

A Benchmark for OCL Engine Accuracy, Determinateness, and Efficiency

Martin Gogolla, **Mirco Kuhlmann**, Fabian Büttner

in cooperation with

Benjamin Büttelmann, Lars Harmann, Friederike Jolk, Bin Sun, Hui Wang, Lei Xia

Database Systems Group
University of Bremen

October 3rd, 2008

Motivation for an OCL Benchmark

Object Constraint Language

- querying system states, formulating textual constraints
- determining model properties, checking applicability of transformations (MDE)
- ImperativeOCL
- temporal OCL, real-time OCL
- many commercial and open source tools

Requirements for practical Application

- correct and complete realization of OCL: conformance to the OCL standard
- enabling heterogenous tool chains (consistent implementations)

Problems

Expression:

```
Set{1,2,3}->collect( i | Seq{i, i*i} )
```

Results:

Tool A: Bag{Seq{1,1}, Seq{2,4}, Seq{3,9}}

Tool B: Seq{Seq{1,1}, Seq{2,4}, Seq{3,9}}

Tool C: Bag{1,1,2,4,3,9}

Expressions and Evaluation:

$set \rightarrow one(e \mid e) \equiv true$

$set = expr \equiv true$

$expr \rightarrow one(e \mid e) \equiv false$

OCL Benchmark

Approach

- develop an OCL benchmark
- consider accuracy, determinateness and efficiency
- apply the benchmark
- point out frequent shortcomings and serious flaws
- encourage tool developers to use the benchmark

Considered Tools

- ATL OCL
- Dresden OCL
- Eclipse MDT OCL
- OCLE
- Octopus
- RocIET
- USE

Further Tools

- MagicDraw
- HOL-OCL
- ITP/OCL
- ...

Structure of the OCL Benchmark

Accuracy	B1	Core (data types, invariants, properties, binary associations)
	B2	Extended core (enumerations, pre- and postconditions, queries)
	B3	Advanced modeling (ternary associations, association classes)
	B4	Three-valued logic (e.g. <code>1/0=1</code> or <code>true</code>)
	B5	OCL laws (e.g. <code>select</code> versus <code>reject</code>)
Determinateness	B6	OCL features with non-deterministic flair (e.g. <code>any</code> , <code>flatten</code>)
Efficiency	B7	Evaluation for data type, user-defined and collection operations

OCL Engine Implementation Accuracy (Problem)

- 1 parser of tool **A** does not accept OCL expressions created with tool **B**
- 2 tool **C** accepts the syntax of tool **B**, but shows different evaluation results

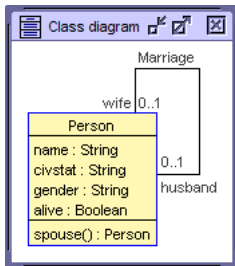
OCL Engine Implementation Accuracy (Measurement)

Components of the Accuracy Benchmark

- core
- extended core
- advanced
- three-valued logic
- OCL laws

OCL Engine Implementation Accuracy (Measurement)

core | extended core | advanced | three-valued logic | OCL laws

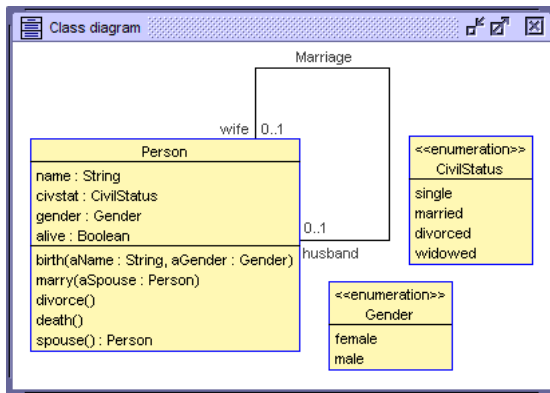


Examples

- `Person.allInstances().name`
- `Person.allInstances->collect(name)`
- `col->reject(gender='male')`
- `col->reject(p|p.gender='male')`
- `col->reject(p:Person|p.gender='male')`

OCL Engine Implementation Accuracy (Measurement)

core | **extended core** | advanced | three-valued logic | OCL laws

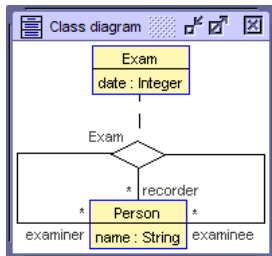


Extended Features

- enumerations
- side-effected operations (pre- and postconditions)
- state-dependent queries

OCL Engine Implementation Accuracy (Measurement)

core | extended core | **advanced** | three-valued logic | OCL laws



Advanced Features

- ternary reflexive association class
- roles

Examples

- `aPerson.examiner [recorder]`
- `aPerson.examiner [examinee]`

OCL Engine Implementation Accuracy (Measurement)

core | extended core | advanced | **three-valued logic** | **OCL laws**

Examples for logical Expressions

- `true or undefExpr`
- `undefExpr or true`

Examples for OCL laws Expressions

- `col->exists(i | e) = col->select(i | e) ->notEmpty()`
- `col->exists(i | e) =
col->iterate(i; r:Boolean=false | r or e)`

OCL Engine Determinateness Properties

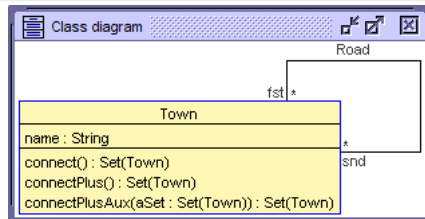
Non-deterministic OCL features

- e.g. **flatten**, **asSequence** and **any**
- no strict specification (**Set**{*a*, *b*}->**any**(**true**) result?)
- *set*->**asSequence**() = **Sequence**{*set*}->**flatten**()
- *set*->**any**(**true**) = *set*->**asBag**()->**any**(**true**)

OCL Engine Efficiency

OCL standard data types

```
Sequence{1..2048}->iterate(  
  i:Integer; res:Sequence(Integer)=Sequence{} |  
  if m.isPrime(i) then res->including(i)  
  else res endif)
```



User-defined Model

```
Set{1..1024*1024}->iterate(i:Integer;  
  res:Bag(Set(Town))=Town.allInstances->  
  collect(t|t.connectPlus()) | res)
```

Evaluation Results (1)

Empirical Data

- **949** test cases
 - ▶ **71** invariants, **878** query expressions
 - ▶ **121** state-dependent, **828** context-free
- **7** OCL evaluation engines tested (including two code generators)
- **401** of the accuracy and determinateness test cases (**46,8%**) correctly evaluated by all tools (every other test case erroneously evaluated by at least one tool)

Evaluation Results (2)

Typical Inconsistencies and Shortcomings

- typing of `let` variables
- range expressions (`Set{1..9}`)
- `'hello' -> substring(1,1)='e'` vs.
`'hello' -> substring(1,1)='h'`
- iterator variables (`c->iterate(x,y|...)`)
- enumeration values
- undefined value (`Set{undefValue}->includes(undefValue), undefValue or true`)
- binding of boolean operations

Conclusion and Future Work

- meaningful results
 - ▶ different interpretations of the OCL standard
 - ▶ faulty implementations of OCL features
- quality measure in OCL engine development
- supply functionality for carrying out the benchmark
- extend the benchmark (not all features covered yet: qualifiers, ordered sets, etc.)

Thank you for your attention!