The TRACER-X System

Joxan JAFFAR & Rasool MAGHAREH

Department of Computer Science, National University of Singapore
*{joxan, rasool}@comp.nus.edu.sg

April 2018
(KLEE Workshop 2018)
Introducing *TRACER-X* symbolic execution approach

- Based on the KLEE symbolic virtual machine

- *Interpolation* for search-space reduction

- **TRACER-X**
- **Github**: [https://github.com/tracer-x/](https://github.com/tracer-x/)
1. Mitigating Search-Space Complexity with Interpolation
2. TRACER-X (KLEE with Interpolation)
3. Weakest Precondition Interpolation
4. Memory Bounds Interpolation
5. Symbolic Heap
6. Results & Current Directions
Problem and Solution

- Naive analysis/verification (e.g., standard model checking) → huge search space: exponential in the size of the program
- To mitigate the problem we employ learning

1: LEARN

2: PRUNE

We use information from already traversed (symbolic execution) subtree to prune other subtrees
Example: Proving Safety

Initially $x > 0$

\[ \langle 0 \rangle \quad \text{if} \ (a = 1) \ \text{then} \ \langle 1 \rangle \quad \text{skip} \ \text{endif} \]

\[ \langle 2 \rangle \quad \text{if} \ (b = 1) \ \text{then} \ \langle 3 \rangle \quad c := 0 \ \text{endif} \]

\[ \langle 4 \rangle \quad \text{if} \ (c = 1) \ \text{then} \ \langle 5 \rangle \quad x := x + 1 \ \text{endif} \]

\[ \langle 6 \rangle \quad \text{assert}(x > 0) \]

Next: The Tree
Symbolic Execution Tree

Constraints with versioned variables for a path in the tree:

\[ x_0 > 0 \langle 0 \rangle a_0 = 1 \langle 1 \rangle \langle 2 \rangle b_0 = 1 \langle 3 \rangle c_1 = 0 \langle 4 \rangle c_1 = 1 \langle 5 \rangle x_1 = x_0 + 1 \langle 6 \rangle \]
Interpolation

- **HALF Interpolant**
  - Path-based “weakest precondition”
  - (Often easy to compute)

- **FULL Interpolant**
  - Combine half interpolants to become Tree-based
  - (Challenge is to obtain compact representation)

Example of the Most Basic Interpolation Method: **UNSAT-CORE**

\[
\begin{align*}
  x_0 &> 0 & a_0 &= 1 & b_0 &= 1 & c_1 &= 0 \\
  c_1 &= 1 & x_1 &= x_0 + 1
\end{align*}
\]

The above constraints are *unsatisfiable*, remove constraints that are not needed to ensure unsatisifiability

\[
\langle 0 \rangle \langle 1 \rangle \langle 2 \rangle \langle 3 \rangle c_1 = 0 \langle 4 \rangle \langle 5 \rangle \langle 6 \rangle
\]
Initially \( x > 0 \)

\[
\begin{align*}
\langle 0 \rangle & \text{ if } (a = 1) \text{ then } \langle 1 \rangle \text{ skip endif} \\
\langle 2 \rangle & \text{ if } (b = 1) \text{ then } \langle 3 \rangle c := 0 \text{ endif} \\
\langle 4 \rangle & \text{ if } (c = 1) \text{ then } \langle 5 \rangle x := x + 1 \text{ endif} \\
\langle 6 \rangle & \text{ assert}(x > 0)
\end{align*}
\]

- DFS traversal.
- **W/o interpolation:** The full tree is traversed.
- **W/ interpolation:** (A) is \( x > 0 \), (B) is \( x > 0 \), \( a \neq 1 \), hence (B) is subsumed by (A), big subtree traversal is avoided.

\( \models x > 0 \) SAFE!
From KLEE TO TRACER-X

- Forward Symbolic Execution to find feasible paths (Similar to KLEE)
- Intermediate execution states preserved (Unlike KLEE)
- Half interpolants are generated by backward tracking
- Full interpolants generated by merging half interpolants
- Full interpolants used for subsumption at similar program points

Figure: Tracer-X Framework
Weakest Precondition VS Strongest Postcondition

- **WP**
  - Goal-directed and often small formula, per path.
  - Unfortunately, not easy to compress individual path WP.
  - Biggest disadvantage: agnostic to context.
  - (eg: Example above, if x had initial value.)

