USING CONSTRAINT PROGRAMMING TO VERIFY UML / OCL MODELS

A SHORT SURVEY

Philippe Morignot

08 / 02 / 2012

Summary

- Introduction
  - Constraint Satisfaction Problems (CSPs)
  - UML / OCL
- Principles for using CSP to check UML / OCL models.
- Turning UML class diagrams into CSP (1)
- Turning UML class diagrams into CSP (2)
- Conclusion & references

Constraint Satisfaction Problem (1 / 2)

- A CSP is expressed as :
  - Variables v_i
  - Finite domains $D_i = \{ d_1, d_2, \ldots, d_{k(i)} \}$
  - Relations, which always hold, among variables: C_j
  - (Cost f)
- Example : SEND + MORE = MONEY

- Goal: For each v_i, find one value d_k from D_i which together satisfy every C_j. (And which minimizes f.)

Constraint Satisfaction Problem (2 / 2)

- Algorithm FIND-FIRST;
  1. Choose an unassigned variable v_i
  2. Choose a value d_k from v_i's domain D_i
  3. v_i <– d_k
  4. Propagate through C_j [NP complete]
     - IF there exists an empty domain D_l , THEN
       a. UNDO propagation of step 4
       b. UNDO assigment of step 3
       c. IF all assignments (v_i , d_k) have already been tried THEN FAILURE
       d. GOTO step 2 or step 1
  5. IF there exists a variable v_j which is not assigned THEN GOTO 1.
  6. SUCCESS
- Backtrack after step 6 : algorithm ENUMERATE.
  - Cost *f*: Constraint Satisfaction and Optimization Problem (CSOP).
- Packages: CHOCO from Bouygues' e-lab, (J)SOLVER from IBM (ex-ILOG), ECLIPSE from IC PARC, CHIP from COSYTEC, AQL from INOVIA, PROLOG IV from univ. Marseille, SICSTUS PROLOG, etc.

Unified Modeling Language (UML)

- Graphical modeling language in object-oriented software engineering.
- Standard of the OMG since 1997.
- Diagrams :
  - Structural (7): Class diagram, …
  - Behavioral (3): Use case diagram, …
  - Interaction (4): Sequence diagram, …

- Meta-modeling architecture: Meta-Object Facility.

Object Constraint Language

- Object-oriented.
- Specified by the OMG
- Used on UML diagrams
- Can represent:
  - Invariants:
    - Predicate which must always hold.
  - Pre- (resp. post-) conditions:
    - Predicate which must hold before (resp. after) an operation.
  - Result of a method (body):
    - The type of a context's result = type of the result of the designated operation.
  - Initial / derived value of an attribute.
  - …

Uses of CSP for checking UML / OCL models

- Uses:
  - Checking that a model (either hand written or generated) verifies the constraints of a meta-model.
    - Dresden OCL Toolkit (univ. Dresden).
  - Generating a sequence of tests
    - A model includes constraints on specifications or tests of an application.
    - Smart Testing.
- Avoiding bugs in class diagrams !
  - Bug: Zero possible instances of a class !
  - Bug: Mismatch in multiplicity of associations !

Turning class diagrams into CSP

- Satisfiability:
  - Definition: A user can possibly create a set of new objects and links over the classes and associations of the model, so that no model constraint is violated.
- A CSP has a solution ⇔ the model is satisfiable.
- Variants:
  - Strong satisfiability: The model must have a finite instantiation where the population of each class and association is at least one.
  - Weak satisfiability: same as above, but for « at least one class ».
  - Liveliness of a class c: same as above, but « where the population in c is non empty ».

