Specification and Verification of Properties for Graph-Based Model Transformations

Gehan M. K. Selim, Levi Lucio, James R. Cordy, Juergen Dingel, Bentley J. Oakes

Agenda

Problem Statement

DSLTrans Model Transformation Language

Symbolic Model Transformation Property Prover

- Overview
- Phase 1: Path Condition Generation
- Phase 2: Property Verification

Industrial Case Study

Discussion

Conclusion & Future work

Problem Statement

Prove pre- post- condition structural properties

- **On translation model transformations**
  - Example: Industrial migration transformations
- **For all executions**
  - No extra elements added/removed

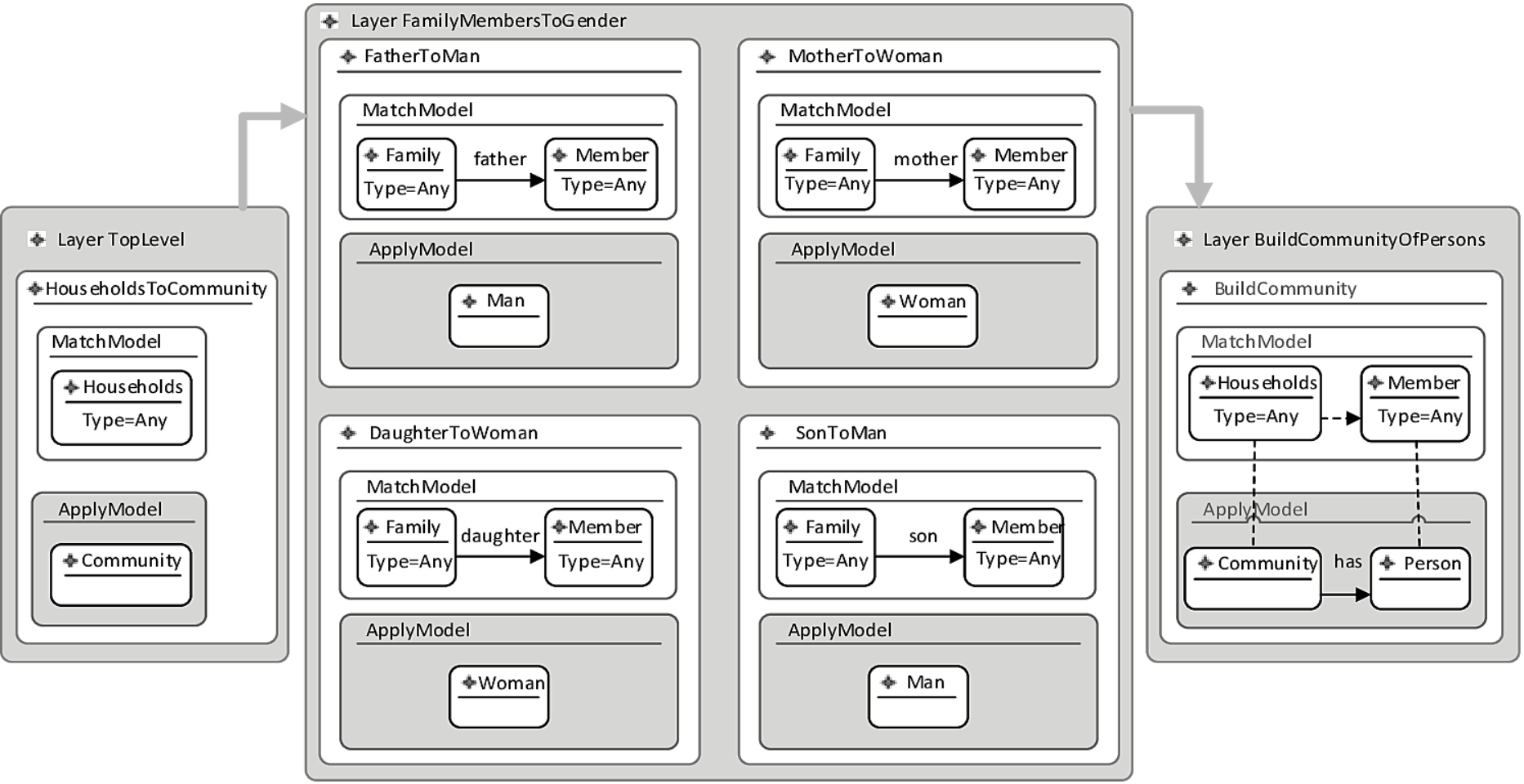
- **Infinite amount of transformation executions means the proof needs to be done on abstractions**
  - **Named path conditions in algorithm**

Differences from Current Transformation Verification Tools

- Input-independent
- Little mathematical background required (v.s. Maude)
- Some scalability tests on industrial-size transformations
- Verifies multiple property types
- Proof for validity and completeness of verification technique

