HiFrog: Interpolation-based Software Verification using Theory Refinement

Sepideh Asadi, Karine Even Mendoza, Grigory Fedyukovich, Antti E.J. Hyvarinen, Hanna Chockler, Natasha Sharygina
University of Lugano (USI), Switzerland, King's College London, UK, University of Washington, USA

What?
A bounded model checker for verification of safety properties for C programs.

How?
Uses function summaries based on Craig interpolation and supports a novel technique for abstraction refinement.

Why?
To avoid repetition of verification tasks while checking multiple properties of same code and to eliminate spurious behaviors!

Interpolation
➢ For $A \land B$ is unsatisfiable, $I$ is a quantifier-free formula such that:
- $A \Rightarrow I$
- $B \land I$ is unsatisfiable
- $I$ is defined over common symbols of $A$ and $B$

Example
➢ Use of function summary in a C program with assertions

```
void main(){
    int y = 1;
    int x = nondet();
    if (x > 0) {
        assert (y >= 1);
    }
}
```

C program + assertions

Theory Refinement Algorithm

```
input : $P = \{x_1 = t_1, \ldots, x_n = t_n\}$: a program, and $t$: a safety property output:
(Safe, I) or (Unsafe, CE)
```

For all $1 \leq i \leq n$ if initial $p[i] = t_i \rightarrow [\neg I, I^t, F_I]$

Features?
- SMT-based Bounded Model Checker
- Controllable interpolation system for SMT (flexible in Size & Strength)
- A new approach called Theory Refinement to have simple proofs using SMT
- Automatically identifies where precision is needed and uses precise theories only when necessary
- Support of user-defined summaries

Architecture

Experiments on SMT vs. Boolean Logic

```
<table>
<thead>
<tr>
<th>C Benchmarks</th>
<th>Assertion</th>
<th>EUF</th>
<th>LRA</th>
<th>Bool</th>
</tr>
</thead>
<tbody>
<tr>
<td>token.c</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>sil.c</td>
<td>131</td>
<td>18</td>
<td>21</td>
<td>26</td>
</tr>
<tr>
<td>mem.c</td>
<td>149</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>disk.c</td>
<td>79</td>
<td>6</td>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>ddc.c</td>
<td>152</td>
<td>47</td>
<td>47</td>
<td>142</td>
</tr>
<tr>
<td>cafe.c</td>
<td>115</td>
<td>15</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>tcs.assert.c</td>
<td>102</td>
<td>16</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>p2p.c</td>
<td>244</td>
<td>8</td>
<td>20</td>
<td>94</td>
</tr>
<tr>
<td>hoppylc</td>
<td>18</td>
<td>15</td>
<td>16</td>
<td>18</td>
</tr>
</tbody>
</table>
```

- Percentage of success: 50.65% 69.2% 100%
- The verification with EUF and LRA is an order of magnitude faster than the verification with BOOL.

Advantages:
- On average, HiFrog is 65.6% faster than EUF and 94.2% faster than LRA.
- For benchmarks such as hoppylc the speedup is even more pronounced.
- Improvements in terms of verification time and memory usage are also observed.

Experiments on Theory Refinement

```
<table>
<thead>
<tr>
<th>CBMC (s)</th>
<th>Theory flatting (s)</th>
<th>BOOL (s)</th>
<th>HiFrog flatting (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 100</td>
<td>157</td>
<td>50.65%</td>
<td>100</td>
</tr>
<tr>
<td>2 100</td>
<td>157</td>
<td>50.65%</td>
<td>100</td>
</tr>
<tr>
<td>3 100</td>
<td>157</td>
<td>50.65%</td>
<td>100</td>
</tr>
</tbody>
</table>
```

- On average, HiFrog is 65.6% faster than EUF and 94.2% faster than LRA.
- For benchmarks such as hoppylc the speedup is even more pronounced.
- Improvements in terms of verification time and memory usage are also observed.

Theory Refiner

- Itp for QF-BOOL
- BVP
- SMT encoder
- QF_BOOL
- QF_UF
- QF_LRA

Experiments on Theory Refinement

```
<table>
<thead>
<tr>
<th>CBMC (s)</th>
<th>Theory flatting (s)</th>
<th>BOOL (s)</th>
<th>HiFrog flatting (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 100</td>
<td>157</td>
<td>50.65%</td>
<td>100</td>
</tr>
<tr>
<td>2 100</td>
<td>157</td>
<td>50.65%</td>
<td>100</td>
</tr>
<tr>
<td>3 100</td>
<td>157</td>
<td>50.65%</td>
<td>100</td>
</tr>
</tbody>
</table>
```

- On average, HiFrog is 65.6% faster than EUF and 94.2% faster than LRA.
- For benchmarks such as hoppylc the speedup is even more pronounced.
- Improvements in terms of verification time and memory usage are also observed.

Publications:
2) Theory Refinement for Program Verification, SAT 2017.

Webpage: http://verify.inf.usi.ch/hifrog