SyVOLT: Full Model Transformation Verification Using Contracts

Levi Lúcio, **Bentley James Oakes**

McGill University, Canada
fortiss, Munich

July 21, 2016

Motivation and Overview

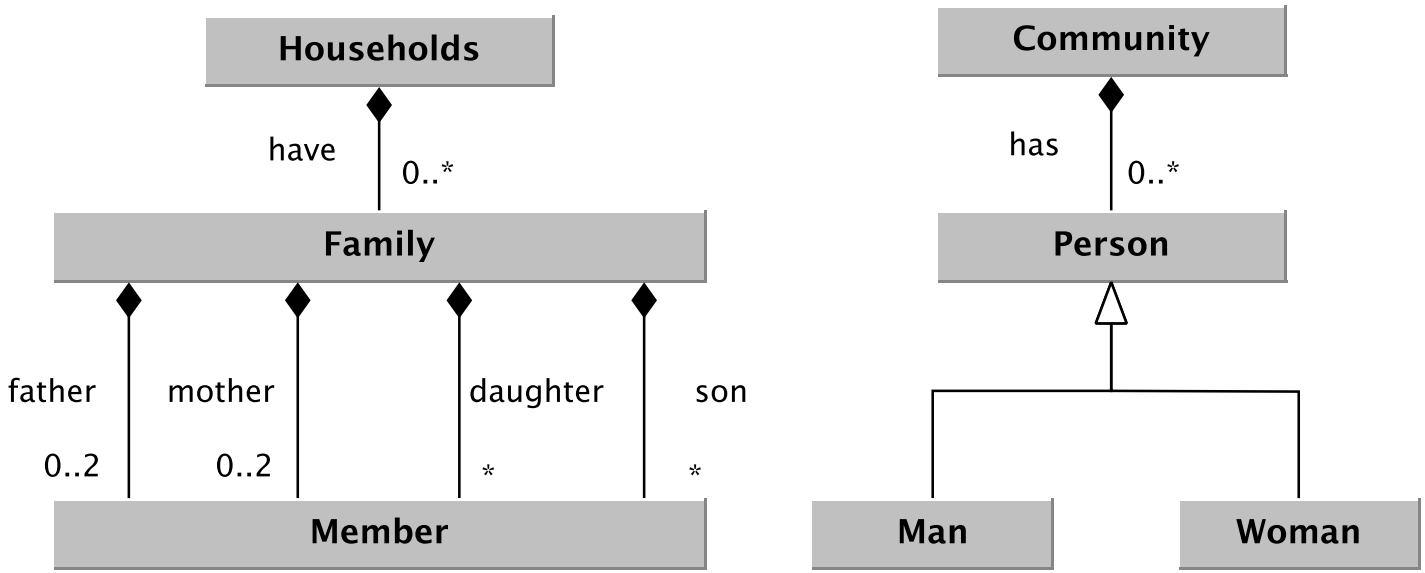
- Model transformations are at the heart of model-driven engineering
- Want to verify correctness for transformation specifications
  - Verify visual/structural contracts
  - Identify those combinations of rules where contracts hold or not
- Objective: Verification for all input models
  - Input independence

