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

Achim D. Brucker\textsuperscript{1}  Burkhart Wolff\textsuperscript{2}

\textsuperscript{1}SAP AG, SAP Research, Germany
achim.brucker@sap.com

\textsuperscript{2}Université Paris-Sud, France
wolff@lri.fr

September 30, 2012

Outline

1 Motivation
2 Featherweight OCL
3 Conclusion and Further Work

Motivation

Semantics in the OCL 2.3 Standard

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
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.

Outline

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

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
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

OCL 2.0: Strict Operations

Example: Addition of integers
The interpretation of "X+Y" for Integers:
\[ I[X + Y] \tau \equiv \begin{cases} 
\lceil I[X] \tau \rceil + \lceil I[Y] \tau \rceil & \text{if } I[X] \tau \not= \bot \text{ and } I[Y] \tau \not= \bot, \\
\bot & \text{otherwise}.
\end{cases} \]

This is a strict version of the addition of Integers.

OCL 2.3: Strict Operations and Null

We define
\[ I[X + Y] \tau \equiv \begin{cases} 
\lceil x \rceil \wedge \lceil y \rceil & \text{if } x \not= \bot, y \not= \bot, \lceil x \rceil \not= \bot \text{ and } \lceil y \rceil \not= \bot, \\
\bot & \text{otherwise}
\end{cases} \]

where \( x = I[X] \tau \) and \( y = I[Y] \tau \).
\( x \not= \bot \iff \text{"x is not invalid"} \) and \( x \not= \bot \iff \text{"x is not null"} \)

Note: \( 3 + \text{nullInteger} = \text{invalid} \)

OCL 2.0: Boolean Operations (Non-strict Operations)

The interpretation of "X and Y" for Booleans:
\[ I[X \wedge Y] \tau \equiv \begin{cases} 
\lceil x \rceil \wedge \lceil y \rceil & \text{if } x \not= \bot \text{ and } y \not= \bot, \\
\text{false} & \text{if } x = \text{false} \text{ or } y = \text{false}, \\
\bot & \text{otherwise}.
\end{cases} \]

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

The OCL standard demands a Strong Kleene Logic.
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:\(^1\)

\[
I[x \land y]_\tau = \begin{cases} 
\text{null} & \text{if } x = \null_\tau \text{ or } y = \null_\tau, \\
\text{false} & \text{if } x = \false_\tau \text{ and } y = \false_\tau, \\
\text{true} & \text{if } x = \true_\tau \text{ and } y = \true_\tau, \\
\bot & \text{otherwise.}
\end{cases}
\]

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

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

This definition deviates from the current OCL 2.3.1 standard.

^1modified for simplifying the presentation

We formally prove the following core properties of “and”:

\[
\begin{align*}
\text{(invalid and true)} &= \text{invalid} \\
\text{(invalid and false)} &= \text{false} \\
\text{(invalid and null)} &= \text{invalid} \\
\text{(invalid and invalid)} &= \text{invalid} \\
\text{(null and true)} &= \text{null} \\
\text{(null and false)} &= \text{false} \\
\text{(null and null)} &= \text{null} \\
\text{(null and invalid)} &= \text{invalid} \\
\end{align*}
\]

As well as:

\[
\begin{align*}
\text{(false and true)} &= \text{false} \\
\text{(false and false)} &= \text{false} \\
\text{(false and null)} &= \text{false} \\
\text{(false and invalid)} &= \text{false} \\
\text{(true and true)} &= \text{true} \\
\text{(true and false)} &= \text{false} \\
\text{(true and null)} &= \text{null} \\
\text{(true and invalid)} &= \text{null} \\
\end{align*}
\]

\[
\begin{align*}
\text{(X and Y)} &= (Y \land X) \\
\text{(X and (Y and Z))} &= (X \land Y \land Z)
\end{align*}
\]
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. 