DSLTrans Transformation

Persons to Community

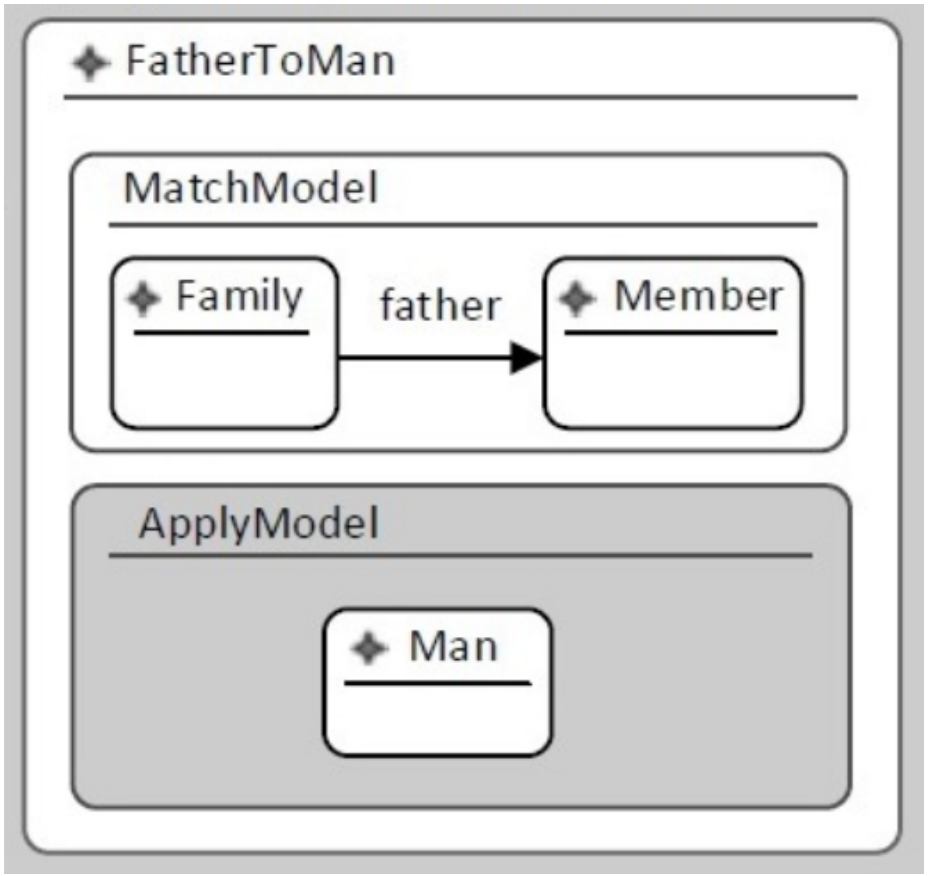


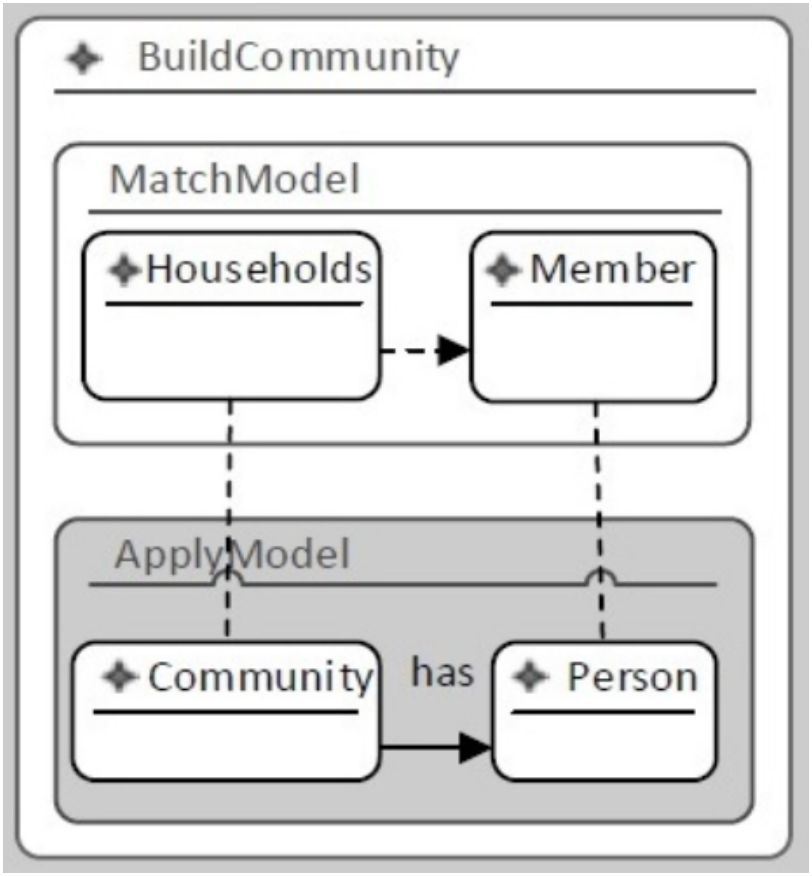
Restricted form of graph transformations