DSLTrans Transformation

- 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

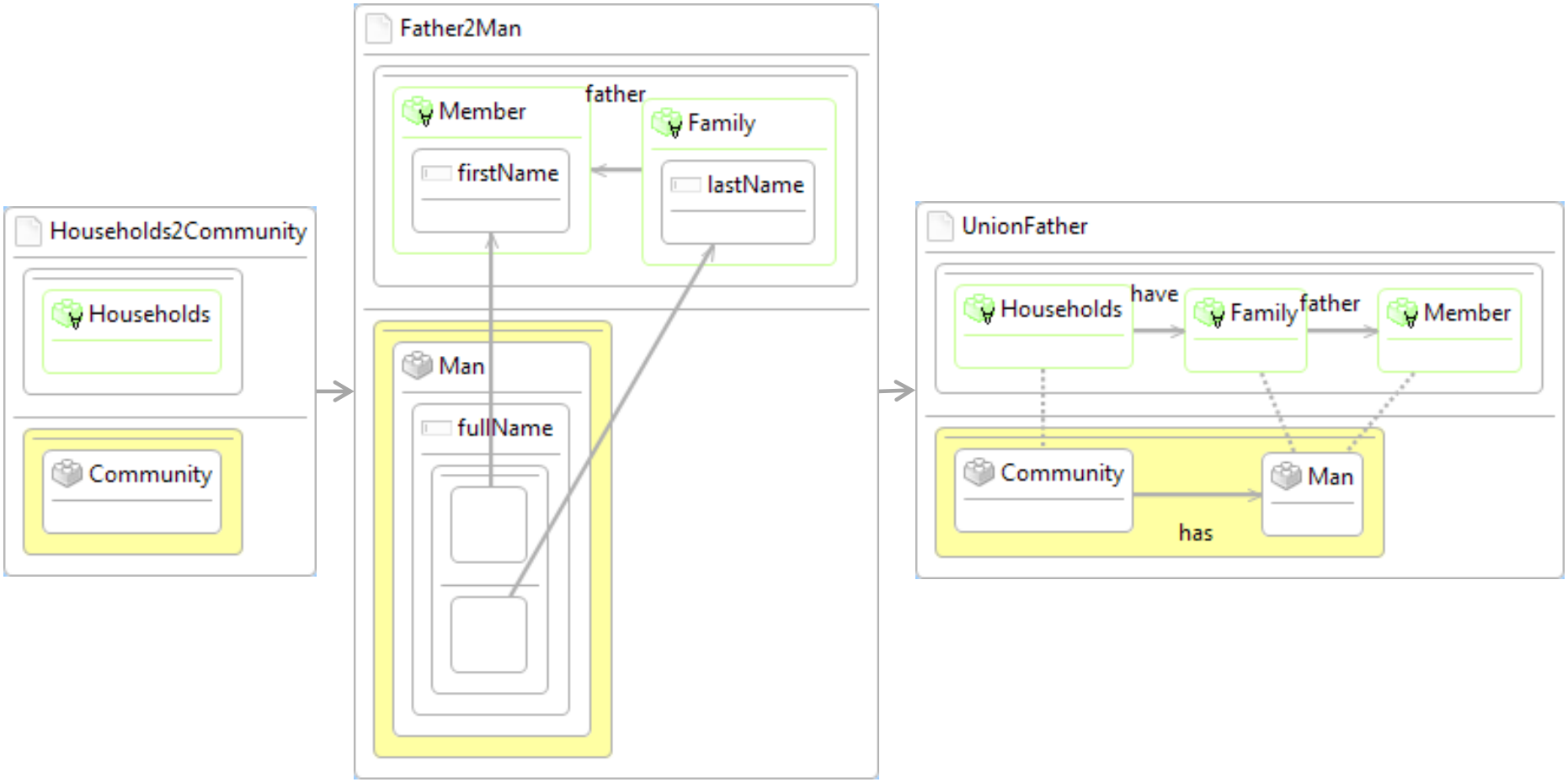
Selim, Gehan et al. "Model transformations for migrating legacy deployment models in the automotive industry." Software and Systems Modeling 14, no. 1 (2013): 365-381.

Transformation Metamodels



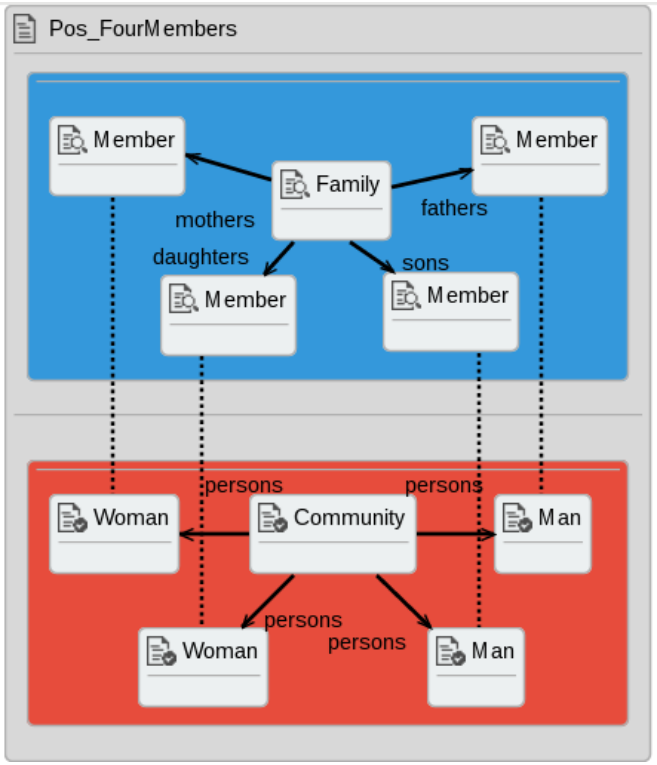
- Transform *Members* to *Men* and *Women*