Turning class diagrams into CSP
The CSP model: Classes

- CSP variables:
  - A list variable *InstanceC* :
    - *struct(c) = (oid,* f_1, …, f_n*)*
  - An integer variable *SizeC* (arbitrarily upper bounded).
- CSP constraints:
  - Number of links: *SizeC = length(InstanceC)*
  - Uniqueness of identifers: c*x* <> *cy* => *cx.oid* <> *cy.oid*

Turning class diagrams into CSP
The CSP model: Associations

- Variables:
  - A list variable *InstanceAS*:
    - $struct(InstanceAS) = (p_1, \ldots, p_n)$ where p_i = role of class
  - A integer variable *SizeAS* (arbitrarily upper-bounded)
- Constraints:
  - Number of links: $SizeAS = length(InstanceAS)$
  - Existence of referenced objects: $link.p_i = x.oid$

Turning class diagrams into CSP
The CSP model: Associations (cont'd)

- CSP constraints (followed):
  - Cardinalities:
    - *SizeAS < SizeClassX * SizeClassY*
    - *minClassXAS * SizeClassY < SizeAS < maxClassXAS * SizeClassY*
    - *minClassYAS * SizeClassX < SizeAS < maxClassYAS * SizeClassX*
  - Multiplicities of associations:
    - *minClassXAS < #{instanceAS.p1 = instanceClassX} < maxClassXAS*
    - *minClassYAS < #{instanceAS.p2 = instanceClassY} < maxClassYAS*

Turning class diagrams into CSP (4/6)
The CSP model: the ISA hierarchy

- No new CSP variables.
- CSP constraints:
  - Existence of instances in supertype:
    - *InstanceSub.oid = InstanceSup.oid*
  - Number of instances: *SizeClassSub* < *SizeClassSup*
  - Disjointness for a sup Csup and subs $Csub_i$
    - $SizeCsup > sum(SizeCsub_i)$
    - *ObjectI.oid = ObjectJ.oid => I = J*
  - Completeness of a super
    - $SizeCsup < sum(SizeCsub_i)$
    - *ObjectSup.oid = ObjectSub.oid*

Turning class diagrams into CSP (5/6)
The CSP model: OCL constraints

- Invariants.
- Expressed in ECLIPSE Prolog.
  - Less direct!
- An OCL constraint is considered as an instance of the OCL meta-model
  - A node corresponds to constants and variables of the constraint.

Turning class diagrams into CSP (6/6) Implementation

- (1) Finding the sizes ; then (2) finding the instances.
- ECLIPSE and JAVA libraries, extending Dresden OCL.
- Tool UMLtoCSP http://gres.uoc.edu/UMLtoCSP/

A second way to solve the same problem

- Principle : represent class diagrams in Description Logics (concepts, roles, individuals) and use a CSP engine as a reasoner.
  - Finite satisfiability problem.
  - Binary associations.
  - No OCL constraints.
- Sketch of the CSP model:
  - A variable is the number of instances of a class.
  - Another variable is the number of associations.
  - Constraints are inequalities among variables.
  - ISA hierarchies with associations lead to an explosion of variables.
- Experiments with OPL Studio (SOLVER) on Common Information Models (management information on a network/company).

Conclusion

- Goal: Avoiding bugs in UML / OCL models
  - A class with zero possible instances.
  - Mismatch in multiplicity of associations.
- Turning a class diagram into a CSP.
  - An UML / OCL model is satisfiable ⇔ a CSP has a solution.
  - Automatically generating a CSP: Representing classes, associations, ISA hierarchy as variables and constraints.
  - Solving the CSP with an off-the-shelf constraint engine.

- Future work:
  - Other UML diagrams ?
  - Other OCL constraints ?
  - Scaling up ? (e.g., n x 100 classes)
  - Using a SAT-solver. What about MIP, evolutionnary algorithms, … ?

References

1. Cabot, J.; Clariso, R.; Riera, D.; Verification of UML/OCL Class Diagrams using Constraint Programming. Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International Conference, 2008.
2. Marco Cadoli, Diego Calvanese, Giuseppe De Giacomo, Toni Mancini. Finite Model Reasoning on UML Class Diagrams Via Constraint Programming. AI*IA 2007: Artificial Intelligence and Human-Oriented Computing Lecture Notes in Computer Science, 2007, Volume 4733/2007, 36-47.
3. Mathias Soeken, Robert Wille, Mirco Kuhlmann, Martin Gogolla, Rolf Drechsler, Verifying UML/OCL models using Boolean satisfiability. Proceedings of the Conference on Design, Automation and Test in Europe, 2010.

THANK YOU FOR YOUR ATTENTION !