Turing-incomplete **Out-place**

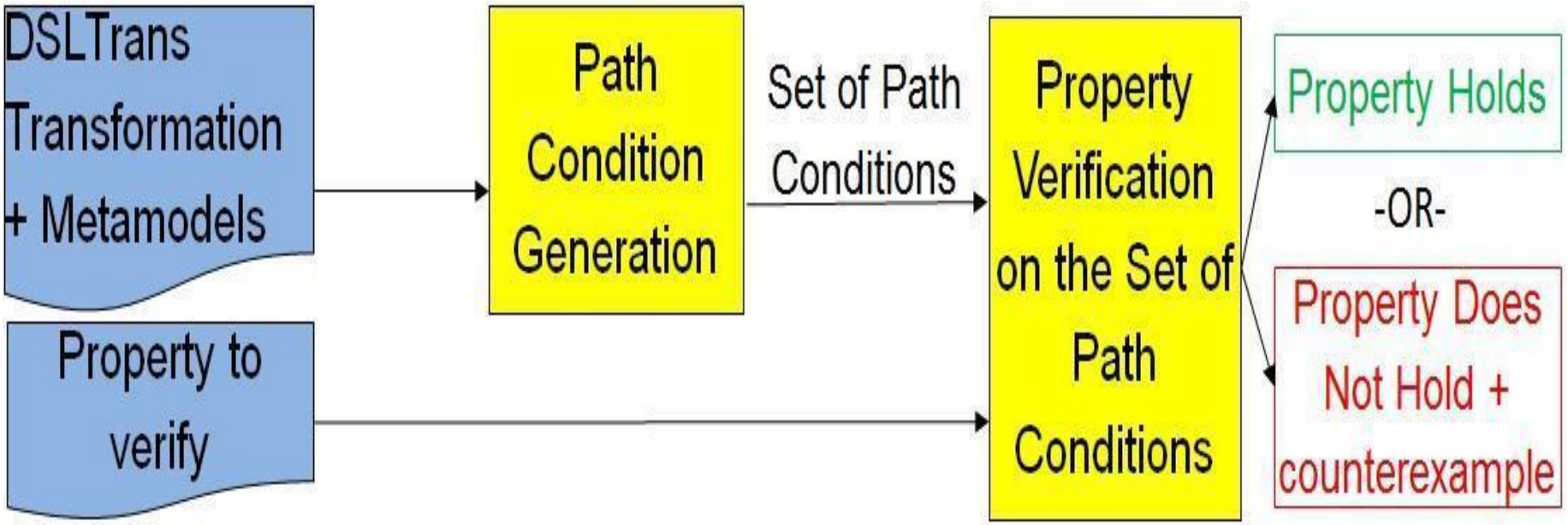
DSLTrans Transformation

Persons to Community Rule

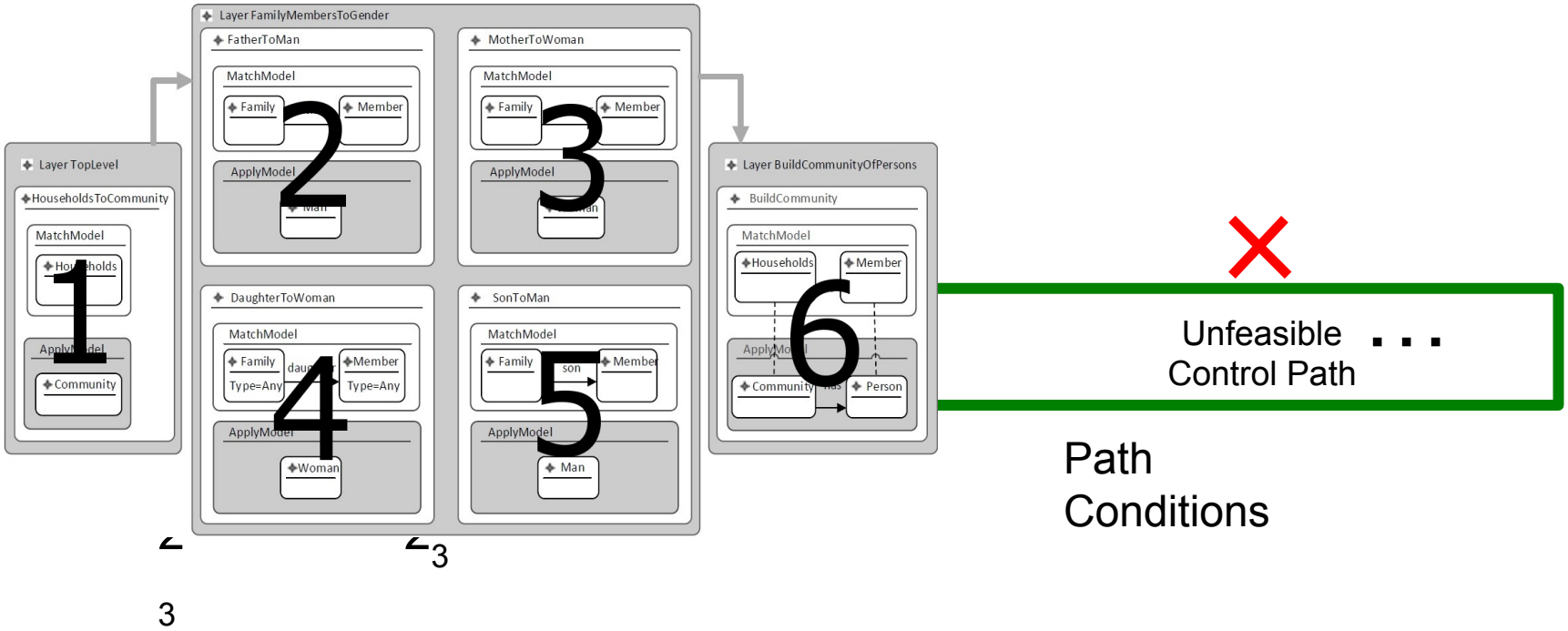




Symbolic Model Transformation Property Prover: Overview

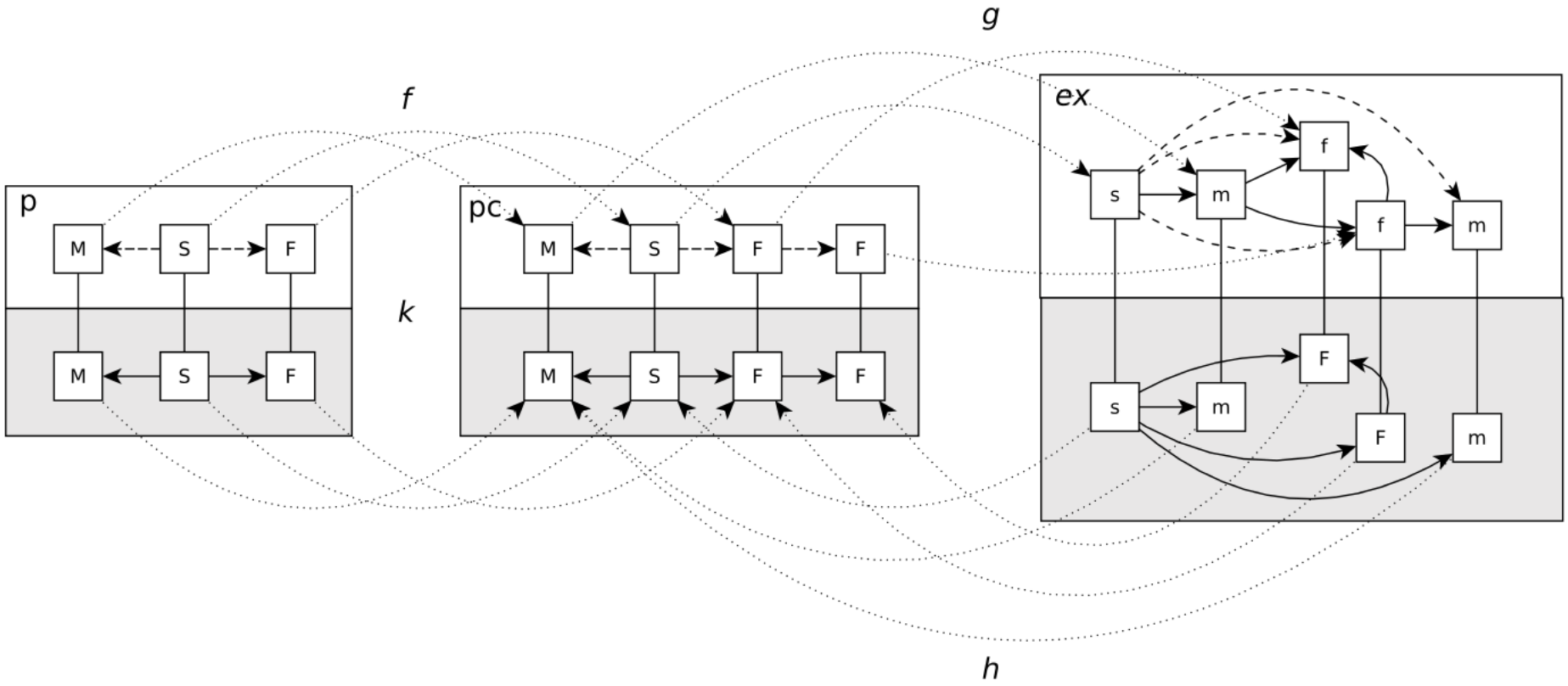


Phase 1- Path Condition Generation



Based on: L. Lucio, B. Barroca, V. Amaral “A Technique for the Verification of Model Transformations” Proceedings of MoDELS, 2010.

