Formal Executable Semantics for Conformance in the MDE Framework

Vlad Rusu, Inria Rennes & Lille
common work (for ISSE version) with
Marina Egea (ETH Zurich)

Formal Executable Semantics?

- Semantics = meaning

- Formal semantics
  - little or no ambiguity, inconsistency, redundancy
  - but requires some theory

- Formal Executable semantics
  - directly understandable/executable by formal software tool
  - no gap between *definition* on paper and *implementation.*

Conformance in the MDE Framework

Level 3 : meta-meta-models

MOF

Level 2 : meta-models

UML metamodel

OCL metamodel

Conformance

Level 1 : models

UML

OCL

Level 0 : programs

Contents

- Background on equational logic and Maude
- Models, meta-models, and conformance
- Representation & Semantics in Maude
- Related & Future Work & Conclusion.

Example of Specification

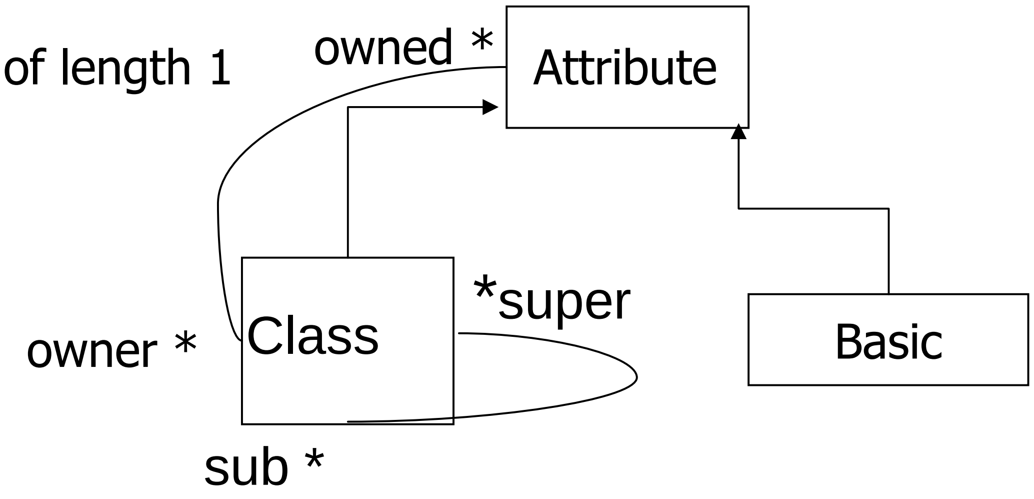
```
spec NAT is
sorts Nat NzNat .
subsort NzNat < Nat .
op 0 : -> Nat .
op s : Nat -> NzNat .
op _+_ : Nat Nat -> Nat .
vars n m : Nat .
eq 0 + n = n .
eq s(n) + m = s(n + m) .
```

What is Maude ?

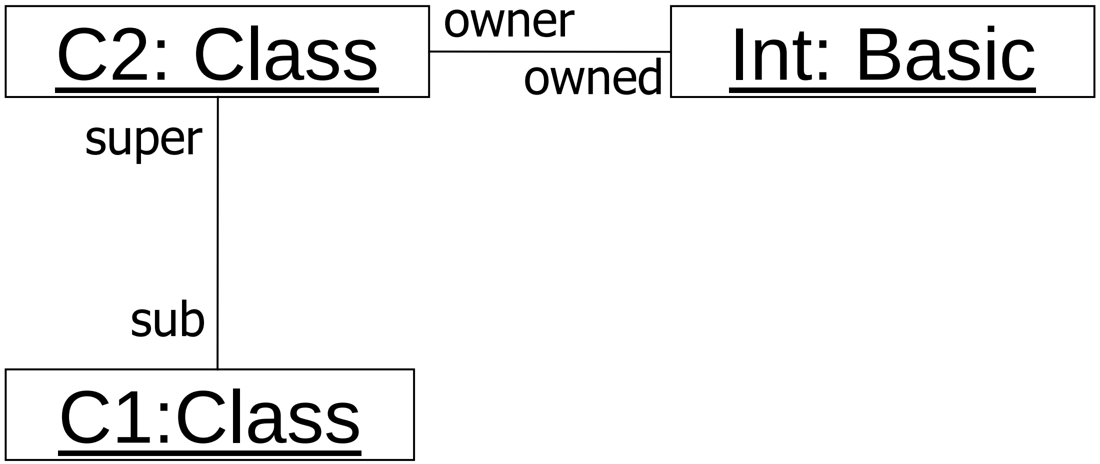
- *Implementation*: *Membership* eq. logic
- A *programming language*: functional style
- A *set of tools* for analysing specifications
    - theorem prover (partially automatic)
    - many others.