- **SP**
  - Not goal-directed and often large formula, for all paths.
  - Per path reasoning is precise.

- **SP with Interpolation**
  - Can exploit learning from the unsat-core: basic interpolation.
  - A remaining disadvantage: interpolation needs to infer new information beyond unsat-core.
Interpolation: Weakest Precondition

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

```c
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}
```

Assume \((b_1 \land \neg b_2 \land \neg b_3)\) is UNSAT. WP is:

- \(b_1 \rightarrow (\neg b_2 \land b_3 \land x \leq 7) \lor (b_2 \land x \leq 4)\)
- \(\neg b_1 \rightarrow x < 3\)

- Essentially, WP is **exponentially disjunctive**
First the Easy Cases:

suppose a context of $\tilde{c}$.

- $WP(t, \omega) = \cdots$ LLVM inverse transition of $t$

- $WP(assume(b), \omega) = \omega \land b$

- $WP(if(b) then S1 else S2, \omega) = \omega \land b$ where $\tilde{c} \models b$

- Similarly for when $\tilde{c} \models \neg b$
The General Case:
if \( (b) \) then \( S_1 \) else \( S_2 \) with postcondition \( \omega \) where

- the context is \( \tilde{c} = c_1, c_2, \ldots, c_n \).
- Neither \( \tilde{c} \models b \) nor \( \tilde{c} \models \neg b \) holds.
- \( wpp(S_1, \omega) \) is \( \omega_1 \) and \( wpp(S_2, \omega) \) is \( \omega_2 \)

In general, the weakest precondition \( \Psi \) is a disjunction:
\[
(b \rightarrow \omega_1) \land (\neg b \rightarrow \omega_2)
\]

We want to compute a convex \( \Phi \). (Therefore \( \tilde{c} \models \Phi \models \Psi \))

Takeaway:

- There is no succinct definition for this convex.
- The above examples show, however, that there are many special cases to exploit.
Choose a candidate to generalize:
\[ c = 2 \land d = 4 \]

Extract the subset of \( W_1 \) and \( W_2 \) which share the same variables with \( c = 2 \land d = 4 \):
- Subset of \( W_1 \):
  \[ c + 2d \leq 57 \]
- Subset of \( W_2 \): \{\}

If one subset is empty, generalize the candidate to the other subset:
\[ c + 2d \leq 57 \]

Original Context:
\[ a > 0 \land b = 5 \land -1 \leq x \leq 1 \land c = 2 \land d = 4 \]

\[ b \leq 580 \land -2 \leq x \leq 5 \land c + 2d \leq 57 \]

\[ W_1 : b \leq 580 \land 0 < x < 5 \land c + 2d \leq 57 \]
\[ W_2 : b \leq 760 \land x \geq -2 \]

\[ x > 0 \]
\[ x \leq 0 \]
When generalizing, arrays candidates should be chosen and generalized carefully:

- **Candidate:**
  
  \[
  \text{int } a[100] \land p = a + 7 \\
  \land *p = 0 \land *a + 6 = 5
  \]

- **Generalization:**
  
  \[
  \text{int } a[100] \land p \leq a + 1000 \\
  \land *p = 0 \land *a + 6 = 5
  \]

Note that the generalization of \( p \) does not include \( p = a + 6 \).
\[ (0) \quad p = \text{malloc}(5) \]
\[ (1) \begin{array}{l}
\text{if (\ldots) then} \\
p++
\end{array} \]
\[ \text{else} \\
p+ = 2 \]
\[ \text{endif} \]
\[ (2) \quad c := *p \]
```c
#define MAX 18
n = input(); // getting a symbolic input
x = malloc(1); *x = n;
for (int i = 0; i < MAX; i++) {
    if (*) { y = malloc(1); *y = *x+1; }
    else { y = malloc(1); *y = *x+1; }
    x = y;
}
```

- `malloc()` is nondeterministic, but enjoys separation
- Branches (essentially) identical
- Times out using KLEE and LLBMC (30 mins)
- Exponential running time for both KLEE and LLBMC (and potentially Veritesting)
Is (B) Subsumed by (A)?

