

SyVOLT: Full Model Transformation Verification Using Contracts

Levi Lúcio, **Bentley James Oakes**

McGill University, Canada
fortiss, Munich

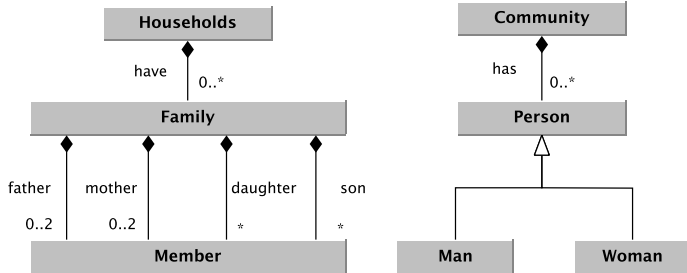
July 20, 2016

- Model transformations are at the heart of model-driven engineering
- Want to verify correctness for transformation specifications
 - Verify visual contracts
 - Identify combinations of rules where contracts hold or not
- Objective: Verification for all input models
 - Input independence

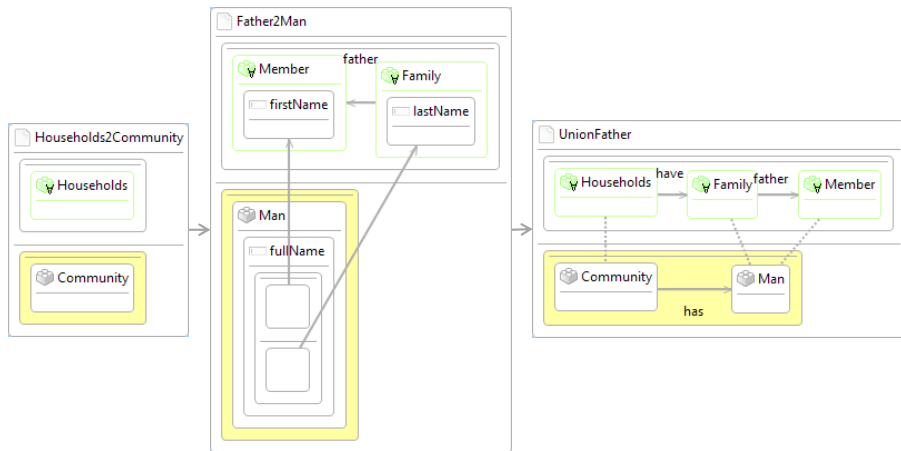
- Visual language for model transformations
- Graph-based, rule-based
- Out-place so no rewriting performed
 - Suited for 'translation' transformations
- All its computations are terminating and confluent
- Unbounded loops during execution are not allowed
- Rules are grouped in sequential layers

Selim, Gehan et al. "Model transformations for migrating legacy deployment models in the automotive industry." *Software and Systems Modeling* 14, no. 1 (2013): 365-381.

Transformation Metamodels

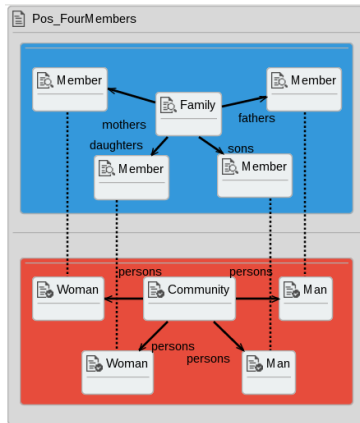


- Transform *Members* to *Men* and *Women*



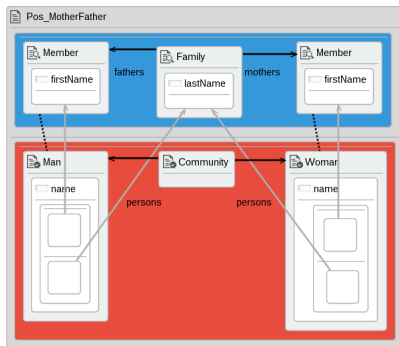
- Rules arranged in layers
- Match graph on top of rules
- Apply graph on bottom
 - Produced when match graph is found