Example : Meta Model+OCL

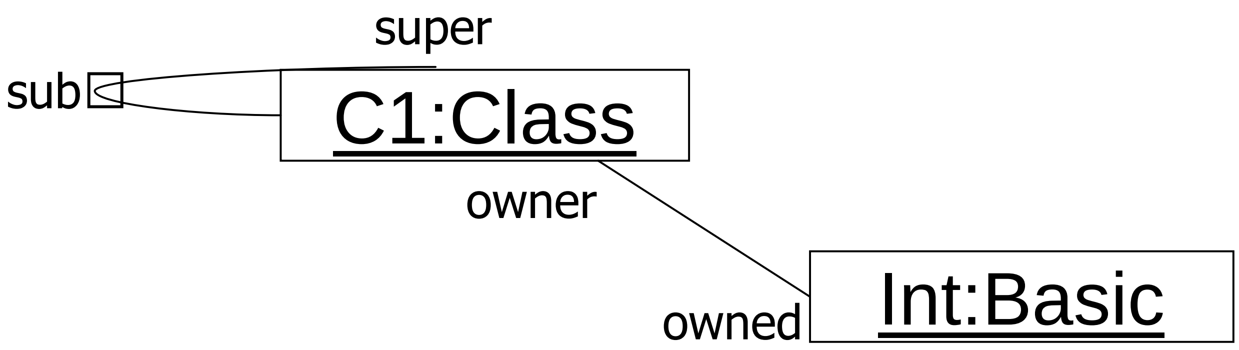
OCL invariant: no cycles of length 1
Class.allInstances ->
forAll (x:Class|
x.super ->excludes(x))



A Conformant Model



A non-Conformant Model



The Meta-Model in Maude (simplified - not the real one)

```
spec Maude(MM) is

sorts Class Attribute Basic . --- the concepts (classes)

subsorts Class  Basic < Attribute . --- subsorts for inheritance

op _super-sub_ : Class Class -> Bool . ---  association

op _owner-owned_ : Class Attribute -> Bool .   ---  association

var x : Class .

eq x super-sub x = false .  --- the OCL invariant
```

The Correct Model in Maude

meta-model

```
spec Maude(M of MM) is --- M is associated with meta-model MM ☐
sorts Class Attribute Basic . --- the concepts
subsorts Class  Basic < Attribute  . --- subsorts for inheritance
op _super-sub_ : Class Class -> Bool . ---  relation
op _owner-owned_ : Class Attribute -> Bool .   ---  relation
var x : Class .
eq x super-sub x = false .  --- the OCL invariant

ops c1 c2 : -> Class--- !!! constants for actual classes
eq c2 super-sub c1 = true .
eq c1 super-sub c1 = false .
…
op Int : -> Basic .
eq c2 owner-owned Int = true .
eq  c1 owner-owned Int  = false .
…
```

M satisfies OCL invariants of MM iff
Maude(M of MM) *logically consistent*

M structurally conforms to MM iff
Maude(M of MM) syntaxically correct

(Very Closely)□ Related Work

- **Clavel & Egea**: correctness of object w.r.t. class diagrams
  - Our +: semantics in terms of *theory interpretations*

- **Boronat & Meseguer** :
  - existing tool (MOMENT2), model transformations
  - different representation (directly : models as terms, meta-models as sorts, complex structure)
  - Our +: *proved correct* conformance checking.

Conclusion & Future Work

- For conformance: automatic verification
  - TBD: “real” case studies
- For *model transformations*: model checking & simulation checking
  - TBD: graphical language, case studies.

Equational Logic: Syntax

A *Specification* consists of

- *sorts* (a.k.a. types), e.g., Bool, Nat…
- *functions* beween sorts
- *equations* defining functions.

Formal Semantics using Algebras

- Models: Initial algebra [Maude(M of MM)]
- Meta-models [[Maude(MM)]]={[Maude(M of MM)] | Maude(M of MM) logically consistent}
- Conformance [Maude(M of MM)]∈[[Maude(MM)]]

There Are Many Algebras !!!

A non-standard interpretation *(too small)*

- *Booleans* for Nat, NzNat
- *false* for zero, *true* for s(x) for all x
- *logical OR* for +.

- There is one *Initial* algebra
  - Sorts = smallest sets satisfying equations
  - Equality = smallest congruence satisfying equations (congruence = equivalence closed on context)

Equational Logic : Semantics

An *algebra* for a specification consists of

- a *set* for each *sort*, compatible with subsorting
- a *function* (resp. *constant*) for each *function symbol* (resp. *constant symbol*)

such that all *equations are satisfied.*