In dynamic symbolic execution and even LLBMC, different concrete values are returned by each `malloc` call (satisfies separation)

→ both states cannot be matched
Is (B) Subsumed by (A)?

Our approach:

- We regard dynamically-allocated addresses symbolically:
  \(1024 = x_0\), \(2048 = y_1\), \(3072 = z_1\).
- Matching: \((y_1, z_1) \rightarrow \text{subsumption holds!}\)
Symbolic Heap Interpolation of TRACER-X

\[ \exists z_1. \left( \begin{array}{l}
  x_0 \leftrightarrow n_0 \land z_1 \leftrightarrow n_0 + 1 \land \\
  n \leftrightarrow n_0 \land x \leftrightarrow z_1 \land y \leftrightarrow z_1
\end{array} \right) \models \]

\[ \exists y_1. \left( \begin{array}{l}
  x_0 \leftrightarrow n_0 \land y_1 \leftrightarrow n_0 + 1 \land \\
  n \leftrightarrow n_0 \land x \leftrightarrow y_1 \land y \leftrightarrow y_1
\end{array} \right) \]

**Existentials**: \( z_1, y_1 \): some addresses dynamically allocated

**Problem**: Prove subsumption by eliminating existentials

\[ \rightarrow \text{SMT solvers are weak in solving quantified formulas} \]

- General problem is NP-Complete or harder (conjecture)
- **Must** use specialized quantifier elimination techniques
Symbolic Heap Interpolation of TRACER-X

\[
\begin{align*}
(x_0 &\mapsto n_0 \land z_1 \mapsto n_0 + 1 \land \\
n &\mapsto n_0 \land x \mapsto z_1 \land y \mapsto z_1)
\end{align*}
\]

\[
\models \exists y_1. \quad \begin{align*}
(x_0 &\mapsto n_0 \land y_1 \mapsto n_0 + 1 \land \\
n &\mapsto n_0 \land x \mapsto y_1 \land y \mapsto y_1)
\end{align*}
\]

\[
\begin{align*}
(x_0 &\mapsto n_0 \land z_1 \mapsto n_0 + 1 \land \\
n &\mapsto n_0 \land x \mapsto z_1 \land y \mapsto z_1)
\end{align*}
\]

Procedure:

1. Unquantify antecedent variables: \(z_1\) becomes free
2. Elimination done by traversal from global/local variables and finding matching substitutions that would work. In our case, \([z_1/y_1]\), replacing existentially-quantified \(y_1\) with free \(z_1\).
3. Solve subsumption using SMT solver via entailment without quantification.
4. In general, compute data structure homomorphisms for quantifier elimination (In general, intractable, but often easy.)
**Figure:** (Both TRACER-X and KLEE Finish Execution)

### KLEE vs. TRACER-X - Analysis Time

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>KLEE Time (s)</th>
<th>TX Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sync</td>
<td>0.25</td>
<td>0.07</td>
</tr>
<tr>
<td>pwd</td>
<td>1.07</td>
<td>0.23</td>
</tr>
<tr>
<td>chroot</td>
<td>14.22</td>
<td>5.29</td>
</tr>
<tr>
<td>dimname</td>
<td>24.92</td>
<td>3.9</td>
</tr>
<tr>
<td>users</td>
<td>26.64</td>
<td>5.52</td>
</tr>
<tr>
<td>uptime</td>
<td>28.26</td>
<td>4.65</td>
</tr>
<tr>
<td>runcon</td>
<td>28.53</td>
<td>5.45</td>
</tr>
<tr>
<td>cksum</td>
<td>60.25</td>
<td>41.06</td>
</tr>
<tr>
<td>basename</td>
<td>70.48</td>
<td>257.86</td>
</tr>
<tr>
<td>printenv</td>
<td>257.86</td>
<td>512.59</td>
</tr>
</tbody>
</table>

