A RODIN PLUG-IN FOR CONSTRUCTING REUSABLE SCHEMATIC LEMMAS

P. Stankaitis, A. Iliasov, D. Adjepon-Yamoah, A. Romanovsky
May 23, 2016
Newcastle University
In this presentation we discuss:

- an approach to making proofs more generic and thus less fragile and more reusable
- a tool and verification results.
There is a number of circumstances when existing interactive proofs become invalidated:

- a part of the model is changed
- incremental changes that alter the goal, set of hypotheses, identifier names or types
There is a large number of essentially identical interactive proofs re-appearing in different projects due specific weaknesses in the underlying automatic provers.
WHY3 PLUGIN
Figure: Verification tool concept
Most of the translation effort goes into the construction and fine-tuning of Why3 support theories.
<table>
<thead>
<tr>
<th>Model</th>
<th>Proof obligations</th>
<th>open, Tactic(^1)</th>
<th>open, Tactic(^2)</th>
<th>open, Why3</th>
<th>open, Why3 (+ SL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order/Supply Communication</td>
<td>276</td>
<td>24</td>
<td>4</td>
<td>8</td>
<td>4 (+2)</td>
</tr>
<tr>
<td>Fisher’s Algorithm</td>
<td>82</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>0 (+1)</td>
</tr>
<tr>
<td>Train Control System</td>
<td>133</td>
<td>36</td>
<td>5</td>
<td>32</td>
<td>32 (+0)</td>
</tr>
<tr>
<td>B2B Communication prot.</td>
<td>498</td>
<td>63</td>
<td>25</td>
<td>20</td>
<td>8 (+5)</td>
</tr>
<tr>
<td>Automated Teller Machine</td>
<td>962</td>
<td>77</td>
<td>28</td>
<td>1</td>
<td>0 (+1)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1951</strong></td>
<td><strong>216</strong></td>
<td><strong>66</strong></td>
<td><strong>62</strong></td>
<td><strong>12</strong></td>
</tr>
</tbody>
</table>

\(^1\)Tactic - rewrite rule + nPP + PP + ML

\(^2\)Tactic - rewrite rule + nPP + PP + ML + SMT
The initial experiments have shown that a minimal axiomatisation support is not sufficient to discharge a sizeable proportion of proof obligations.
SCHEMATIC LEMMAS
A schematic lemma is a provable conjecture that does not reference any model identifiers.
⋯ † database ≜ \{a_i \mapsto a\} \in \text{Attr}_\text{id} \to \text{Attrs}
lemma lemma_total_overriding:
    forall f:rel 'a 'b, s:set 'a, t:set 'b, x: 'a, y : 'b.
    mem f (s --> t) \ mem x s \ mem y t ->
    mem (f <+ singleton (x, y)) (s --> t)

Figure: Schematic Lemma
lemma lemma_total_overriding_help0:
  forall f : rel 'a 'b, x : 'a, y : 'b.
  subset (dom f) (dom (f <+ (singleton (x, y)))))

lemma lemma_total_overriding_help1:
  forall f:rel 'a 'b, s:set 'a, t:set 'b, x: 'a, y: 'b.
  mem f (s --> t) \ mem x s \ mem y t ->
  mem (f <+ singleton (x, y)) (s --> t)

Figure: Schematic Lemmas
As we have stated previously, it has been one of the goals of this research to establish to what degree schematic lemmas are reusable at least within the same project.
The numbers show how each next lemma (L₁, L₂, ...) affects the overall number of open proof obligations.

<table>
<thead>
<tr>
<th>Model</th>
<th>open,</th>
<th>open,</th>
<th>open,</th>
<th>open,</th>
<th>open,</th>
<th>open,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Why3</td>
<td>+ L₁</td>
<td>+ L₂</td>
<td>+ L₃</td>
<td>+ L₄</td>
<td>+ L₅</td>
</tr>
<tr>
<td>B2B Communication prot.</td>
<td>20</td>
<td>16</td>
<td>14</td>
<td>12</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>
lemma lemma_natural_increment:
forall x, n : int.
mem x bnatural1 \ n >= 0 ->
mem (x + n) bnatural1

Figure: Schematic Lemma
lemma lemma_finite_partial_domain:
  forall f : rel 'a 'b, s : set 'a, t : set 'b.
  finite (dom f) \(\land\) mem f (s ++ t) →
  finite f

Figure: Schematic Lemma
There is a fine interplay between the functioning of the schematic lemmas plug-in and the Why3 plug-in filtering mechanism.
SCHEMATIC LEMMA TOOL
· It integrates into the prover perspective and offers an alternative way to conduct an interactive proof.

· The Why3 notation is employed, but the first release will support entering schematic lemma in the Event-B mathematical notation.
Figure: Schematic Lemma Plugin
The plug-in automatically constructs the first attempt at a schematic lemma through a simple syntactic transformation of a context proof obligation.
From this starting point it is up to the modeller to construct a sensible lemma by changing identifier, hypotheses and goal definitions.
CONCLUSION & FUTURE WORK
· Applying schematic lemma technique to a large number of models.

· Release the first version of the plugin.
The idea of generalisation for the purpose of proof reuse has been explored in different settings.

We also hypothesise that at a certain stage accumulated schematic lemma make automatic proof support so complete that interactive proofs are no longer necessary.
Questions?