DSLTrans



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

Pre- / Post- Visual Contracts



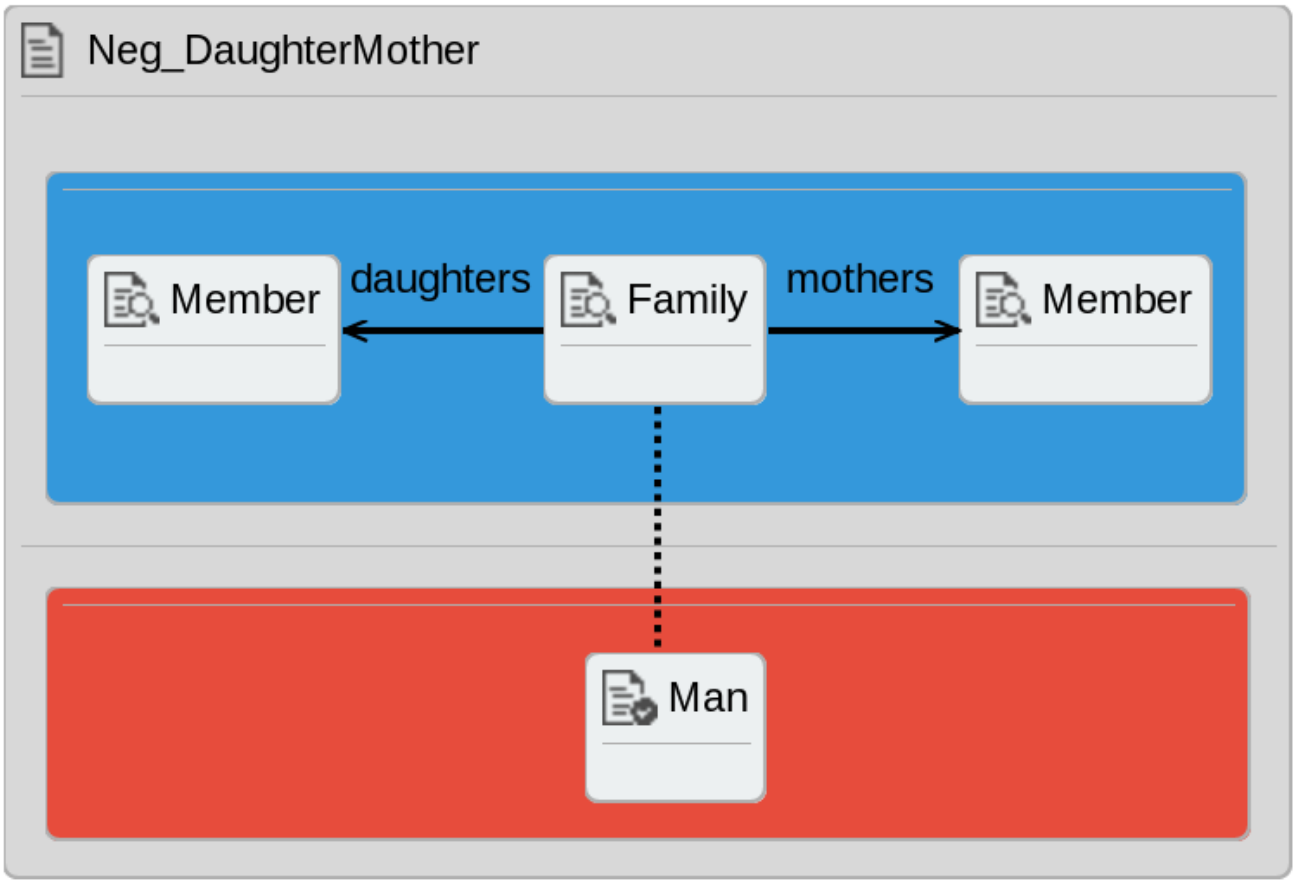
- If blue graph is found in input model, then red graph is found in output model
- Objective: Prove for all input models/transformation executions - input independence
- *A family with a father, mother, son, daughter should always produce two males and two females in the target community*

