Budapest University of Technology and Economics
Department of Automation and Applied Informatics
Applied Computer Science Group

Graph Transformation: What can and what cannot be proven?

Levels of Formality

Offline Validation

- Motivation
- Backgrounds
  - Typed attributed graphs
  - Production, GTS
  - Negative Application Conditions (NACs)
  - Concurrency theorem
  - Termination
- Termination criteria
  - E-Based Composition
- Summary

Motivation-1

- Transformation tools with sophisticated control flow
  - Attribute Constraints
  - Branches
  - LHS and RHS Constraints
- Examples
  - Object-Relational Mapping

Motivation -2

Definition 2.8 A graph transformation system $GTS = (P)$ terminates if there is no infinite sequence of direct graph transformations $G_0 \Rightarrow G_1 \Rightarrow \ldots$ applying rules from P starting from any input graph G_0, with respect to the control structure of the given graph transformation system.

- Undecidable in general
- Other solutions: algorithm for special cases
  - Execution units
  - Layered GTS

Motivation -3

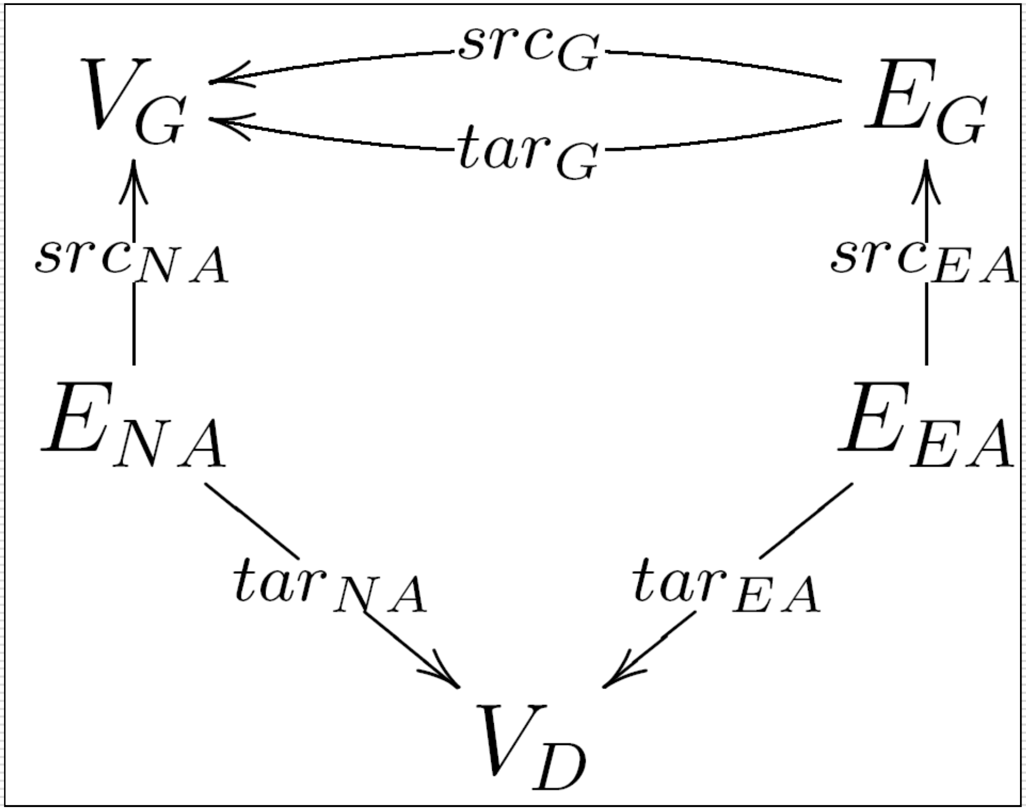
- Practical applications
  - Have special structure/characteristics in the rules
  - Have rich control and constraint features
  - Need a criterion connected with the rules MORE DIRECTLY

The termination criteria are not prepared for all the specialties and structure that may occur, and that make the transformation decidable!

