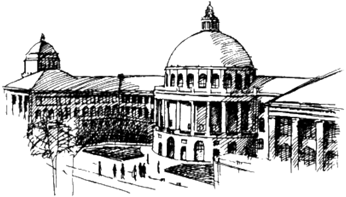
Semantic Issues of OCL: Past, Present, and Future

Achim D. Brucker
joint work with
Jürgen Doser, and Burkhart Wolff



Information Security, ETH Zurich, Switzerland

OCL for (Meta-) Models in Multiple Application Domains
October 2, 2006

Semantics in the OCL 2.0 Standard

The semantics of OCL 2.0 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.

Defining Semantics

Formal OCL Semantics

Textbook Semantics

- good to communicate
- no calculi

Machine Checkable Semantics

Language Research

- Language Analysis
- Language Consistency

Applications

- Verification
- Refinement
- Specification Consistency

Analyze Structure of the Semantics,
Basis for Tools, Reuseability

Textbook Semantics:

- The Interpretation of "`X->union(Y)`" for sets ("$X \cup Y$"):

$$I(\cup)(X, Y) \equiv \begin{cases} X \cup Y & \text{if } X \neq \perp \text{ and } Y \neq \perp, \\ \perp & \text{otherwise.} \end{cases}$$

- This is a strict and lifted version of the union of "mathematical sets".

Machine-checkable Semantics:

- The Interpretation of "`X->union(Y)`" for sets ("$X \cup Y$"):

$$\_\cup\_ \equiv \mathrm{lift}_2\Big(\mathrm{strictify}\big(\lambda\, X.\ \mathrm{strictify}(\lambda\, Y.\ \llcorner\ulcorner X\urcorner \cup \ulcorner Y\urcorner\lrcorner)\big)\Big).$$

- We make concept like "strict" and "lifted" explicit.
- Many theorems, like

$$A \cup B = B \cup A$$

can be automatically lifted based on their HOL variants.

Proving Requirements

isEmpty() : Boolean (11.7.1-g)

Is self the empty collection?

```
post: result = ( self->size() = 0 )
```

Bag

$$\textit{\textbf{lemma}}\ (\varnothing \doteq \mathrm{self}) = \|(\mathrm{self}, \beta :: \mathrm{bot})\mathrm{Bag})\| \doteq 0$$

apply(rule Bag_sem_cases_ext, simp_all)

apply(simp_all add: OCL_Bag.OclSize_def OclMtBag_def OclStrictEq_def Zero_ocl_int_def ss_lifting')

done

List of Glitches

- We found several glitches:
  - inconsistencies between the formal semantics and the requirements
  - missing pre- and postconditions
  - wrong (e.g., to weak) pre- and postconditions
  - ...
- and examined possible extensions (open problems):
  - operations calls and invocations
  - smashing of datatypes
  - equalities
  - recursion
  - semantics for invariants (type sets)
  - ...

Conclusion

A machine-checked formal semantics should be a "first class" citizen of the next OCL standard.

- UML/OCL could be used for accredited certification processes like EAL7 or Common Criteria,
- this would open the door for a wide range of semi-formal and formal tools.
- whereas formalizing to early, can kill the standardization process, for OCL the time is ripe.

HOL-OCL

- a formal, machine-checked semantics for OCL 2.0,
- an interactive proof environment for OCL,
- servers as a basis for examining extensions of OCL,
- publicly available:
  `http://www.brucker.ch/projects/hol-ocl/`.

Bibliography

The HOL-OCL website, Mar. 2006.
`http://www.brucker.ch/research/hol-ocl/`.

UML 2.0 OCL specification, Oct. 2003.
`ptc/2003-10-14`.

M. Richters.
A Precise Approach to Validating UML Models and OCL Constraints.
PhD thesis, Universität Bremen, 2002.

Appendix

Outline

Motivation

Background

Past:

Present:

Why OCL is Important

OCL can be a success story.

- UML/OCL attracts the applied OO community:
  - is defined by the OO community,
  - has a "programming language face,"
  - increasing tool support.
- UML/OCL is attractive to researchers:
  - defines a "core language" for object-oriented modelling,
  - provides good target for OO semantics research,
  - offers the chance for bringing formal methods closer to industry.

This motivates our interests in formal tools support for OCL.

Shallow vs. Deep Embeddings

Representing the logical operations *or* and *and* via a

- shallow embedding:

  Direct definition of the semantics, e.g. each construct is represented by some function on a semantic domain.

  $$x \text{ and } y \equiv \lambda\, e.\; x\; e \wedge y\; e \qquad x \text{ or } y \equiv \lambda\, e.\; x\; e \vee y\; e$$

- deep embedding:

  The abstract syntax is presented as a datatype and a semantic function I from syntax to semantics.

  $$expr = \text{var } var \mid expr \text{ and } expr \mid expr \text{ or } expr$$

  and the explicit semantic function I:

$$\begin{aligned} I[\![\text{var } x]\!] &= \lambda\, e\,.\, e(x) \\ I[\![x \text{and} y]\!] &= \lambda\, e\,.\, I[\![x]\!]\; e \wedge I[\![y]\!]\; e \\ I[\![x \text{or} y]\!] &= \lambda\, e\,.\, I[\![x]\!]\; e \vee I[\![y]\!]\; e \end{aligned}$$

Machine-Checkable Semantics

Motivation: Honor 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.

Evolving Standards

- OCL introduced to complete the UML 1.1 standard.
- research helped to improve the standard
- The work of Richters [3] provides a formal semantics for OCL

The Semantics Foundation of the Standard

We see the formal foundation of OCL critical:

- no normative formal semantics.
- no consistency and completeness check.
- no proof that the formal semantics satisfies the normative requirements.

Nevertheless, we think the OCL standard (“`ptc/03-10-14`”) is mature enough to serve as a basis for a machine-checked semantics and formal tools support.