SyVOLT Tool

- Proves 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 finite set of path conditions represents all possible transformation executions
- A contract holds for a transformation if it holds for all generated path conditions
- Otherwise, counter-example is produced

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



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

- Multiple contracts can be proved in seconds

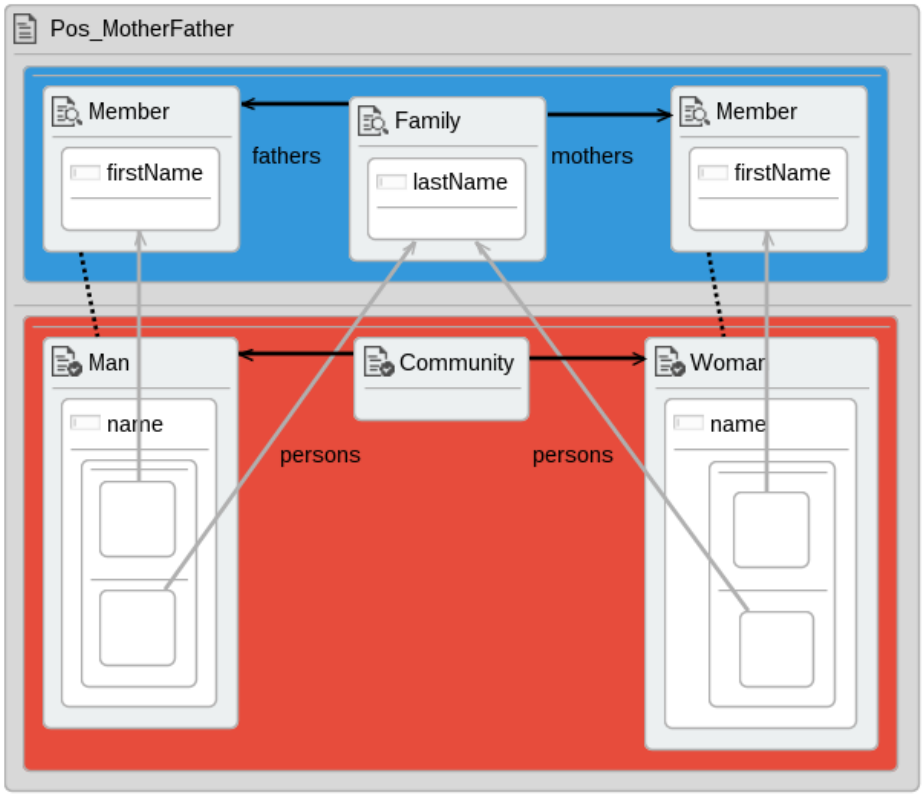
Contract Expressibility

- Next few slides will discuss concepts expressible with our contract language.
  - Element attributes
  - Propositional logic and pivots
  - Multiplicity invariants
  - Syntactic invariants
  - Pattern contracts
- Rule reachability is also handled by the prover, which reports if a rule cannot be proven to execute