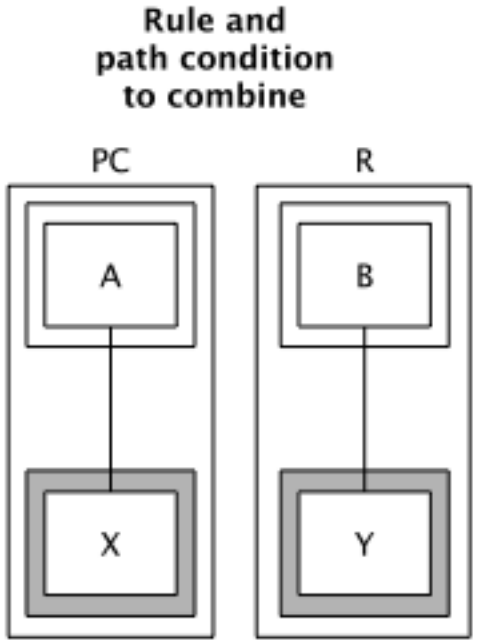
Abstraction Relation



Prove properties on path condition
Holds on abstracted transformation executions

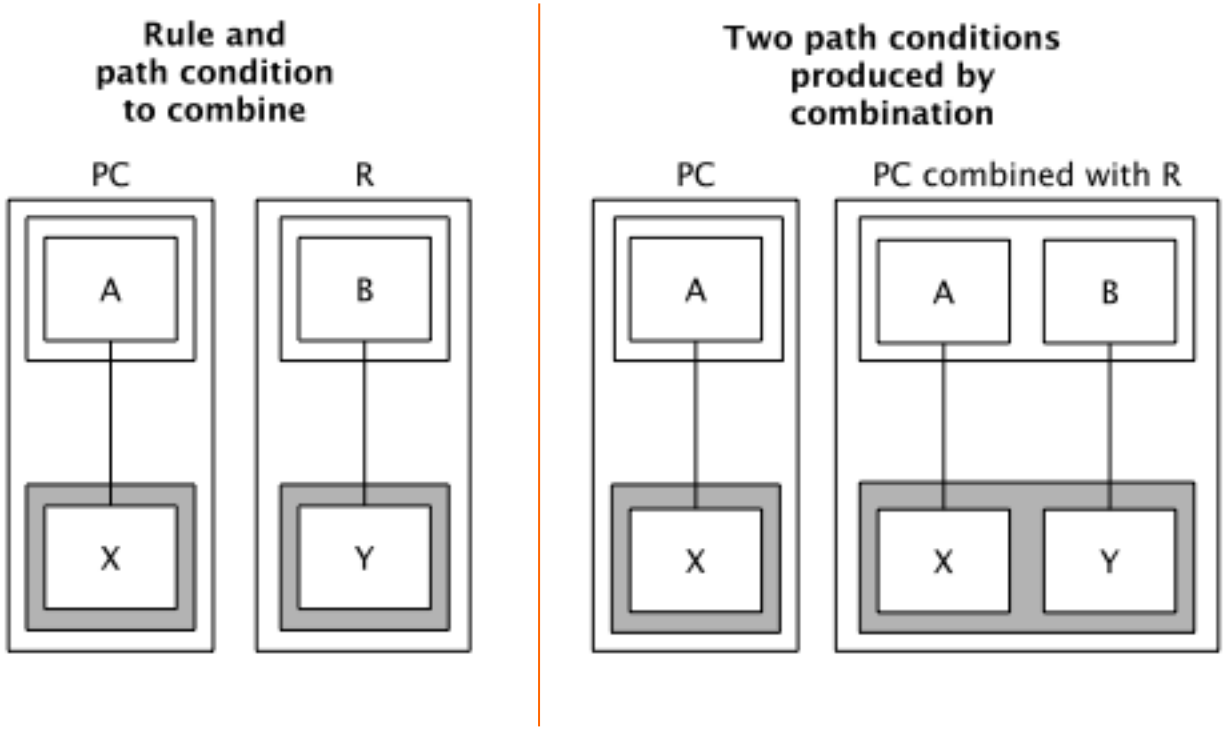
Combining Path Condition with Rule

Case 1: No Dependencies



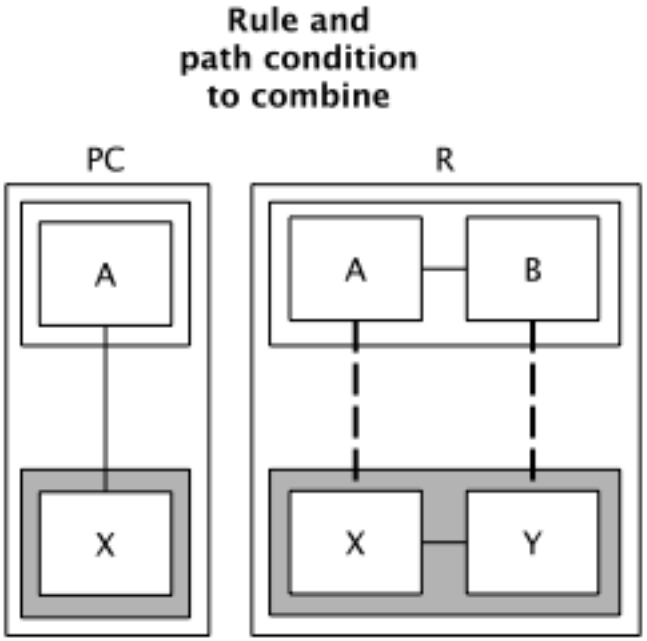
Combining Path Condition with Rule

Case 1: No Dependencies



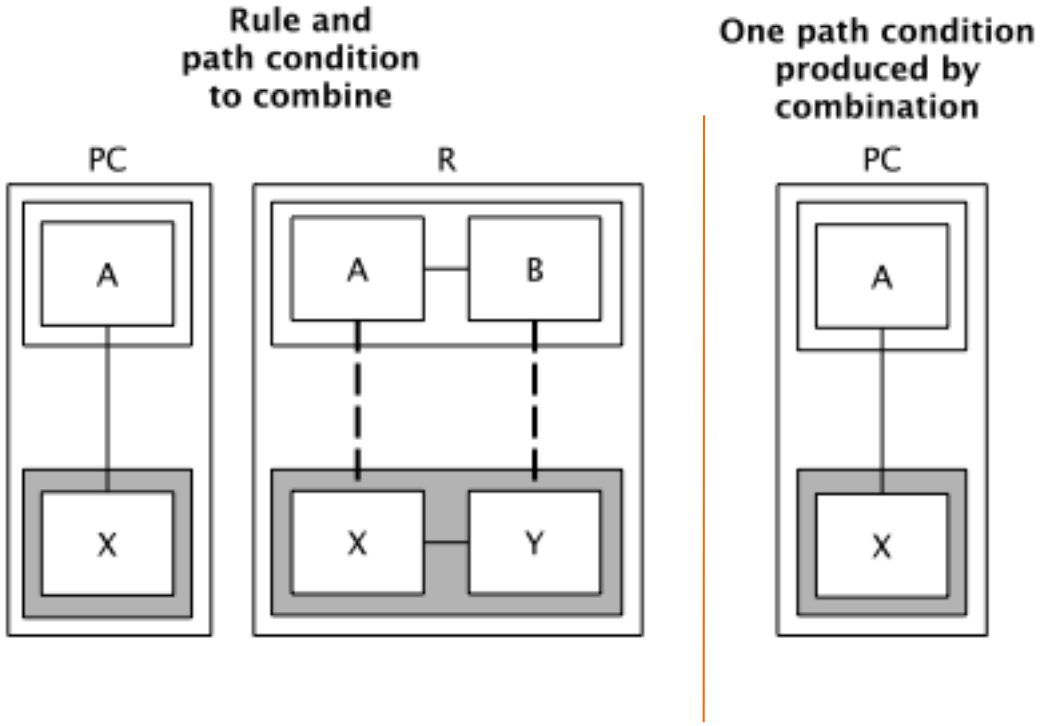