Contracts



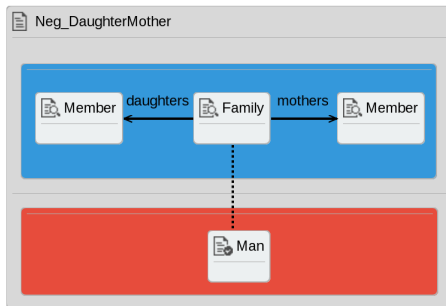
- Pre- / post- visual contracts
- If blue graph is found in input model, then red graph is found in output model
- Objective: Prove for all input models/transformation executions - input independence

Contracts



- Reasoning about attributes of elements
- *Is the full name of the produced Person correctly created from the last name of the Family and the first name of the Member?*

Contracts

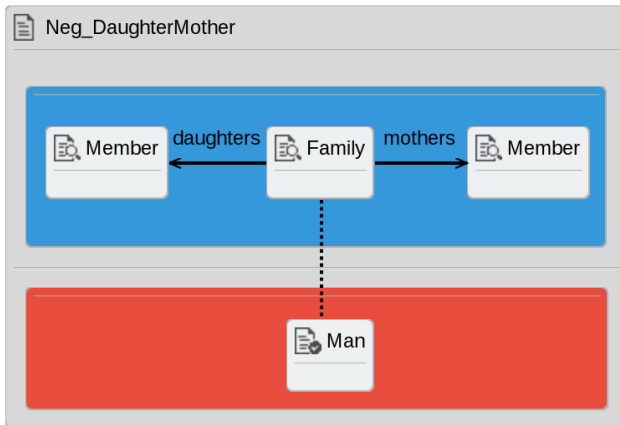


- A contract that will not hold
- *A family with a mother and a daughter will always produce a community with a man*

- Tool to prove visual contracts on DSLTrans transformations
- All possible executions of the transformation are symbolically constructed
 - Built as sets of rules called path conditions
 - No rules execute, only rule 1 executes, rule 1 and rule 2 both execute
 - Rule dependencies/combinations resolved
 - Final set of path conditions represents all possible transformation executions
- A contract holds for a transformation if it holds for all generated path conditions

L. Lúcio, B. Oakes, and H. Vangheluwe. A technique for symbolically verifying properties of graph-based model transformations. Technical report, Technical Report SOCS-TR-2014.1, McGill U, 2014.

Contract Proving Example

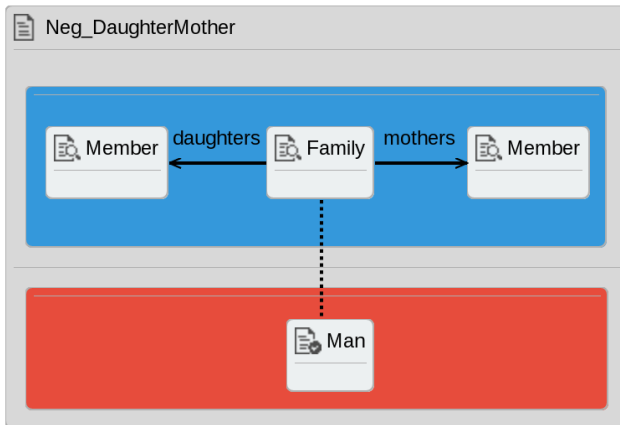


- *A family with a mother and a daughter will always produce a community with a man*
- Fails on path condition:
'HEmpty_HRoot_HMotherRule_HDaughterRule'