---

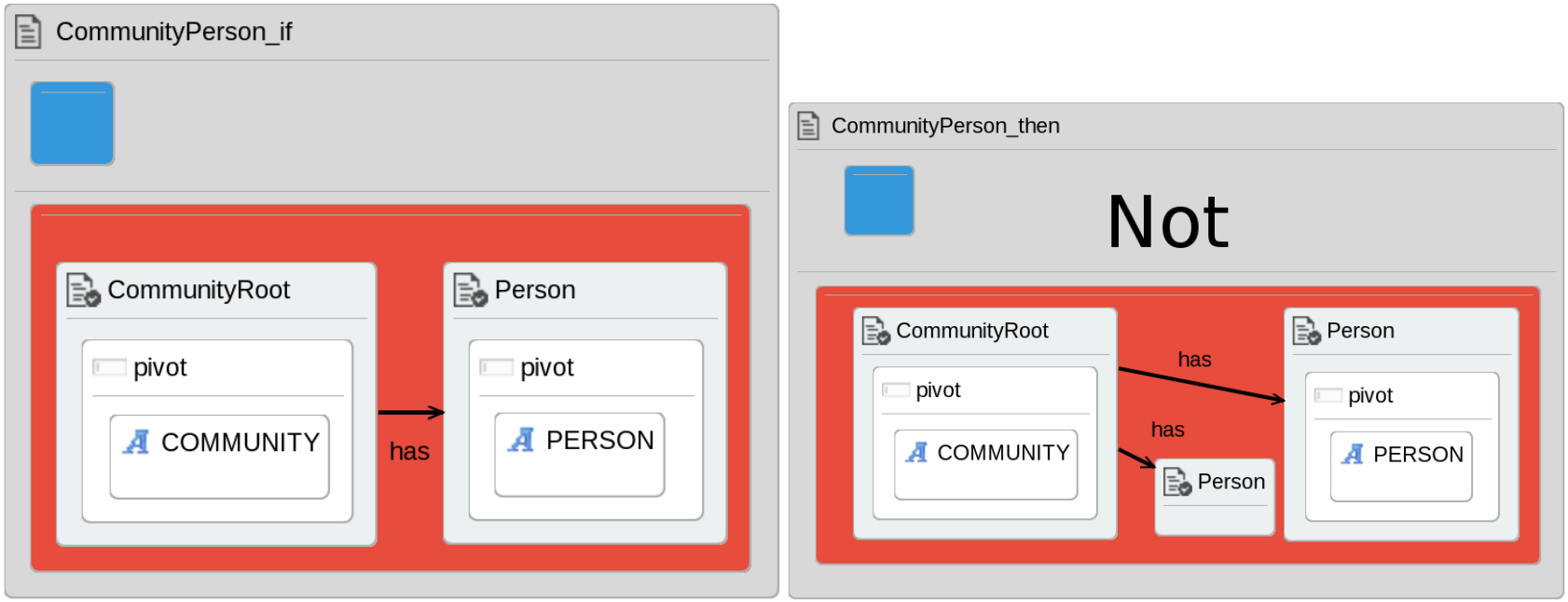
Selim, G.M.: Formal Verification of Graph-Based Model Transformations. Ph.D. thesis, Queen's University. 2015.

Element Attributes



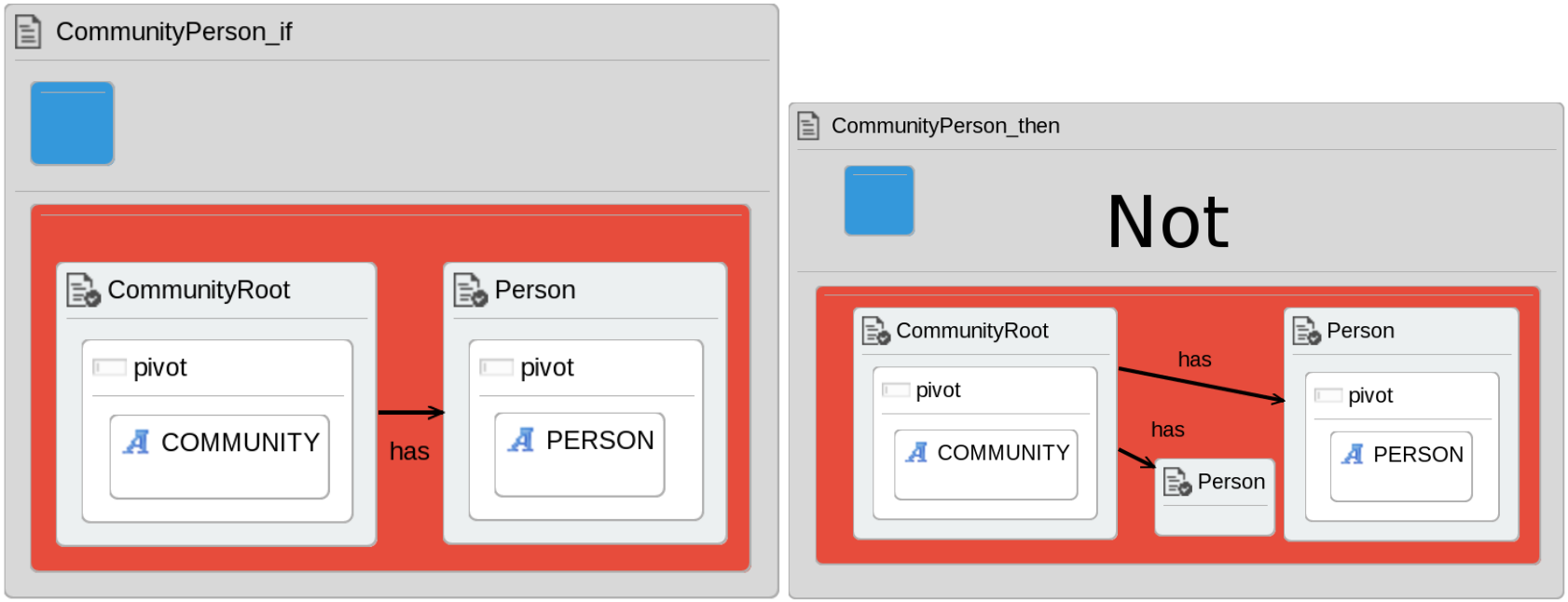
- Reasoning about (String) attributes of elements
- *Is the full name of the produced Person correctly created from the last name of the Family and the first name of the Member?*