Average Time:

- Benchmark: 512.59
- Average: 70.92
- Average: 0.1383561911
## Table: (TRACER-X Finishes Execution but KLEE does not Finish)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>LOC</th>
<th>KLEE (TIMEOUT: 3600 S)</th>
<th>TRACER-X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#ERR #EP #PEP</td>
<td>T (s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>#ERR #EP #SP #PEP</td>
</tr>
<tr>
<td>base64</td>
<td>401</td>
<td>0 0 2115667</td>
<td>3327.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 294256 20551</td>
</tr>
<tr>
<td>cat</td>
<td>339</td>
<td>1 2761 2729703</td>
<td>1824.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 45546 127526 14128</td>
</tr>
<tr>
<td>chcon</td>
<td>604</td>
<td>0 0 1663596</td>
<td>1927.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 221628 15722</td>
</tr>
<tr>
<td>chgrp</td>
<td>612</td>
<td>0 0 778227</td>
<td>2461.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 150836 34330</td>
</tr>
<tr>
<td>comm</td>
<td>255</td>
<td>0 0 1860052</td>
<td>2748.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 67496 39677</td>
</tr>
<tr>
<td>df</td>
<td>547</td>
<td>2 2108 13114</td>
<td>900.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 1456 1531 2321</td>
</tr>
<tr>
<td>dircolors</td>
<td>241</td>
<td>0 0 1824002</td>
<td>3366.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 115004 7883</td>
</tr>
<tr>
<td>env</td>
<td>286</td>
<td>0 0 1846675</td>
<td>63.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 5078 3124</td>
</tr>
<tr>
<td>fold</td>
<td>98</td>
<td>0 0 1959113</td>
<td>1292.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 46899 49494</td>
</tr>
<tr>
<td>head</td>
<td>482</td>
<td>1 4 1950422</td>
<td>2438.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 3 6323 3375</td>
</tr>
<tr>
<td>hostid</td>
<td>175</td>
<td>0 0 2107218</td>
<td>3494.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 1 332198 19060</td>
</tr>
<tr>
<td>hostname</td>
<td>180</td>
<td>0 0 2323263</td>
<td>968.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 116020 7876</td>
</tr>
<tr>
<td>ln</td>
<td>497</td>
<td>0 0 578519</td>
<td>3064.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 145226 16363</td>
</tr>
<tr>
<td>logname</td>
<td>181</td>
<td>0 0 2125296</td>
<td>3018.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 315266 18633</td>
</tr>
<tr>
<td>mkdir</td>
<td>237</td>
<td>0 0 902244</td>
<td>1964.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 53072 35288</td>
</tr>
<tr>
<td>mkfifo</td>
<td>206</td>
<td>0 0 906846</td>
<td>1930.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 4 52775 35516</td>
</tr>
<tr>
<td>mknod</td>
<td>597</td>
<td>2 155 2519413</td>
<td>3300.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 2 243958 15422</td>
</tr>
<tr>
<td>mktemp</td>
<td>650</td>
<td>0 0 2448222</td>
<td>3131.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 278334 9539</td>
</tr>
<tr>
<td>nice</td>
<td>238</td>
<td>0 0 2372168</td>
<td>215.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 16973 1963</td>
</tr>
</tbody>
</table>
Table: (TRACER-X Finishes Execution but KLEE does not Finish)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>LOC</th>
<th>KLEE (TIMEOUT: 3600 S)</th>
<th></th>
<th></th>
<th>TRACER-X</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>nl</td>
<td>1293</td>
<td>#ERR</td>
<td>0</td>
<td>#EP</td>
<td>0</td>
<td>#PEP</td>
</tr>
<tr>
<td>nohup</td>
<td>209</td>
<td>0</td>
<td>0</td>
<td>832849</td>
<td>2351.5</td>
<td>0</td>
</tr>
<tr>
<td>paste</td>
<td>135</td>
<td>0</td>
<td>0</td>
<td>1760657</td>
<td>754.7</td>
<td>0</td>
</tr>
<tr>
<td>pinky</td>
<td>514</td>
<td>0</td>
<td>0</td>
<td>461789</td>
<td>445.9</td>
<td>0</td>
</tr>
<tr>
<td>pr</td>
<td>598</td>
<td>0</td>
<td>0</td>
<td>528712</td>
<td>900.9</td>
<td>0</td>
</tr>
<tr>
<td>printf</td>
<td>553</td>
<td>0</td>
<td>0</td>
<td>2065362</td>
<td>2736.7</td>
<td>0</td>
</tr>
<tr>
<td>readlink</td>
<td>301</td>
<td>0</td>
<td>0</td>
<td>3076444</td>
<td>8.8</td>
<td>0</td>
</tr>
<tr>
<td>rm</td>
<td>656</td>
<td>0</td>
<td>0</td>
<td>1330799</td>
<td>1341.9</td>
<td>0</td>
</tr>
<tr>
<td>rmdir</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>765438</td>
<td>2174.1</td>
<td>0</td>
</tr>
<tr>
<td>setuidgid</td>
<td>290</td>
<td>0</td>
<td>0</td>
<td>1124960</td>
<td>1578.9</td>
<td>0</td>
</tr>
<tr>
<td>shred</td>
<td>472</td>
<td>0</td>
<td>0</td>
<td>22206</td>
<td>397.1</td>
<td>0</td>
</tr>
<tr>
<td>sleep</td>
<td>204</td>
<td>0</td>
<td>0</td>
<td>926939</td>
<td>1778.6</td>
<td>0</td>
</tr>
<tr>
<td>tee</td>
<td>88</td>
<td>0</td>
<td>0</td>
<td>2000807</td>
<td>67.9</td>
<td>0</td>
</tr>
<tr>
<td>tty</td>
<td>176</td>
<td>0</td>
<td>0</td>
<td>1564733</td>
<td>1160.6</td>
<td>0</td>
</tr>
<tr>
<td>unlink</td>
<td>177</td>
<td>0</td>
<td>0</td>
<td>1596777</td>
<td>970.7</td>
<td>0</td>
</tr>
<tr>
<td>who</td>
<td>749</td>
<td>0</td>
<td>0</td>
<td>2903432</td>
<td>909.3</td>
<td>0</td>
</tr>
<tr>
<td>whoami</td>
<td>183</td>
<td>0</td>
<td>0</td>
<td>2106260</td>
<td>3164.8</td>
<td>0</td>
</tr>
</tbody>
</table>
Note our good performance on coverage.
Current Directions: Testing