Performance

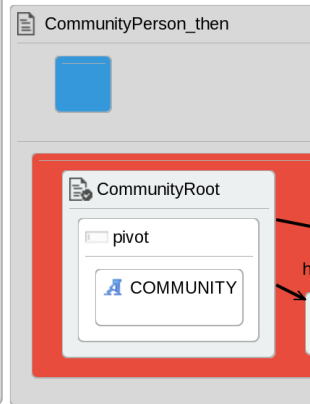
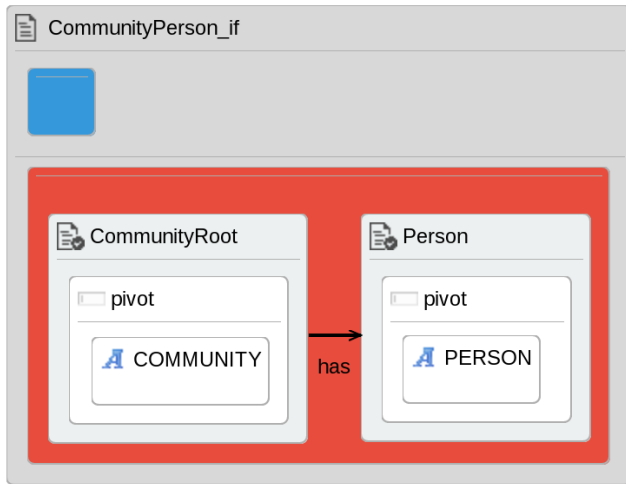
	ATL/ DSLTrans Rules	Path Conds. Generated	Time (s)	Contracts Proved	Time (s)	Memory (MB)
Families-to-Person	5 / 9	101	0.24	4	0.52	54
Extended Families-to-Person	10 / 19	366	3.89	10	7.35	59
GM-to-AUTOSAR (handbuilt)	5 / 9	13	0.18	9	0.15	58
GM-to-AUTOSAR (HOT)	5 / 9	10	0.26	9	0.15	60
UML-to-Kiltera	20 / 17	322	1.86	15	11.99	55

Contract Expressibility



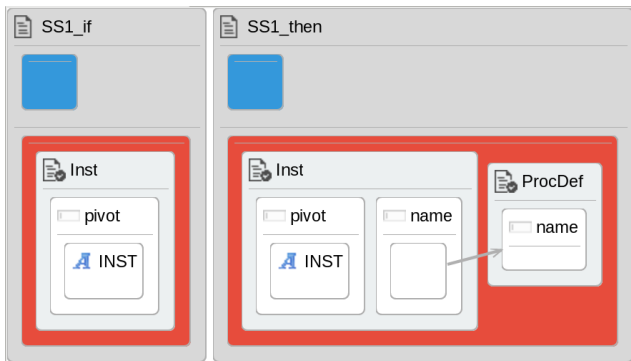
- The next few slides will discuss some concepts that can be expressed with our contracts.
- This information is reproduced from: Selim, G.M.: Formal Verification of Graph-Based Model Transformations. Ph.D. thesis, Queen's University (2015)

Multiplicity Invariants



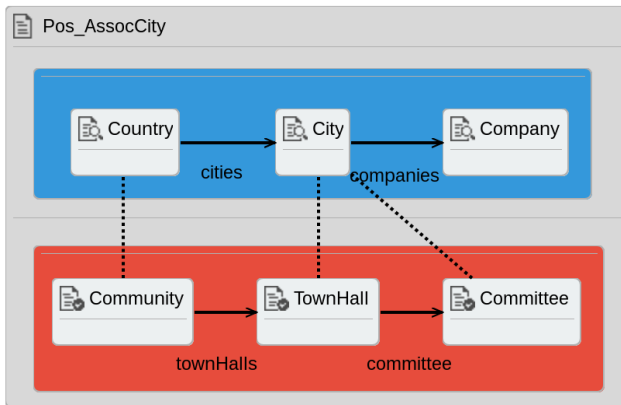
- Can express multiplicity invariants contained in the source or target metamodel.