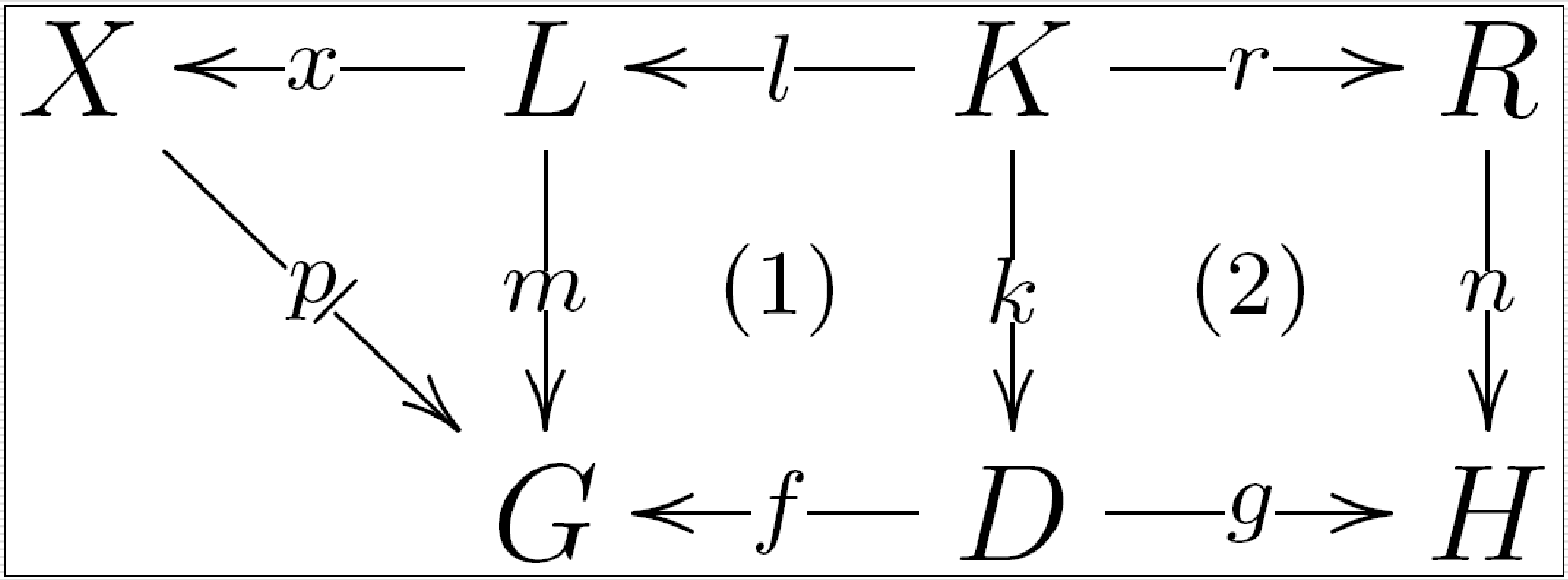
Motivation - 4

- We need a criterion that ...
  - ...is a paper and pencil method
  - ...is applicable for any constraint/attribute constructs
  - ...is equivalent to our initial termination definition in practical cases
  - ...directly corresponds to the rules (usability for engineers)