- Modified Condition/Decision Coverage (MC/DC): A minimal set of test-cases needed to ensure the safety

- DSE-based approaches: Unguided search for test-cases
- Cannot prove test-case non-existence (not fully traversed SET)

- TRACER-X Approach:
  - Guided search to find a path reaching a target test-case
  - Proving non-existence of a test-case if not found in the end of search
Current Directions: Incremental Quantitative Analysis

- **Quantitative Analysis**: Ensure safety of non-functional features in embedded systems and IoT

- **Exact Methods**: Not Scalable
- **Abstraction-based methods**: Scalable but Inaccurate

- **TRACER-X Approach**:
  - Given an upper and lower-bound check the mid-point
  - If safe: Decrease the upper-bound to the mid-point
  - If counter-example found: Increase the lower-bound (unavailable for abstraction-based analyses)

- **progressively** increasing certified accuracy
- **Stop-any-time**
- **Dynamic** Resource Cost Model
COP is widely applicable in AI
A good solution is usually good enough

Traditional methods: Mathematical Programming & Constraint Programming

TRACER-X Approach:
Run TRACER-X on a program that check a given solution
Maintain lower and upper-bounds (similar to Quantitative Analysis)
Use Interpolation and Symmetry to prune

progressively walking towards optimal solution
Stop-any-time
Conclusion

- **TRACER-X:**
- **Website:** [http://www.comp.nus.edu.sg/~tracerx](http://www.comp.nus.edu.sg/~tracerx)
- **Github:** [https://github.com/tracer-x/](https://github.com/tracer-x/)
- (with Unsat-Core & some Weakest-Precondition interpolation)