Formal Reasoning on OCL Constraints

Matthias P. Krieger and Burkhart Wolff

Université Paris-Sud

10 Years Dresden OCL — Quo Vadis?

Dresden, October 15th, 2009

Advances in Automated Theorem Proving and Constraint Solving

- Bugs can be detected automatically in huge chips
- Enormous systems of equations can be solved for optimal planning
- Mathematical theorems proved with computer support

Properties of OCL Constraints are Undecidable

Solutions:

- Analyze constraints for bounded system states (*small scope hypothesis*)
- Rely on hints from the user (interactive theorem proving)

On the other hand: Satisfaction of OCL constraints is decidable

Outline

Automated Analysis 1: Verification of OCL Constraints

Correctness properties of class diagrams defined by Cabot et al.:

- Liveliness of a class
- Lack of constraint subsumptions

Correctness properties for operation contracts:

- Applicability
- Non-redundant precondition
- Executability
- Correctness preserving
- Determinism

Tools for Verifying OCL Constraints

- UMLtoCSP
  - Analysis of OCL with the Eclipse constraint solver
  - Supports analysis of operation contracts

- UML2Alloy
  - Translation of OCL to the *Alloy* language

Automated Analysis 2: Animation of Operation Contracts

System state at precondition time
Operation arguments

Valid system state at postcondition time
Return value

Synonyms:

- Model execution
- Model simulation
- Code generation
- Compiler for "very high level language"

OCL as constraint programming language?

Animation Obstacles

- Efficiency
- Missing information in operation contracts
  - Modifies clauses
  - Objective function to optimize for

Workaround: Provide this information within UML profile

Problems that can be specified without objective function:

- (Topological) sorting
- Assembly line planning
- Stable marriage

Interactive Analysis: HOL-OCL

HOL-OCL embeds OCL into Higher Order Logic (HOL)

- HOL-language $\approx$ functional programming with quantifiers
- HOL can be analyzed with the interactive theorem prover Isabelle
  - supports user-defined tactical reasoning
  - automated decision procedures
- Library of formally proven mathematical theories

Applications of HOL-OCL:

- Proof of refinement relations
- Refinement to executable code
- Code verification

Another interactive approach:
KeY (verification of Java Card programs)

Conclusion

There are a variety of formal methods with application to OCL:

- Constraint verification
- Animation
- Refinement
- Code verification
- Test case generation

Tools are not yet fully developed.

$\Longrightarrow$ Breakthrough for OCL?