**Proof–Producing Decompilation**

Magnus.Myreen@cl.cam.ac.uk

---

**Advertisement**

Want to prove properties of real code,

- ✔️ without trusting C compilers and C semantics?
- ✔️ without the ugliness of machine code and goto-like control flow?
- ✔️ without tying your tool to a specific machine language?

... then link your tool to my decompiler.

---

**Tool Interface**

Given some ARM, x86, or PowerPC machine code, my decompilation tool produces recursive functions describing the code:

<table>
<thead>
<tr>
<th>ARM</th>
<th>x86</th>
<th>PowerPC</th>
<th>Recursive functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>E3A00000</td>
<td>31C0</td>
<td>38A00000</td>
<td>( f(r_0, r_1, m) = ) let ( r_1 = 0 ) in ( g(r_0, r_1, m) )</td>
</tr>
<tr>
<td>E3510000</td>
<td>85F6</td>
<td>2C140000</td>
<td>( g(r_0, r_1, m) = ) if ( r_0 = 0 ) then ( (r_0, r_1, m) ) else let ( r_0 = m(r_0) ) in let ( r_1 = r_1 + 1 ) in ( g(r_0, r_1, m) )</td>
</tr>
<tr>
<td>12800001</td>
<td>7405</td>
<td>40820010</td>
<td></td>
</tr>
<tr>
<td>15911000</td>
<td>40</td>
<td>7E80A02E</td>
<td></td>
</tr>
<tr>
<td>1AFFFFFB</td>
<td>8B36</td>
<td>38A50001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EBF7</td>
<td>4BFFFF0</td>
<td></td>
</tr>
</tbody>
</table>

The translation is valid, because the tool proves in HOL4: “the code executes \( f \)”.

As a result, any property proved of the recursive functions carries over to a property of the original machine code. By linking your tool to my decompiler you can prove properties of real machine code by just proving properties of recursive functions.

---

**How the tool works**

Decompilation algorithm:

1. automatically derive separation-logic-like Hoare triples for each machine instruction based on models of ARM [Fox, TPHOLs 2003], x86 [Sarkar et al. POPL 2009] or PowerPC [Leroy, POPL 2008];

2. analyse the Hoare triples to discover: (a) the control flow, (b) the function \( f \);

3. compose together Hoare triples for the instructions and apply a special loop rule to automatically prove: “the code executes \( f \)”. No heuristics are used at this stage, which makes the tool reasonably robust.

Details in [Myreen et al. FMCAD 2008]. For more, talk to me or email me.