Propositional Logic and Pivots



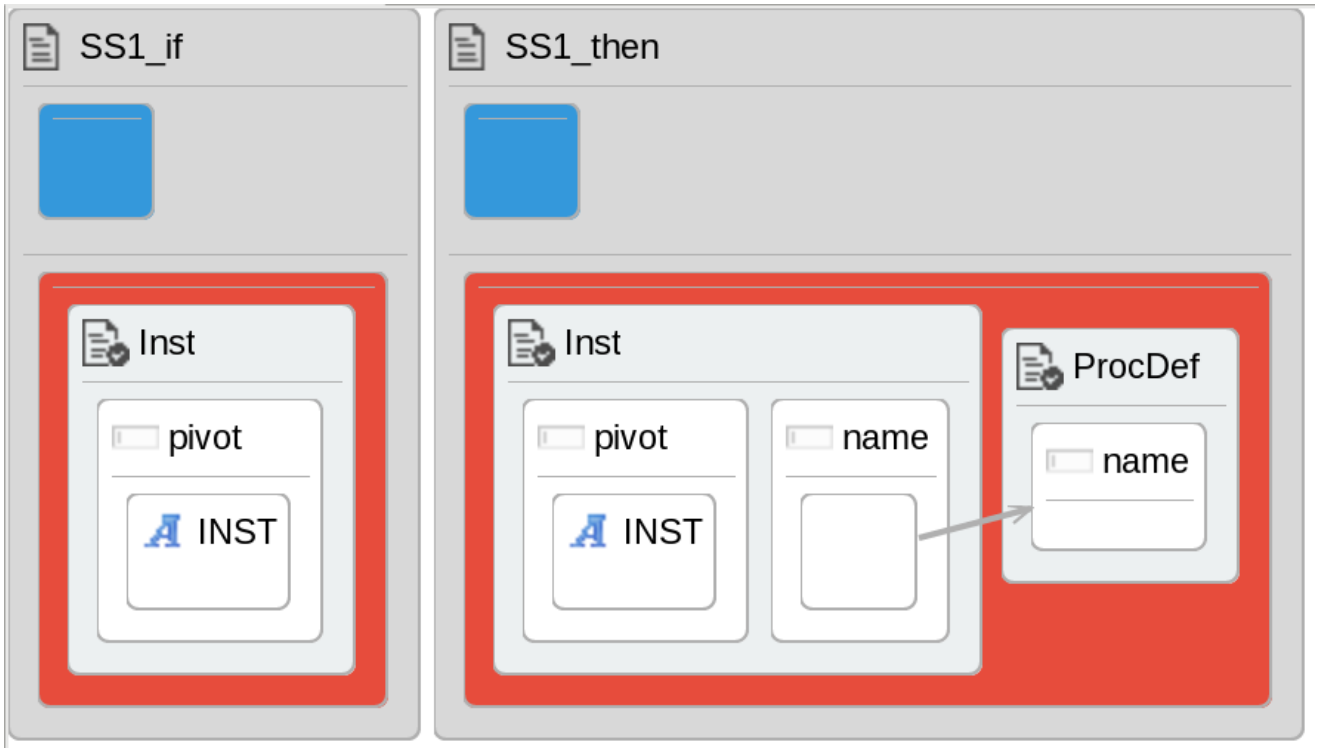
- Contracts can be combined with AND, OR, NOT, IF-THEN
- Pivots ensure that same element is bound in both contracts
- *Only one Person is in the Community in the output model*