Combining a Path Condition with a Rule

Case 2: Rule has Dependencies and Cannot Execute



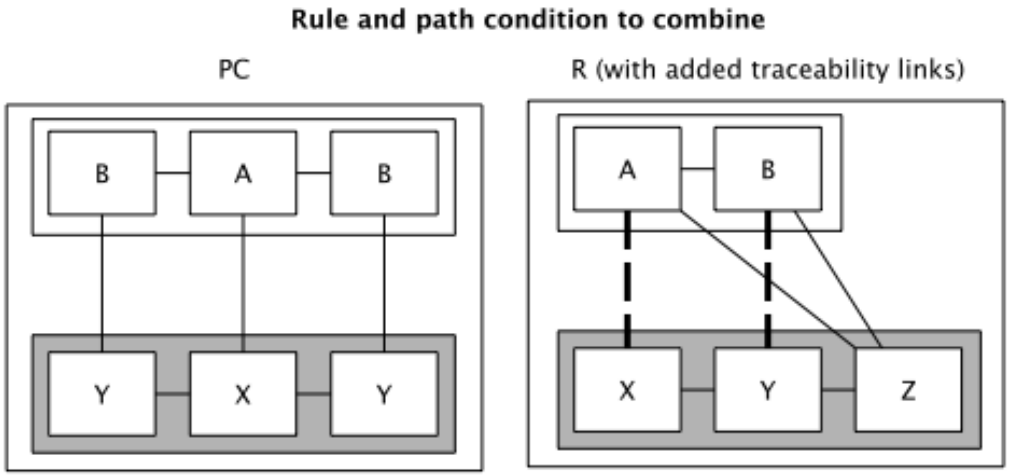
Combining a Path Condition with a Rule

Case 2: Rule has Dependencies and Cannot Execute



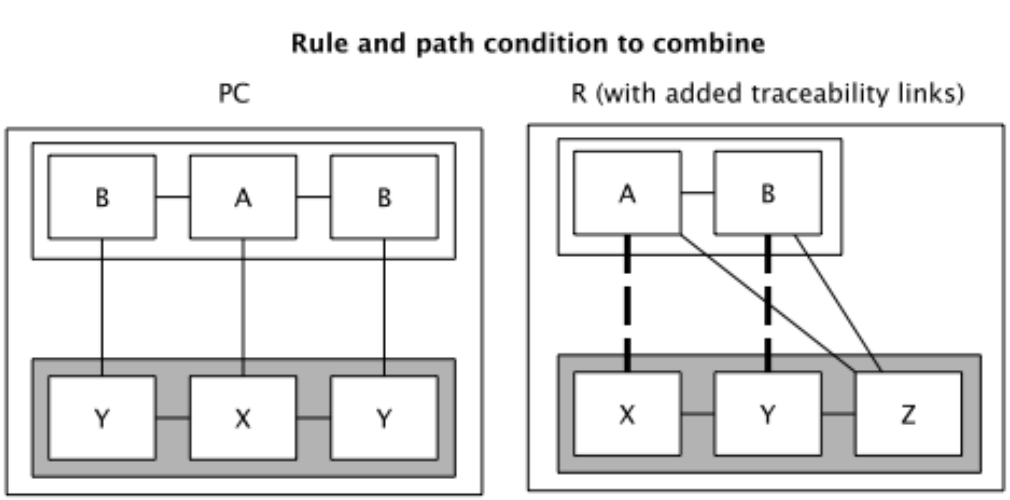
Combining a Path Condition with a Rule

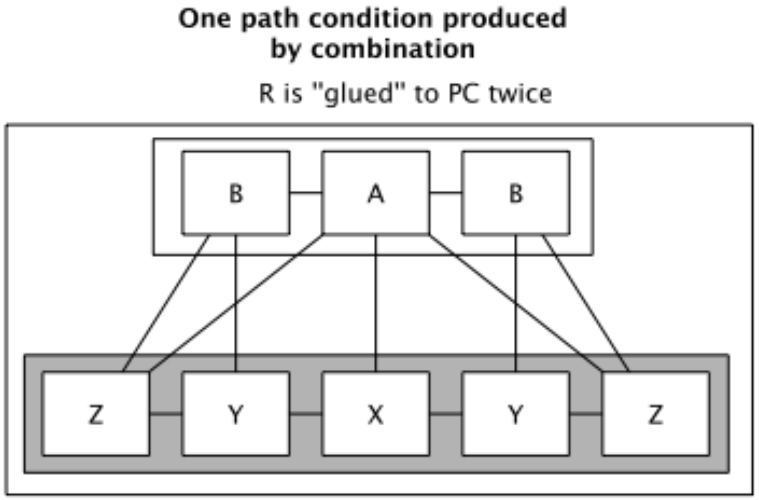
Case 3: Rule has Dependencies and *Will* Execute



