End-to-end Verification of Stack-space Bounds for C Programs
Quentin Carbonneaux, Jan Hoffmann, Tahina Ramananandro and Zhong Shao

“A with experience, one learns the standard, scientific way to compute the proper size for a stack: Pick a size at random and hope.”
— Jack Ganssle, *The Art of Designing Embedded Systems*

**Contributions**
1. Prove that CompCert preserves the stack consumption of C programs by compilation.
2. Design and prove sound a Quantitative Hoare Logic that infers stack bounds on C programs.
3. Implement an automatic procedure to derive proofs in the previous logic on simple code.

**A New Quantitative CompCert**
- We propose to add call and return events to CompCert traces.
- Example trace
  
  ```
  call(f), ret(f), call(f), call(g), ret(g)
  ```
- Event metrics assign weights to program events.
- We define value and weight of program traces:
  
  
  ```
  V_M(i) = 0
  V_M(i) = |V_M(h) | | i = h | h |
  ```
- \( M \) is the event metric, it must satisfy
  
  \[
  M(\text{call}(f)) + M(\text{ret}(f)) = 0 \quad M(\text{call}(f)) > 0.
  \]

**A New Quantitative Hoare Logic**
- We define a logic inspired by Hoare logics to bound stack consumption.
- Assertions on the program state are extended to map to \( \mathbb{N} \cup \{\infty}\).
- \( \perp \) is now \( \perp \rightarrow \infty \), \( \top \) is refined by \( \perp \) is +, \( \vee \) is min, and so on.

**Hoare-like Reasoning for Stack Bounds**

\[
\{ Z = \log_2(h - l) \Rightarrow M_0 \cdot Z \} \\
\text{bsearch}(x, l, h) \{ \\
\text{if } (h-l) <= 1 \text{ return } 1; \\
\{Z>0 \land Z = \log_2(h-l) \Rightarrow M_0 \cdot Z \} \\
\lambda = \frac{h+l}{2}; \\
\{Z>0 \land Z = \log_2(h-l) \land m = \frac{h{l}}{2} \Rightarrow M_0 \cdot Z \} \\
\text{if } (a[x]=x) h = m \text{ else } l = m; \\
\{Z<1 \Rightarrow \log_2(h-l) \Rightarrow M_0 \cdot (Z-1) + M_b \} \\
\text{return bsearch}(x, l, h); \\
\{M_0 \cdot (Z-1) + M_b \} \\
\{M_0 \cdot Z \}
\]

**System Overview**

- **C Program**
- **Automatic Procedure**
- **Derivation in Quantitative Logic**
- **Soundness Proof**
- **Event Metric**

**Automatically and Manually Verified Bounds**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Manually Verified Stack Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>bsearch(x, l, h)</td>
<td>( 40(1 + \log_2(h - l)) ) bytes</td>
</tr>
<tr>
<td>fib(n)</td>
<td>( 24(n + 1) ) bytes</td>
</tr>
<tr>
<td>qsort(a, lo, hi)</td>
<td>( 48(h - l + 1) ) bytes</td>
</tr>
<tr>
<td>filter_pos(a, sz, lo, hi)</td>
<td>( 48(h - l + 1) ) bytes</td>
</tr>
<tr>
<td>sum(a, lo, hi)</td>
<td>( 32(h - l + 1) ) bytes</td>
</tr>
<tr>
<td>fact_log(n)</td>
<td>( 40 + 24n^2 ) bytes</td>
</tr>
</tbody>
</table>

**Logic Soundness**

If \( \{ P \} s(Q) \) then

\[
\forall \sigma \in t M, (s, \sigma) \rightarrow \top \text{ (skip, } \sigma) \Rightarrow W_M(t) < P(M, \sigma)
\]

The implemented version is stronger and uses postconditions.