TracerX-Pruning Dynamic Symbolic Execution with Weakest Precondition Interpolation

Arpita Dutta[1], Rasool Maghareh[2], and Joxan Jaffar[3]

[1,3]National University of Singapore, Singapore
{joxan,arpita}@comp.nus.edu.sg
[2]Huawei Canada Research Centre, Canada
rasool.maghareh@huawei.com

4th International KLEE Workshop on Symbolic Execution
15-16 April 2024, Lisbon, Portugal

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

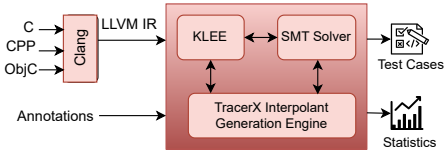


Figure: TracerX Framework

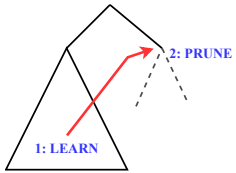
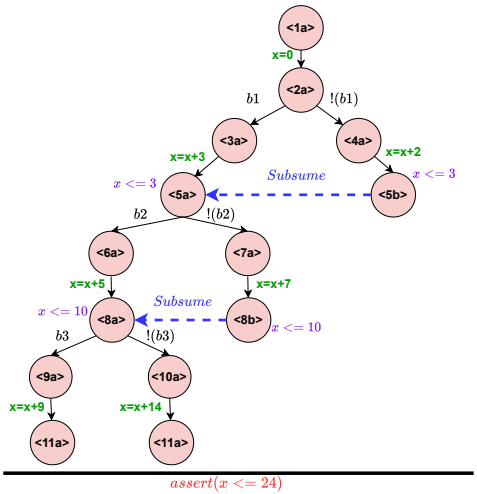


Figure: Pruning of subtree

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
assert(x <= 24)
```

- **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.

Interpolation: Weakest Precondition

- Ideal interpolant is the weakest precondition (WP) of the target. Unfortunately, WP is intractable to compute.
- For example, Assume $(b1 \wedge \neg b2 \wedge \neg b3)$ is **UNSAT**.
  WP before first "if-statement" 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

A Path is a sequence of assignment and assume instructions:

1. Interpolant of Assignment instruction:
   - $\mathrm{WP}(inst, \omega) = \cdots$ inverse transition of $inst$ over ω
   - e.g. $\omega : x \leq 15$ and $inst : x = z + 2$, then $\mathrm{WP}(inst, \omega) : z \leq 13$
2. Interpolant of Assume instruction (C is incoming Context): $\{C\}$ assume(B) $\{\omega\}$
   - WP Approximation: find $\bar{C}$ to replace C
   - ABDUCTION PROBLEM !!!

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:

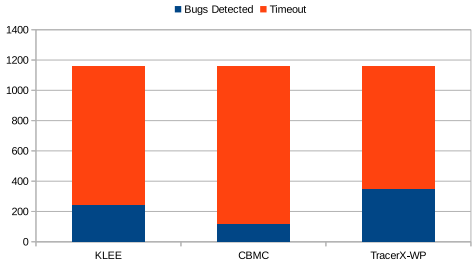
   - **Target independent:** The C_i which are separate from B and ω.
     Action: Replace C_i with *true*, i.e. remove C_i.
   - **Guard independent:** Consider $C_{gi} \equiv C_i$ s.t. $C_i * B$; and, $\omega_{gi} \equiv \omega_j$ s.t. $B * \omega_j$.
     Action: Replace C_{gi} by ω_{gi}.
   - **Remainder of the C_i:** We do not capture exact WP for this group.
     e.g. $\{z == 5\}$ assume$(x > z - 2)$ $\{x > 0\}$ (Here, $z > 2$ is the WP)
     Action: No change to C_i, i.e. keep C_i.

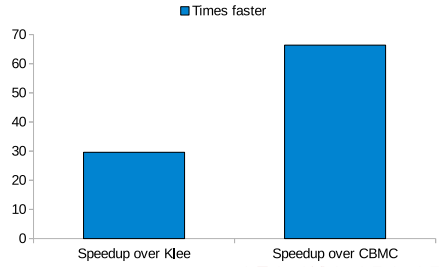
Experimental Results

Data set: All C-programs from RERS-2012 Challenge [6].

- **Total targets:** 1159
- All three systems **KLEE** [1], **CBMC** [5] and **TracerX-WP** [4] are run for 3600 seconds

1. TracerX-WP able to detect 348 targets, while KLEE and CBMC are detected 245 and 117 targets respectively.
2. TracerX-WP is 29.59x faster than KLEE and 66.37x faster than CBMC





Resources on TracerX

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,
5. **Toward Optimal MC/DC Test Case Generation**
   S. Godboley, J. Jaffar, R. Maghareh, A. Dutta, *ISSTA 2021*
6. **TracerX: Pruning Dynamic Symbolic Execution with Deletion and Weakest Precondition Interpolation (competition contribution)**
   A. Dutta, R. Maghareh, J. Jaffar, S. Godboley, X. L. Yu, *FASE 2024*

References

[1] C. Cadar et al. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: OSDI, 2008.

[2] J. Jaffar et al. TRACER: A symbolic execution tool for verification. In: CAV, 2012.

[3] J. Jaffar et al. TracerX: Dynamic symbolic execution with interpolation (competition contribution) . In: FASE, 2020.

[4] A. Dutta et al. TracerX: Pruning Dynamic Symbolic Execution with Deletion and Weakest Precondition Interpolation (competition contribution). In: FASE, 2024.

[5] D. Kroening D et al. CBMC-C Bounded Model Checker. In: TACAS 2014.

[6] http://rers-challenge.org/