Multiplicity Invariants



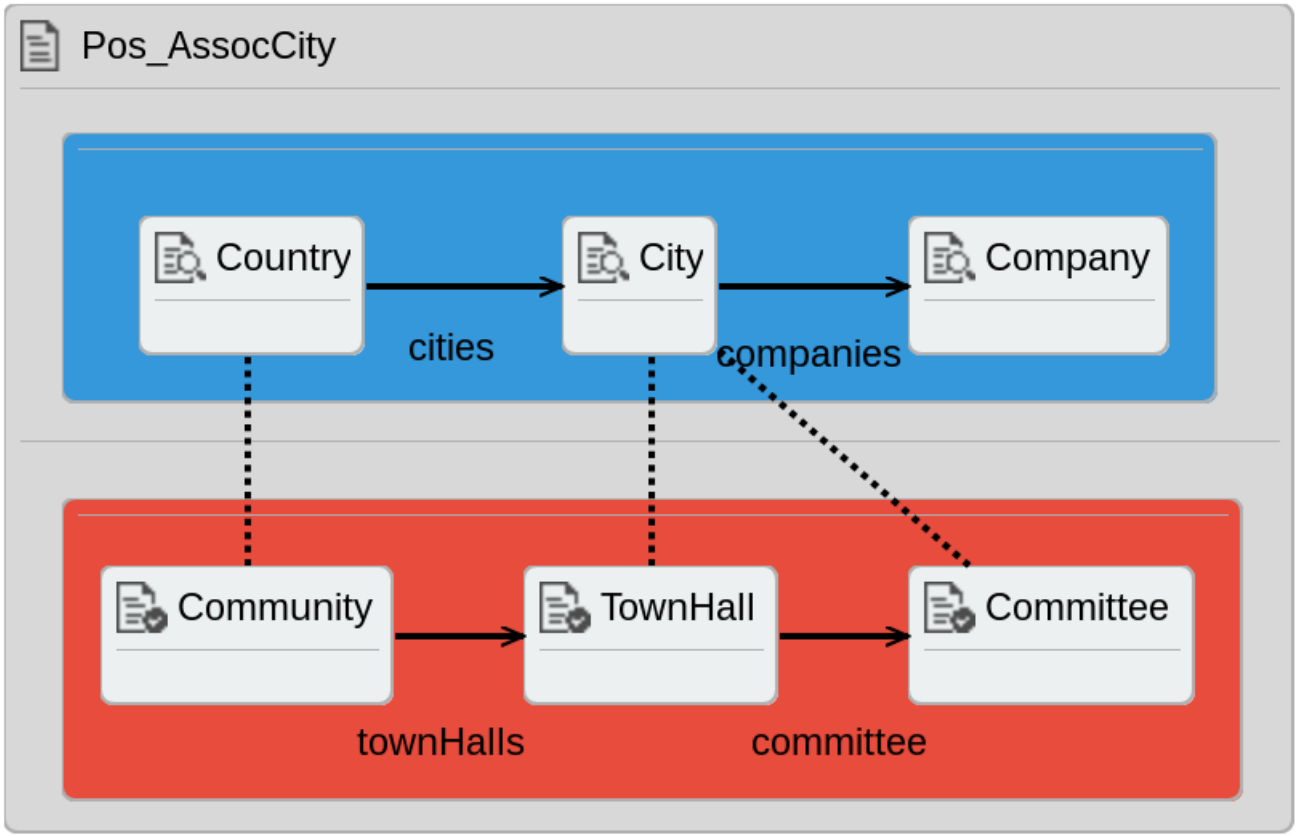
- *Only one Person is in the Community in the output model*
- Abstraction of our approach loses multiplicity information
  - Multiple applications of a rule are not represented
- Contract only fails if *two Persons are always created in output*

Syntactic Invariants



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

Pattern Contracts



- Relates elements in input model to elements in output model
- *A country which contains a city, which contains a company, produces a corresponding community, town hall, and committee*
- Intention is to allow verification that multiple rules are interacting in a valid way

Contract Limitations

- Limitations from symbolic execution technique
  - Loses multiplicity information
  - Difficult to reason about negative contracts
  - Cannot validate instance data
- DSLTrans transformation language only manipulates Strings
  - Can be worked around

Contract Limitations

- Contract language equipped for structural queries
  - Cannot compute queries
- 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.