Syntactic Invariants



- Check whether the path condition is well-formed with respect to the input or output syntax
- *if there is an Inst element, then that Inst element has the same name as a ProcDef element*

Pattern Contracts

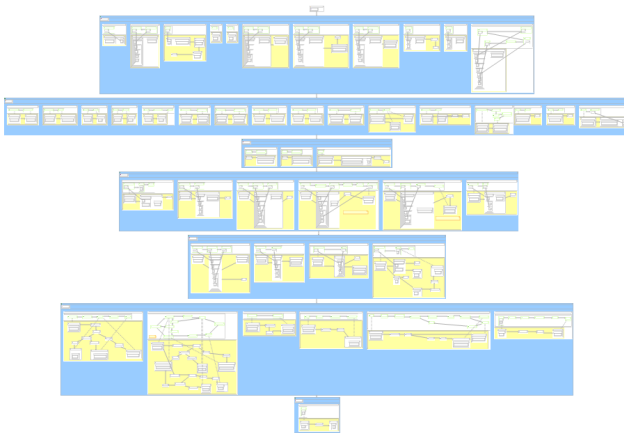


- Relate elements in the input model to elements in the output model
- *A country which contains a city, which contains a company, should produce a corresponding community, town hall, and committee*

- Contract language is currently limited as it can only represent structural conditions and not arbitrary expressions
 - Much less powerful than OCL
 - Current implementation does not contain operators for sets, or for handling non-String attributes.
- Abstractly representation of the rules that execute, and the number of times each rule executes,
- Cannot validate instance data for input or output models
 - All input names start with a capital letter.

- Verify contracts on the mbeddr to C transformation
- mbeddr takes components of embedded system expressed in MPS and generates C code
- Transformation must be verified in order to ensure that resulting C code is valid

Challenges

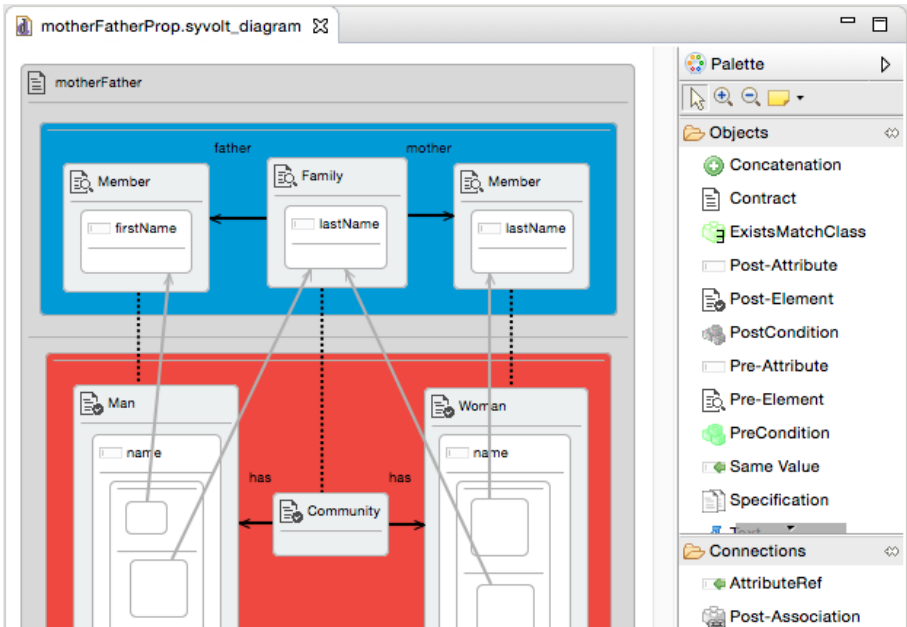


- Scalability
- What are appropriate contracts for this transformation?

- Fully Verifying Transformation Contracts for Declarative ATL (MODELS 2015)
Bentley James Oakes, Javier Troya, Levi Lucio, and Manuel Wimmer (McGill University, Canada; Vienna University of Technology, Austria)
 - Extended to journal article: Full Contract Verification for ATL using Symbolic Execution, SoSym (to appear)
- Finding and fixing bugs in model transformations with formal verification: An experience report (AMT)
Gehan M. K. Selim, James R. Cordy, Juergen Dingel, Levi Lucio and Bentley James Oakes
(Queen's University, Canada; McGill University, Canada)

■

Eclipse Plugin



- Can verify visual contracts on DSLTrans transformations in feasible time
- Contracts verified on all transformation executions
- Eclipse plugin to build transformation and contracts
- Future work is to focus on contract-driven development of model transformations
- Thank you for your time!

SyVOLT: Full Model Transformation Verification Using Contracts