Combining a Path Condition with a Rule

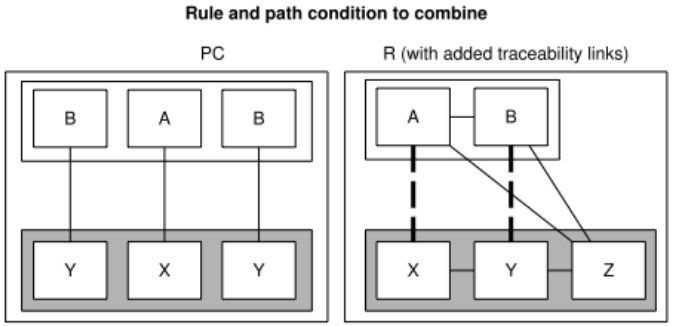
Case 3: Rule has Dependencies and *Will* Execute





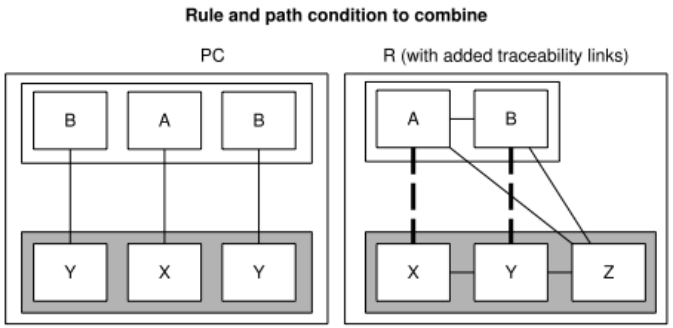
Combining a Path Condition with a Rule

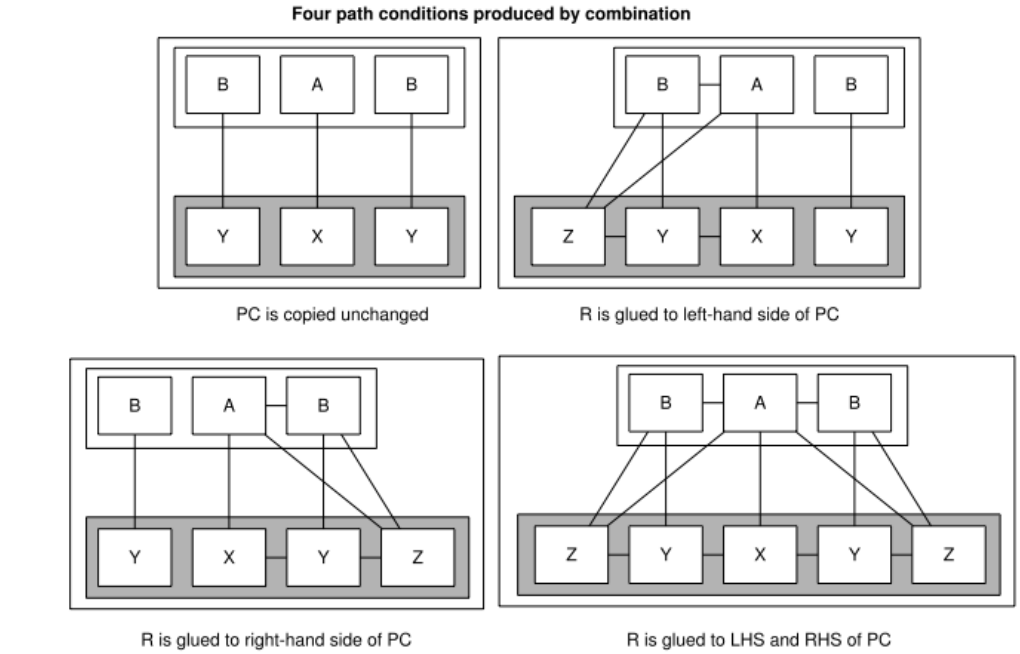
Case 4: Rule has Dependencies and *May* Execute