Current Work - mbeddr

- 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

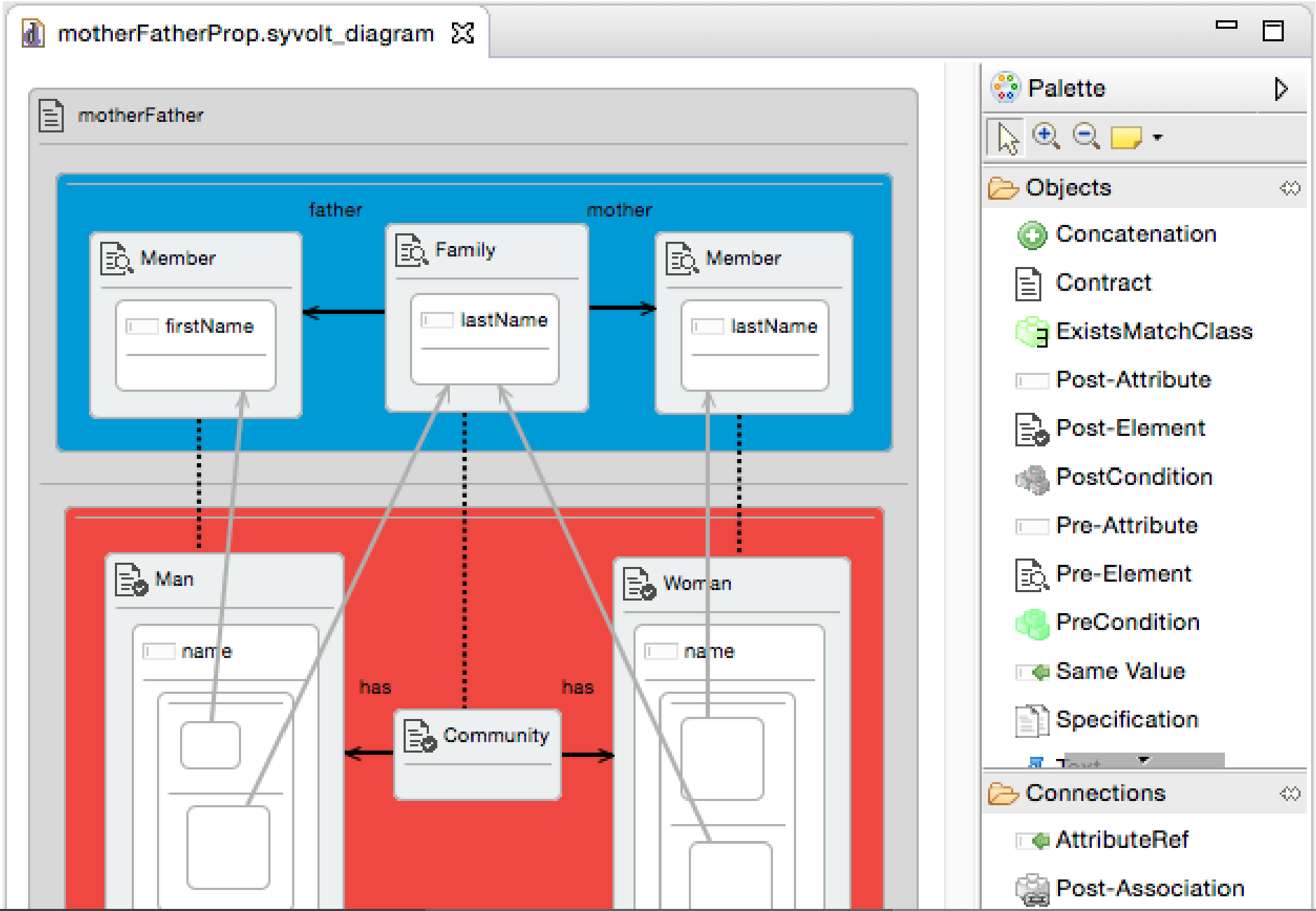
Challenges

- Scalability
- What are appropriate contracts for this transformation?

Other Work

- 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



Conclusion

- 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