Featherweight OCL
A study for the consistent semantics of OCL 2.3 in HOL

Achim D. Brucker$^1$  Burkhart Wolff$^2$

$^1$SAP AG, SAP Research, Germany
achim.brucker@sap.com

$^2$Université Paris-Sud, France
wolff@lri.fr

September 30, 2012
Outline

1. Motivation
2. Featherweight OCL
3. Conclusion and Further Work
Outline

1. Motivation
2. Featherweight OCL
3. Conclusion and Further Work
The semantics of OCL 2.3 is spread over several places:

**Chapter 7 “OCL Language Description” (informative):** introduces OCL informally using examples,

**Chapter 10 “Semantics Described using UML” (normative):** presents an “evaluation” environment,

**Chapter 11 “The OCL Standard Library” (normative):** describes the requirements (pre-/post-style) of the library,

**Appendix A “Semantics” (informative):** presents a *formal semantics* (textbook style), based on the work of Richters.

And all that needs to be aligned with all other UML (sub-)standards.
History: A Singe Undefined Value (invalid)

- OCL was equipped with a single exception element: 
  invalid (previously called oclUndefined)

- invalid is used to model all exceptional situations
  - division by zero, e.g., \( 1/0 \)
  - accessing elements of a empty list, e.g., \( \text{Seq}{}\rightarrow\text{first}() \)
  - representation of “absence of a value”
  - ...

- Most operations are strict, e.g.,
  
  \[
  \text{self.x} \rightarrow \text{including} (\text{invalid}) = \text{invalid}
  \]

- Exception: Boolean operations, e.g.,
  
  \[
  \text{invalid or true} = \text{true}
  \]
Adding a New “Undefinedness”

Motivation and Intuition

■ **Main Motivation:**
  Alignment with the UML standard.

■ **Action Taken by OMG:**
  Introduction of a second exception element: `null`.

■ **Intuition:**
  ■ `null` represents *absence of value*.
  ■ `null` is a potentially *non-strict* exception element.
Adding a New “Undefinedness”

Observation

In OCL 2.2, his extension has been done in an ad hoc manner, e.g.,

- Both `invalid` and `null` conform to all classifiers.
- In particular `null` conforms to `invalid` and vice versa.
- The conforms relationship is antisymmetric, thus `invalid` and `null` are indistinguishable.
- Contradiction to: `null` being non-strict and `invalid` being strict.

Our Contribution:

- At the OCL Workshop 2009, we presented a “paper and pencil” integration of `null` into the semantics of OCL 2.0
- Featherweight OCL formalizes this semantics in Isabelle/HOL (following the tradition of HOL-OCL)
Outline

1 Motivation

2 Featherweight OCL

3 Conclusion and Further Work
Featherweight OCL
Formalizing the Core of OCL

- Embedding into Isabelle/HOL
- Shallow embedding
- Strongly typed
- Any Featherweight OCL type contains at least invalid and null
- All objects are represented in an object universe
- Featherweight OCL types may be arbitrarily nested
- Support for infinite sets
- Support for equational reasoning and congruence reasoning
Example: Addition of integers

The interpretation of "X+Y" for Integers:

\[ I[X + Y]_\tau \equiv \begin{cases} 
I[X]_\tau + I[Y]_\tau & \text{if } I[X]_\tau \neq \bot \\
\bot & \text{and } I[Y]_\tau \neq \bot, \\
\bot & \text{otherwise.}
\end{cases} \]

This is a \textbf{strict} version of the addition of Integers.
We define

\[ I[X + Y] \tau \equiv \begin{cases} 
\downarrow x \downarrow + \downarrow y \downarrow & \text{if } x \neq \bot, y \neq \bot, \uparrow x \neq \bot \\
\downarrow & \text{and } \uparrow y \neq \bot \\
\bot & \text{otherwise}
\end{cases} \]

where \( x = I[X] \tau \) and \( y = I[Y] \tau \).

\((x \neq \bot \iff \text{“x is not invalid”} \text{ and } x \neq \bot \iff \text{“x is not null”})\)

Note: \( 3 + \text{null}_{\text{Integer}} = \text{invalid} \)
The interpretation of “X and Y” for Booleans:

\[
I[X \text{ and } Y]_\tau \equiv \begin{cases} 
\llbracket x \land y \rrbracket & \text{if } x \neq \bot \text{ and } y \neq \bot, \\
\text{false} & \text{if } x = \llbracket \text{false} \rrbracket \text{ or } y = \llbracket \text{false} \rrbracket, \\
\bot & \text{otherwise}.
\end{cases}
\]

where \( x = I[X]_\tau \) and \( y = I[Y]_\tau \).

