coverage (MC/DC): A minimal set of test-cases needed to ensure the safety (ISSTA 2021)
- Guided search to find a path reaching a target test-case and proving non-existence if not found in the end of search

Incremental Quantitative Analysis:

- Ensure safety of non-functional features in embedded systems and safety critical systems

Combinatorial Optimization (COP):

- COP is widely applicable in AI
- Run TracerX on a program that simulates a COP problem and use Interpolation and Symmetry to prune (Submitted to CP 2021)

Conclusion

TracerX, Further Reading:

1. Website: `https://tracer-x.github.io/`
2. Github: `https://github.com/tracer-x/`
3. **TracerX: Dynamic Symbolic Execution with Interpolation**
   J. Jaffar, R. Maghareh, S. Godboley, X.L. Ha, 2020
   `https://arxiv.org/abs/2012.00556`
4. **TracerX: Dynamic Symbolic Execution with Interpolation (competition contribution)** J. Jaffar, R. Maghareh, S. Godboley, X.L. Ha, *FASE 2020*
5. **Toward Optimal MC/DC Test Case Generation**
   S. Godboley, J. Jaffar, R. Maghareh, A. Dutta, *ISSTA 2021*

Backup: WP Interpolation Example

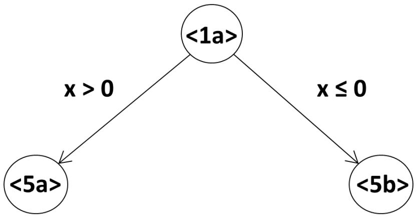
Compute path interpolant for left path:

1. **Target independent:** $a > 0$ (remove it).
2. **Guard independent:** $b = 5 \wedge c = 2 \wedge d = 4$
   Replace with $b \leq 580 \wedge c + 2d \leq 57$.
3. **Rest:** $-1 \leq x \leq 1$ (keep it).

Result (left path):
$b \leq 580 \wedge c + 2d \leq 57 \wedge -1 \leq x \leq 1$
Result (right path):
$b \leq 760 \wedge -1 \leq x \leq 1$
Tree Interpolant: Conjunction of both.
After applying OPTIONAL algorithm:
$b \leq 580 \wedge c + 2d \leq 57 \wedge -2 \leq x \leq 5$

Incoming Context:
$a > 0 \ \wedge \ b = 5 \ \wedge \ -1 \ \leq \ x \ \leq \ 1 \wedge \ c = 2 \wedge \ d = 4$

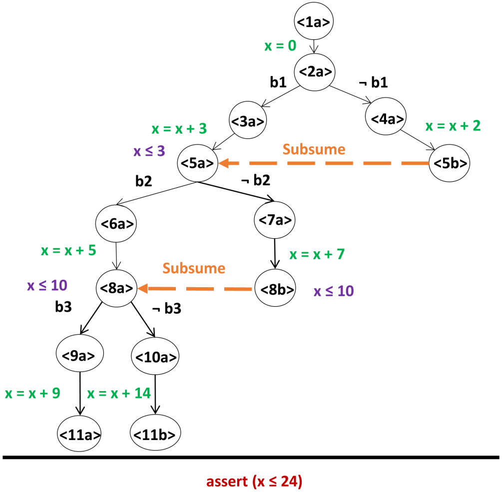
Tree intp: b ≤ 580 ∧ -2 ≤ x ≤ 5 ∧ c + 2d ≤ 57



Path intp 1: b ≤ 580 ∧ 0 ≤ x ≤ 5 ∧ c + 2d ≤ 57

Path intp 2: b ≤ 760 ∧ x ≥ -2

Backup: Full Example



- DFS traversal.
- **Without interpolation:** The full tree is traversed.
- **With interpolation:**
  1. $\langle 8b \rangle$ context contains $x = 10$. It is subsumed with the tree interpolant from $\langle 8a \rangle$: $x \leq 10$.
  2. $\langle 5b \rangle$ context contains $x = 2$. Subsumed with the tree interpolant from $\langle 5a \rangle$: $x \leq 3$.
  3. Big subtree traversal is avoided.