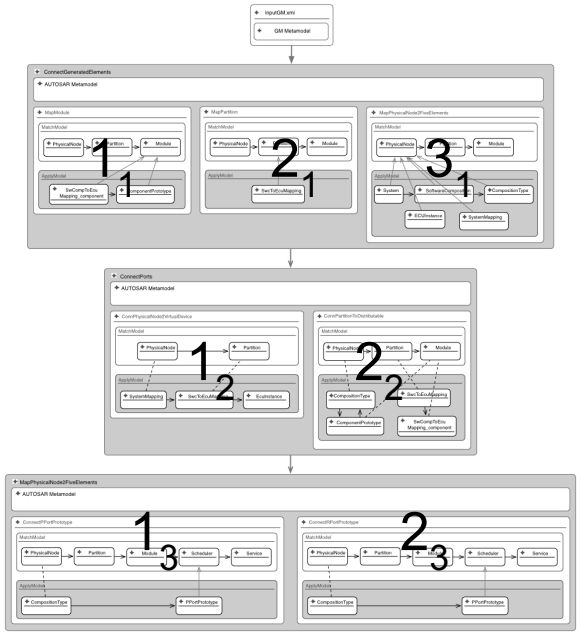
Combining a Path Condition with a Rule

Case 4: Rule has Dependencies and *May* Execute





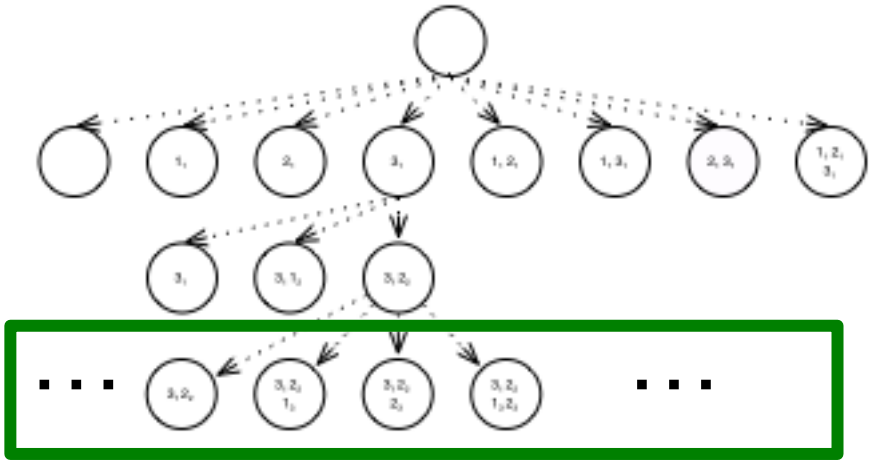
Reminder: Path Condition Generation



Process Layer 1

Process Layer 2

Process Layer 3



Path Conditions

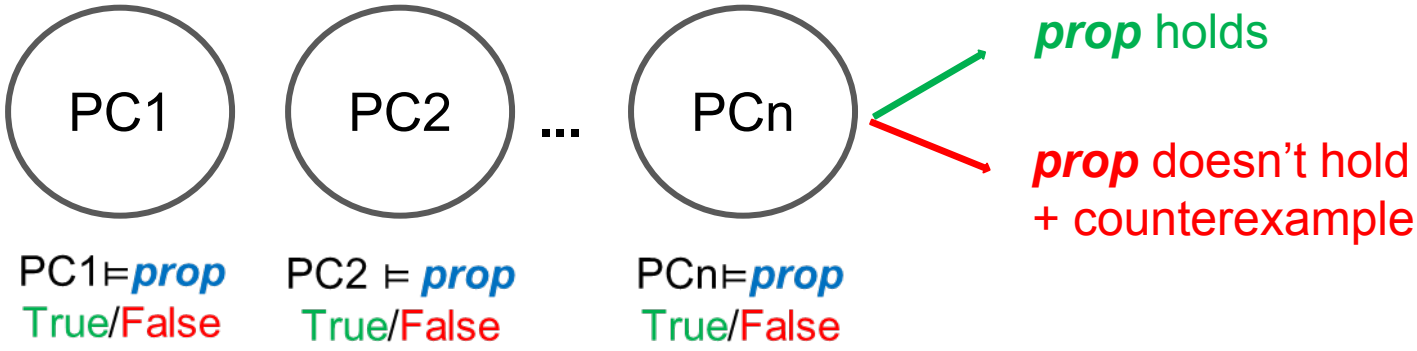
Based on: L. Lucio, B. Barroca, V. Amaral “A Technique for the Verification of Model Transformations” Proceedings of MoDELS, 2010.

Phase 2- Property Verification

Takes 2 inputs:

1. **Path conditions generated from phase 1**
2. **Property to verify**

1. Generated Path Conditions



2. Input Property ***prop***

a) *AtomicContracts: Precondition & Postcondition*

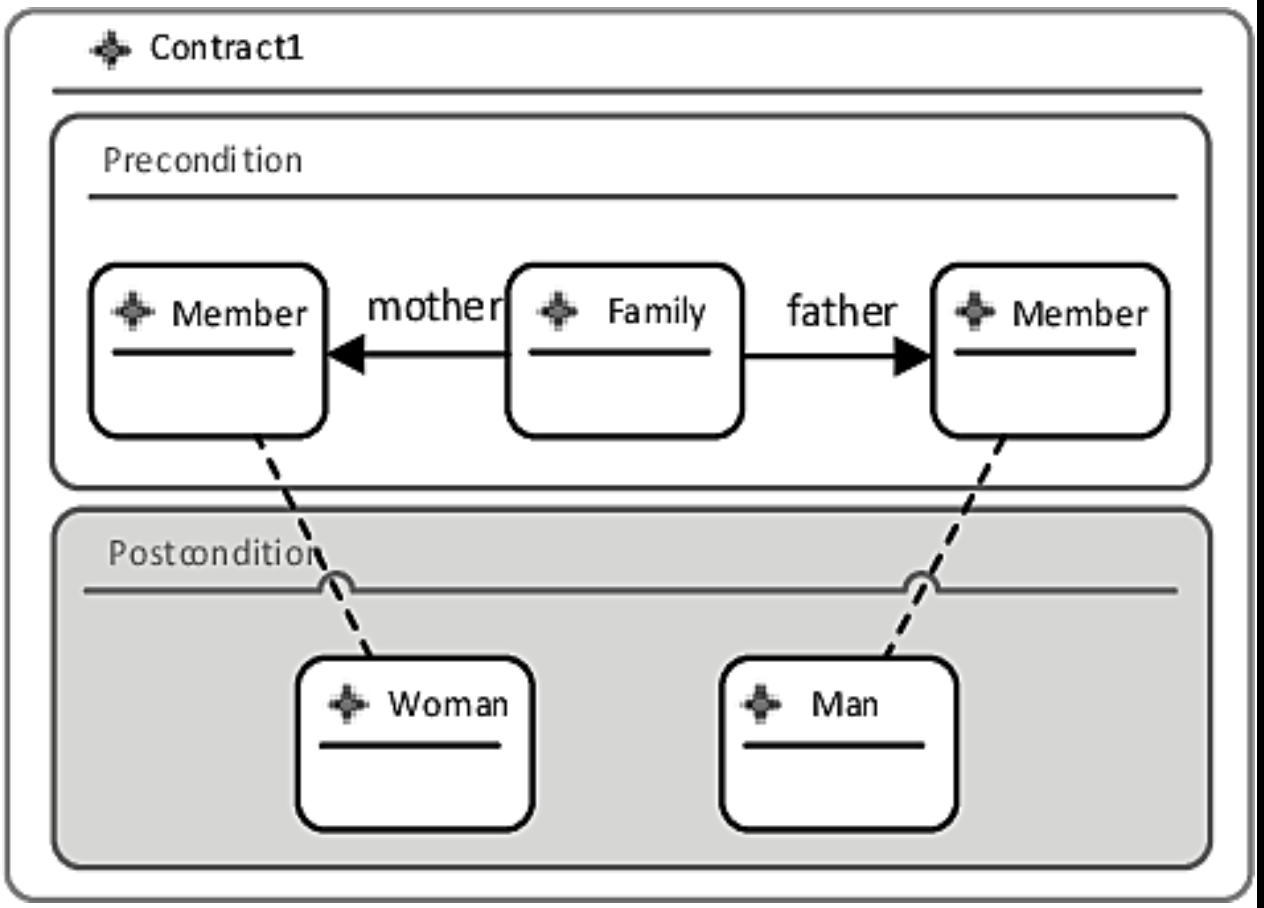
b) Propositional formulae of AtomicContracts (And, Or, Not, If/Then)