The OCL standard demands a Strong Kleene Logic.
OCL 2.3: Challenges in the Standard

- The standard defines
  
  \[
  \text{not (null)} = \text{invalid}
  \]

- With the consequence, that
  
  \[
  \text{not (not } X) = X
  \]

  does not hold for all values of \( X \):

  \[
  \text{not (not null)} = \text{invalid}
  \]

- Similarly:
  
  \[
  \text{null and null} = \text{invalid}
  \]
We recommend:\(^1\)

\[
I[X \text{ and } Y]_{\tau} \equiv \begin{cases} 
\neg x \land \neg y & \text{if } x \neq \bot \text{ and } y \neq \bot \\
\text{false} & \text{or } \neg x \neq \bot \text{ and } \neg y \neq \bot, \\
\bot & \text{if } x = \bot \text{ or } y = \bot, \\
\text{false} & \text{or } x = \text{true} \text{ and } y = \bot, \\
\bot & \text{or } x = \bot \text{ and } y = \text{true}, \\
\bot & \text{otherwise.}
\end{cases}
\]

where \( x = I[X]_{\tau} \) and \( y = I[Y]_{\tau} \).

Note: \( \bot \) represents \text{null} and \( \bot \) represents \text{invalid}.

This definition deviates from the current OCL 2.3.1 standard.

\(^1\)modified for simplifying the presentation
We formally prove the following core properties of "and":

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(invalid and true)</td>
<td>= invalid</td>
<td>(false and true)</td>
<td>= false</td>
</tr>
<tr>
<td>(invalid and false)</td>
<td>= false</td>
<td>(false and false)</td>
<td>= false</td>
</tr>
<tr>
<td>(invalid and null)</td>
<td>= invalid</td>
<td>(false and null)</td>
<td>= false</td>
</tr>
<tr>
<td>(invalid and invalid)</td>
<td>= invalid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(null and true)</td>
<td>= null</td>
<td>(true and true)</td>
<td>= true</td>
</tr>
<tr>
<td>(null and false)</td>
<td>= false</td>
<td>(true and false)</td>
<td>= false</td>
</tr>
<tr>
<td>(null and null)</td>
<td>= null</td>
<td>(true and null)</td>
<td>= null</td>
</tr>
<tr>
<td>(null and invalid)</td>
<td>= invalid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(false and true)</td>
<td>= false</td>
<td>(false and false)</td>
<td>= false</td>
</tr>
<tr>
<td>(false and null)</td>
<td>= false</td>
<td>(false and null)</td>
<td>= false</td>
</tr>
<tr>
<td>(false and invalid)</td>
<td>= false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As well as:

(X and X) = X
X and true = X
X and false = false

(X and Y) = (Y and X)
(X and (Y and Z)) = (X and Y and Z)
Demo

**Boolean Operations (Non-strict Operations)**

---

*Note that @term "not" is defined to mean that we need a definition of @text "not(not(x))=x". *

**Lemma 1:** $(\neg X) \Rightarrow (\neg (\lambda x. X) x)$

by (simp add: not_def)

**Lemma 2:** $\neg invalid = invalid$

by (rule ext, simp add: not_def null_def invalid_def)

---

```isabelle
val it = (): unit
val commit = fn: unit -> bool
val it = (): unit
ML>
Welcome to Isabelle/HOL (Isabelle2011-1: October 2011)
process ready
```
Outline

1 Motivation
2 Featherweight OCL
3 Conclusion and Further Work
Conclusions

We understand OCL as a specification language

- Should be more abstract than a programming language
- The usual algebraic laws should hold
- Four-valued Kleene-Logic (lattice like organization of values)

Formalizing the core of OCL

- Helps to clarify the semantics
- Helps to preserve consistency while extending the language
- Can provide input for updating "Annex A"

Many new interesting extensions are discussed, e.g.,

- $\lambda$-expression
- ...
Personal Opinion

Status of the standard

- OCL 2.2 was a total mess with respect to null
- OCL 2.3 is an improvement, still many glitches

The OMG standardization process where members vote on changes
- is maybe not best process to achieve a consistent standard

Technical standards should use authoring systems that ensure
- the syntactical correctness
- semantical consistency
Thank you for your attention!

Any questions or remarks?
Related Publications

Extending OCL with null-references.
Selected best papers from all satellite events of the MoDELS 2009 conference.

Achim D. Brucker and Burkhart Wolff.
Featherweight OCL: A study for the consistent semantics of OCL 2.3 in HOL.
In *Workshop on OCL and Textual Modelling (OCL 2012)*. 2012.