1. OCL-Treffen 2003

HOL-OCL:
Embedding OCL into Isabelle/HOL

Achim D. Brucker (ETH Zürich, Switzerland)
and
Burkhart Wolff (Albert-Ludwigs Universität Freiburg, Germany)

Januar 17, 2003

Machine-Checkable Semantics

Motivation: Respect the semantical structure of the language.

- A machine-checked semantics
  - conservative embeddings guarantee consistency of the semantics.
  - builds the basis for analyzing language features.
  - allows incremental changes of semantics.

- As basis of further tool support for
  - reasoning over specifications.
  - refinement of specifications.
  - automatic test data generation.

The definition of the logical and (Kleene-logic):

\[ S \land T \equiv \lambda c. \begin{cases} \text{if } \text{DEF} (S \ c) \text{ then} & \begin{cases} \text{if } \text{DEF} (T \ c) \text{ then } \lfloor S \ c \rfloor \land \lfloor T \ c \rfloor \\ \text{else if } S \ c = (\lfloor \text{False} \rfloor) \text{ then } \lfloor \text{False} \rfloor \text{ else } \bot \\ \text{else if } T \ c = (\lfloor \text{False} \rfloor) \text{ then } \lfloor \text{False} \rfloor \text{ else } \bot \end{cases} \end{cases} \]

The truth-table can be derived from this definition.

The union of sets is defined as the strict and lifted version of \( U \):

\[ \text{union} \equiv \text{lift}_2 (\text{strictify}_N (\lambda X. \text{strictify}_N (\lambda Y. \text{Abs}_{\text{SSet}} (\lfloor \text{Rep}_{\text{SSet}} X \rfloor \cup \lfloor \text{Rep}_{\text{SSet}} Y \rfloor)))) \]

These definitions can be automatically rewritten into “Textbook-style”.
Foundations: Using Isabelle/HOL for defining semantics

Foundation:

- **Isabelle** is a generic theorem prover.
- **Higher-order logic (HOL)** is a classical logic with higher-order functions.

**HOL-OCL:** A Shallow Embedding of OCL into HOL:

- is a shallow embedding of OCL into HOL.
- provides a consistent (machine checked) OCL semantics.
- allows the examination of OCL features.
- builds the basis for OCL tool development.
- follows OCL 1.4 and the RfP for OCL 2.0
- over 2000 theorems (language properties) proven.

---

**HOL-OCL Application: Test Data Generation**

Based on a UML/OCL specification a minimal set of test data is calculated which can be used for validating an implementation.

```
context 
Triangle::isTriangle(s0,s1,s2:Integer):Boolean
pre:
(s0 > 0) and (s1 > 0) and (s2 > 0)
post:
result = 
if (isTriangle(s0,s1,s2)) then
  if (s0 = s1) then
    if (s1 = s2) then
      Equilateral::TriangType
    else
      Isosceles::TriangType endif
  else
    Isosceles::TriangType endif
else
  if (s1 = s2) then
    Isosceles::TriangType
  else
    if (s0 = s2) then
      Isosceles::TriangType
    else
      Scalene::TriangType endif
  else
    Invalid::TriangType endif
endif
```

1. Reduce all logical operation to the basis operators: and, or, und not

2. Determine disjunctive normal Form (DNF):

   \[ x \land (y \lor z) \landarrow (x \land y) \lor (x \land z) \]

3. Eliminate unsatisfiable sub-formulae, e.g.:

   scalene and invalid

4. Select test data with respect to boundary cases.

---

1. OCL-Treffen 2003
Partitioning of the Test Data

\[
\text{triangle } s_0 \ s_1 \ s_2 \Rightarrow \text{result } = \text{invalid and not isTriangle } s_0 \ s_1 \ s_2
\]
\[
\text{result } \iff \text{equilateral and isTriangle } s_0 \ s_1 \ s_2 \text{ and } s_0 \neq s_1 \text{ and } s_1 \neq s_2
\]
\[
\text{result } \iff \text{isosceles and isTriangle } s_0 \ s_1 \ s_2 \text{ and } s_0 \neq s_1 \text{ and } s_1 \neq s_2
\]
\[
\text{result } \iff \text{isosceles and isTriangle } s_0 \ s_1 \ s_2 \text{ and } s_0 \neq s_1 \text{ and } s_0 \neq s_2
\]
\[
\text{result } \iff \text{scalene and isTriangle } s_0 \ s_1 \ s_2 \text{ and } s_0 \neq s_1 \text{ and } s_0 \neq s_2 \text{ and } s_1 \neq s_2
\]

1. Input describes no triangle.
2. Input describes an equilateral triangle.
3. Input describes an isosceles triangle:
   (a) with \( s_0 \) equals \( s_1 \).
   (b) with \( s_0 \) equals \( s_2 \).
   (c) with \( s_1 \) equals \( s_2 \).
4. Input describes an scalene triangle.

For each partition, concrete test data has to be selected with respect to boundary cases (e.g. max./min. Integers, ...).

Conclusion

A theorem prover based OCL definition of the OCL semantics:

- provides a sound and consistent semantic “Textbook”.
- allows the definition of a proof calculi over OCL.
- Gives OCL/UML the power of well-known Formal Methods (e.g. Z, VDM), e.g. for:
  - validation...
  - verification.
  - Refinement.
  - automated test data generation.
  - ...

<table>
<thead>
<tr>
<th>extendible universes</th>
<th>OCL 1.4</th>
<th>OCL 2.0 RfP</th>
<th>HOL-OCL preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>general recursion</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>smashing</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>automated flattening</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>tuples</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>finite state</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>general Quantifiers</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>allInstances finite</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Kleene logic</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>strong and weak equality</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>