Theory Plug-in for Rodin 3.0

T.S. Hoang\textsuperscript{1}  A. Salehi\textsuperscript{1}  M. Butler\textsuperscript{1}  L. Voisin\textsuperscript{2}

\textsuperscript{1}\textit{ECS, University of Southampton, U.K.}
\textsuperscript{2}\textit{Systerel, France}

RODIN Workshop 2016
Linz, Austria
23rd May 2015
From Rodin 2.8 to Rodin 3.0 (1/2)

Major (necessary) changes to the Rodin Core.

Stronger AST Library

- Mitigate risks of unsoundness: mixing several formula factories.
- Every AST node carries its building formula factory.
- Operation combining formulas check for factories compatibility.
Major (necessary) changes to the Rodin Core.

Stronger sequent prover

- Introduction of context-dependent reasoner.
- Context-dependent reasoner cannot be reused.
- Context-dependent reasoner has to be replayed.
- The rule-based provers’ reasoners are context-dependent.
Problems for the Theory Plug-in

- Exceptions when opening proof obligation.
- Exceptions when applying rule-based provers’ reasoners.
- Changing the model has no effects on existing proofs.
- Losing proofs when saving (the exact problem is in loading previously saved proof).
Use *ISpecialization* instead of *ISubstitution*.

Allows to specialize types consistently.

<table>
<thead>
<tr>
<th>Patterns</th>
<th>Formulae</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$P(S)$</td>
</tr>
<tr>
<td>$S$</td>
<td>$S \times T$</td>
</tr>
</tbody>
</table>
Proper implementation for matching **associative operators**.

<table>
<thead>
<tr>
<th>Patterns</th>
<th>Formulae</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f; {x \mapsto c}$</td>
<td>$g; h; {y \mapsto c}$</td>
<td>$f \leftarrow g; h$</td>
</tr>
<tr>
<td>$x \leftarrow y$</td>
<td>$c \leftarrow c$</td>
<td></td>
</tr>
</tbody>
</table>
Proper implementation for matching **associative operators**.

<table>
<thead>
<tr>
<th>Patterns</th>
<th>Formulae</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f; {x \mapsto c} )</td>
<td>( g; h; {y \mapsto c} )</td>
<td>( f \leftarrow g; h )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( x \leftarrow y )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( c \leftarrow c )</td>
</tr>
<tr>
<td>( e; f )</td>
<td>( g; h; {y \mapsto c} )</td>
<td>( e \leftarrow g )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( f \leftarrow h; {y \mapsto c} )</td>
</tr>
</tbody>
</table>
Datatype and Operator Extensions

- Correctly implement equality for datatype/operator extensions.

- Datatypes/Operators with the same definition will be assigned identical IDs.

- Formula factories can be correctly compared and upgraded.

- Saved proofs are loaded with the correct formula factories.
Theory Plug-in Version 4.0

- Major upgrade of the Theory Plug-in
- Previously saved proofs will be lost.
- The upgrade requires fixed in the Rodin Core
- Will be available after the next release of the Rodin Platform (Rodin 3.3)
Further Development

- Support for **infix predicate operators**.
- Support for **predicate variables** in theories.
- **Usability** improvement
- Improve matching facility for **associative commutative operators**
- **Tactics** for theory.
- Theory **instantiation**
Infix Predicate Operators

- **Cosmetic** changes to improve readability.
- For example, for real numbers $x_1$, $x_2$, insteads of
  \[ smr(x_1, x_2), \]
  we can write
  \[ x_1 \preceq x_2 \]
- *(No overloading of arithmetic operators).*
Predicate Variables in Theories

- Currently cannot be statically checked
- Despite the rule-based provers already have some support.
- Need some additional supports from the Rodin Core.
Interactive proofs slow in computing “applicable positions”

Compute applicable positions on demand.

Rodin Interactive proofs support needs to be changed.
Matching for AC operators

- Matching for **Associative and Commutative** operators use the same algorithm for Associative operators.

- More matching can be found if take into account **commutivity**.

**Example**

- Pattern: $x + f(y)$
- Formula: $a + f(b) + c$
- Match: $x \leftarrow a + c$, $y \leftarrow b$. 
Tactics for Theory

- Proof rules and definitions are applied in some predefined order.
- Often, users want dedicate tactics
- Simple tactic language: Sequential composition, loops (similar to the current Rodin’s preferences)
- Tactics associated with theories or with the developments?
Theory Instantiation

- Enhance reuse of theories.
- Suited for defining Abstract Data Types and their concrete representation.
- Supporting model variations through theories.
A Wish List

- Support for **infix predicate operators**.
- Support for **predicate variables** in theories.
- **Usability** improvement
- Improve matching facility for **associative commutative operators**
- **Tactics** for theory.
- Theory **instantiation**
- ...

---

16/16