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

Outline

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

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., `Seq{}->first()`
  - representation of "absence of a value"
  - ...
- Most operations are *strict*, e. g.,

```
self.x->including(invalid) = invalid
```

- Exception: Boolean operations, e. g.,

```
invalid or true = 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

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

OCL 2.0: Strict Operations

- Example: Addition of integers
- The interpretation of “X+Y” for Integers:

$$I\llbracket X + Y\rrbracket\,\tau \equiv \begin{cases} \llcorner \ulcorner I\llbracket X\rrbracket\,\tau \urcorner + \ulcorner I\llbracket Y\rrbracket\,\tau \urcorner \lrcorner & \text{if } I\llbracket X\rrbracket\,\tau \neq \bot \\ & \quad \text{and } I\llbracket Y\rrbracket\,\tau \neq \bot, \\ \bot & \text{otherwise}\,. \end{cases}$$

- This is a **strict** version of the addition of Integers.

OCL 2.3: Strict Operations and Null

- We define

$$I\llbracket X + Y\rrbracket\,\tau \equiv \begin{cases} \llcorner\llcorner \ulcorner\ulcorner x \urcorner\urcorner + \ulcorner\ulcorner y \urcorner\urcorner \lrcorner\lrcorner & \text{if } x \neq \bot, y \neq \bot, \ulcorner x \urcorner \neq \bot \\ & \quad \text{and } \ulcorner y \urcorner \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

  where $x = I\llbracket X\rrbracket\,\tau$ and $y = I\llbracket Y\rrbracket\,\tau$.
  ($x \neq \bot \iff$ “x is not invalid” and $x \neq \bot \iff$ “x is not null”)
- Note: $3 + \texttt{null}_{\texttt{Integer}} = \texttt{invalid}$

OCL 2.0: Boolean Operations (Non-strict Operations)

- The interpretation of “X and Y” for Booleans:

$$I\llbracket X \text{ and } Y\rrbracket\tau \equiv \begin{cases} \llcorner \ulcorner x \urcorner \land \ulcorner y \urcorner \lrcorner & \text{if } x \neq \bot \text{ and } y \neq \bot, \\ \llcorner \text{false} \lrcorner & \text{if } x = \llcorner \text{false} \lrcorner \text{ or } y = \llcorner \text{false} \lrcorner, \\ \bot & \text{otherwise}\,. \end{cases}$$

  where $x = I\llbracket X\rrbracket\tau$ and $y = I\llbracket Y\rrbracket\tau$.
- The OCL standard demands a Strong Kleene Logic.

OCL 2.3: Challenges in the Standard

- The standard defines

```
not (null) = invalid
```

- With the consequence, that

```
not (not X) = X
```

does not hold for all values of X:

```
not (not null) = invalid
```

- Similarly:

```
null and null = invalid
```


OCL 2.3: Boolean Operations (Non-strict Operations)

- We recommend:[1]

$$I[\![X \text{ and } Y]\!]\tau \equiv \begin{cases} \llcorner\ulcorner x \urcorner \wedge \ulcorner y \urcorner\lrcorner & \text{if } x \neq \bot \text{ and } y \neq \bot \\ & \quad \text{or } \ulcorner x \urcorner \neq \bot \text{ and } \ulcorner y \urcorner \neq \bot, \\ \llcorner \text{false} \lrcorner & \text{if } x = \llcorner \text{false} \lrcorner \text{ or } y = \llcorner \text{false} \lrcorner, \\ \llcorner \bot \lrcorner & \text{if } x = \llcorner \bot \lrcorner \text{ and } y = \llcorner \bot \lrcorner \\ & \quad \text{or } x = \llcorner \text{true} \lrcorner \text{ and } y = \llcorner \bot \lrcorner \\ & \quad \text{or } x = \llcorner \bot \lrcorner \text{ and } y = \llcorner \text{true} \lrcorner, \\ \bot & \text{otherwise}. \end{cases}$$

where $x = I[\![X]\!]\tau$ and $y = I[\![Y]\!]\tau$.
Note: $\llcorner \bot \lrcorner$ *represents* `null` and $\bot$ represents `invalid`.

- This definition deviates from the current OCL 2.3.1 standard.

[1] modified for simplifying the presentation

OCL 2.3: The Boolean Operations “and”

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

```
(invalid and true)    = invalid
(invalid and false)   = false
(invalid and null)    = invalid
(invalid and invalid) = invalid
```

```
(false and true)      = false
(false and false)     = false
(false and null)      = false
(false and invalid)   = false
```

```
(null and true)       = null
(null and false)      = false
(null and null)       = null
(null and invalid)    = invalid
```

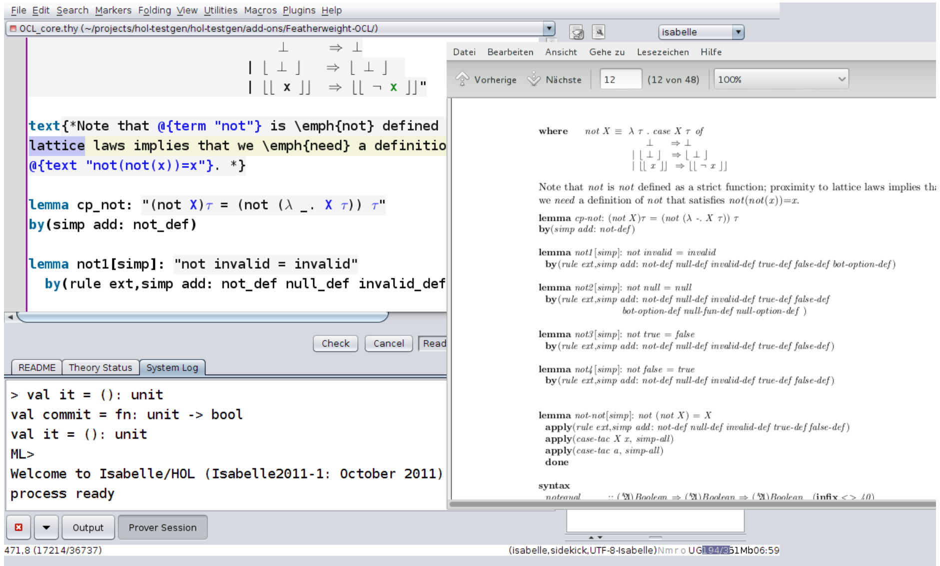
```
(true and true)       = true
(true and false)      = false
(true and null)       = null
(true and invalid)    = invalid
```

- 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



Outline

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.,

- λ-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 remaks?

Related Publications

Achim D. Brucker, Matthias P. Krieger, and Burkhart Wolff.
Extending OCL with null-references.
In Sudipto Gosh, editor, *Models in Software Engineering*, number 6002 in LNCS, pages 261–275. Springer, 2009.
`http://www.brucker.ch/bibliography/abstract/brucker.ea-ocl-null-2009`.
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.
`http://www.brucker.ch/bibliography/abstract/brucker.ea-featherweight-2012`.