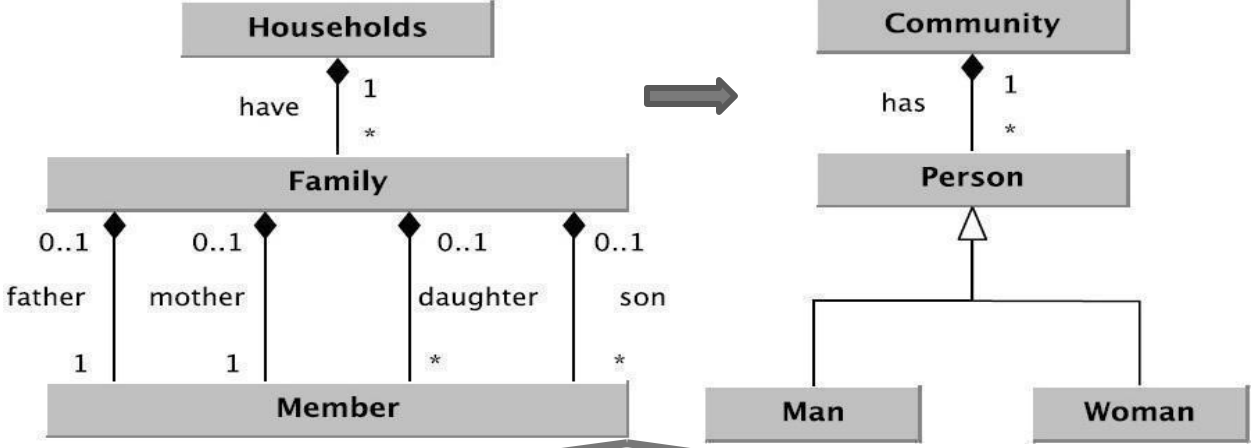
Phase 2- Property Verification

Example of AtomicContract:

If a pattern of elements exists in the input

Then another pattern of elements must exist in the output





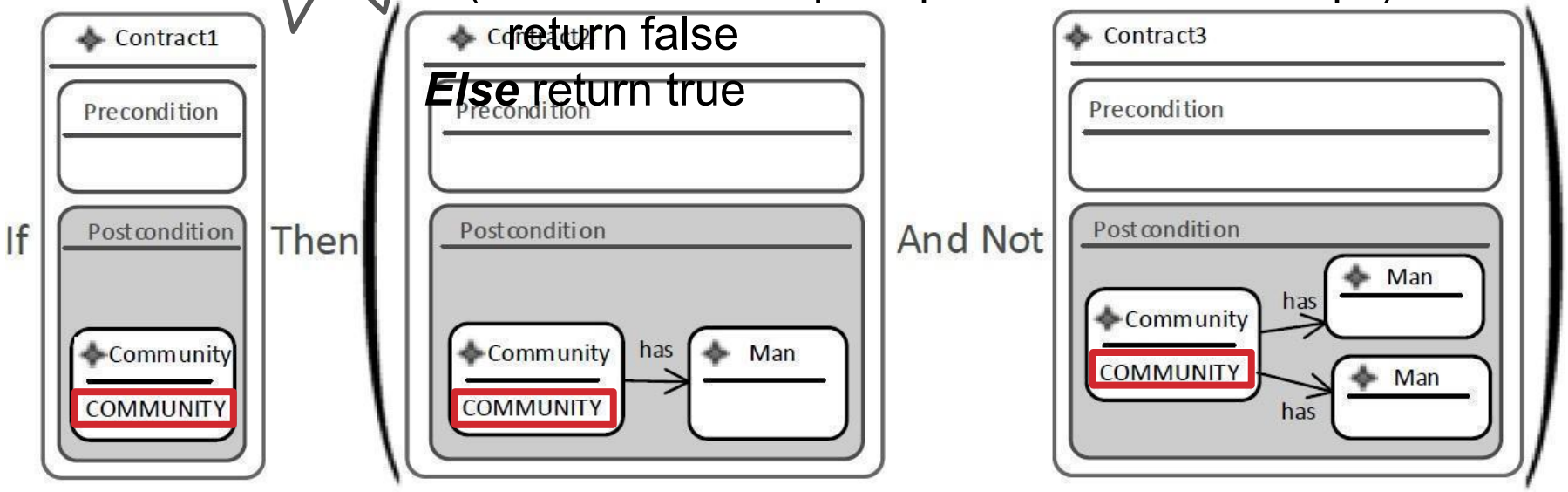
AtomicContracts
e.g.,PatternContracts

Propositional Formula of AtomicContracts
e.g., “1..1” Multiplicity Invariants

Free Variables !

Verification for a pathcondition pc using ***Subgraph Isomorphism:***
if (Precondition in pc & postcondition not in pc)
return false
Else return true

If The output has a *Community*
Then {That *Community* will be associated to one *Man* ***And Not*** More than one *Man*}



Industrial Case Study

- **GM-2-AUTOSAR migration transformation [1]**
- **GM-2-AUTOSAR Transformation Size**

DSLTrans	ATL
3 Layers, 2 or 3 rules per layer	2 matched rules, 9 functional helpers, 6 attribute helpers

- **GM-2-AUTOSAR transformation Properties [2]:**
  - ***Multiplicity Invariants:*** The transformation's output preserves the multiplicities in the output metamodel
  - ***Security Invariant:*** A physical node does not refer to a software component that is not deployed on that node.
  - ***Pattern Contracts:*** If a pattern of elements exists in the input, then a corresponding pattern must exist in the output
  - ***Uniqueness Contracts:*** An output element of a rule is uniquely named if the corresponding input element is uniquely named, too. (Not handled in our prover)

1. G. Selim, S. Wang, J. R. Cordy, J. Dingel. “Model Transformations for Migrating Legacy Models: An Industrial Case Study”. ECMFA, 2012.
2. G. Selim, F. Büttner, J. R. Cordy, J. Dingel, Shige Wang.”Automated Verification of Model Transformations in the Automotve Industry”. MODELS, 2013.

Industrial Case Study

- Time to generate path conditions (performed once) = 0.6 secs
- Time to verify properties:

	Multiplicity Invariants						Security Invariant	Pattern Contracts	
Property	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**S1**	**P1**	**P2**
Time (sec)	0.013	0.017	0.013	0.017	0.017	0.019	0.017	0.02	0.02

- Maximum time to verify a property = 0.02 sec

DISCUSSION

Compared To

Pros

- Result holds for any input; not limited to a scope
- No translation needed
- Verification is much faster using our prover ★

Cons

- Cannot prove properties that reason about attributes
- Cannot verify transformations with NACS

Property	M1	M2	M3	M4	M5	M6	S1	P1	P2
Time in our prover (sec)	.013	.017	.013	.017	.017	.019	.017	.02	.02
Time in [1] within a scope of 6 (sec)	76	73.4	75	75	75.5	74.5	114	256	251

1. G. Selim, F. Büttner, J. R. Cordy, J. Dingel, Shige Wang.”Automated Verification of Model TransfoSrmations in the Automotve Industry”. MODELS, 2013.

Conclusion & Future Work

Conclusion

- Extended an input-independent property prover
- Property prover can verify a variety of property types
- Proved soundness & completeness of property prover
- Conducted a case study
- Compared our prover with another verification tool

Future Work

- Extended scalability tests
- Handle properties that reason about attributes
- Verify transformations with NACs.