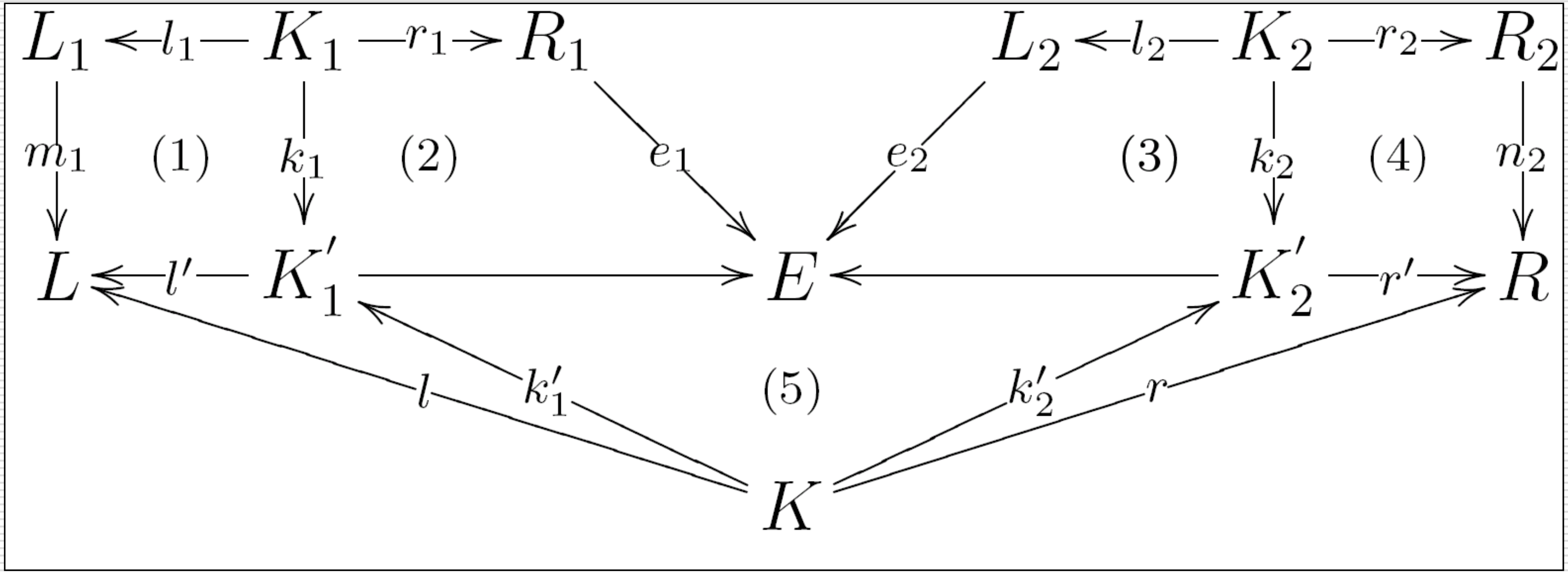
Typed attributed graphs



Productions, DGTs, and NACs



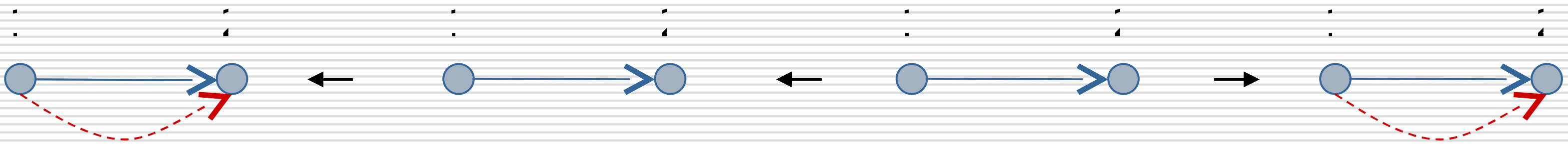
E-Concurrent Production



Transitive Closure

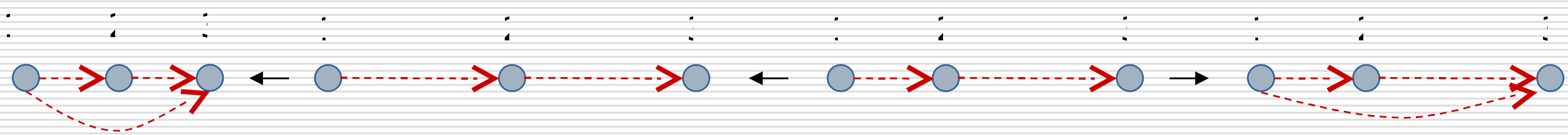
Layer 1

NAC_1 **L_1** **K_1** **R_1**



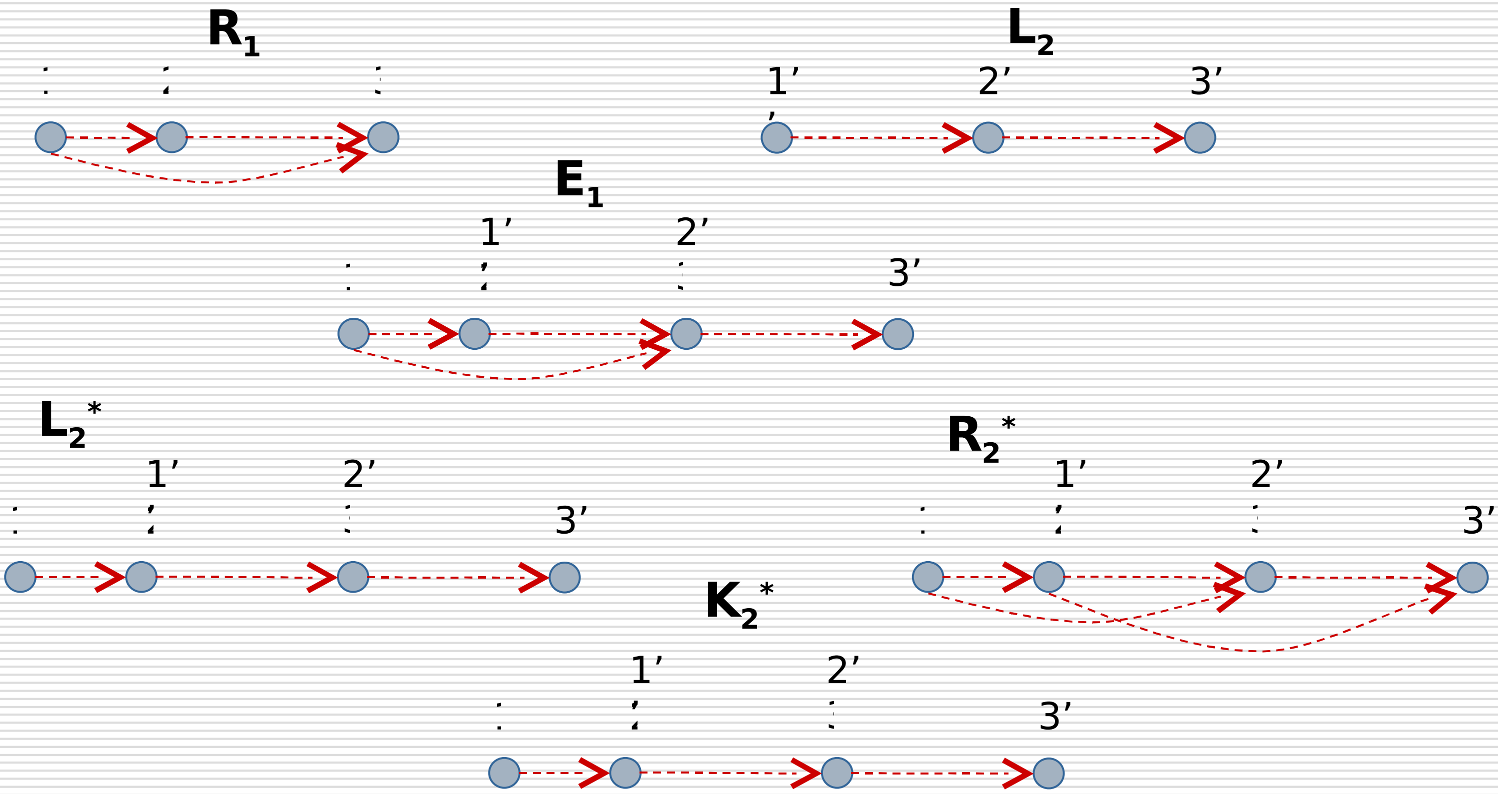
Layer 2

NAC_2 **L_2** **K_2** **R_2**



E.g.: Inheritance hierarchy (blue)

E-Concurrent Production



Cumulative LHS Series

Definition 3.2 Consider a possibly infinite sequence of graph productions p_i, $(i = 1, 2, ...)$ and a sequence of E-dependency relations $((E_i, e_i^*, e_{i+1}))$ leading to a sequence of their E-based compositions $(p_i^* = (L_i^* \leftarrow K_i^* \rightarrow R_i^*))$ with $p_1^* = p_1$ and $p_n^* = (p_1 *_{E_1} p_2) *_{E_2} ... *_{E_n} p_n$.

A cumulative LHS series of this sequence is the graph series L_n^* consisting of the left-hand side graphs of p_n^*. Moreover, a cumulative size series of a production sequence is the nonnegative integer series $|L_n^*|$.

Termination

Theorem *A GTS $= (P)$ terminates if for all infinite cumulative LHS sequences (L_i^*) of the graph productions created from the members of P, it holds that*

$$\lim_{i \to \infty} |L_i^*| = \infty.$$

Note that we assume finite input graphs and injective matches.

Theorem *If a GTS $= (P)$ terminates and we have only a finite number of input graphs up to isomorphism, then there are no infinite cumulative LHS sequences (L_i^*) of graph productions created from the members of P.*

Lemma *If $L_i^* \not\cong L_{i+1}^*, \forall i$ for every cumulative LHS series, then the GTS terminates. If each graph appears only finitely many times in all cumulative LHS series, the GTS still terminates.*

Summary

- Existing approaches
  - algorithms that can be applied in tools
  - work for non-injective matches
- The concurrency theorem-based approach
  - can treat infinite rule sequences
  - is not an algorithm, cannot be applied in tools, but patterns can be recognized
  - works for injective matches only
  - best for systems with rich control structure and complex constraints
- Extending to constraints

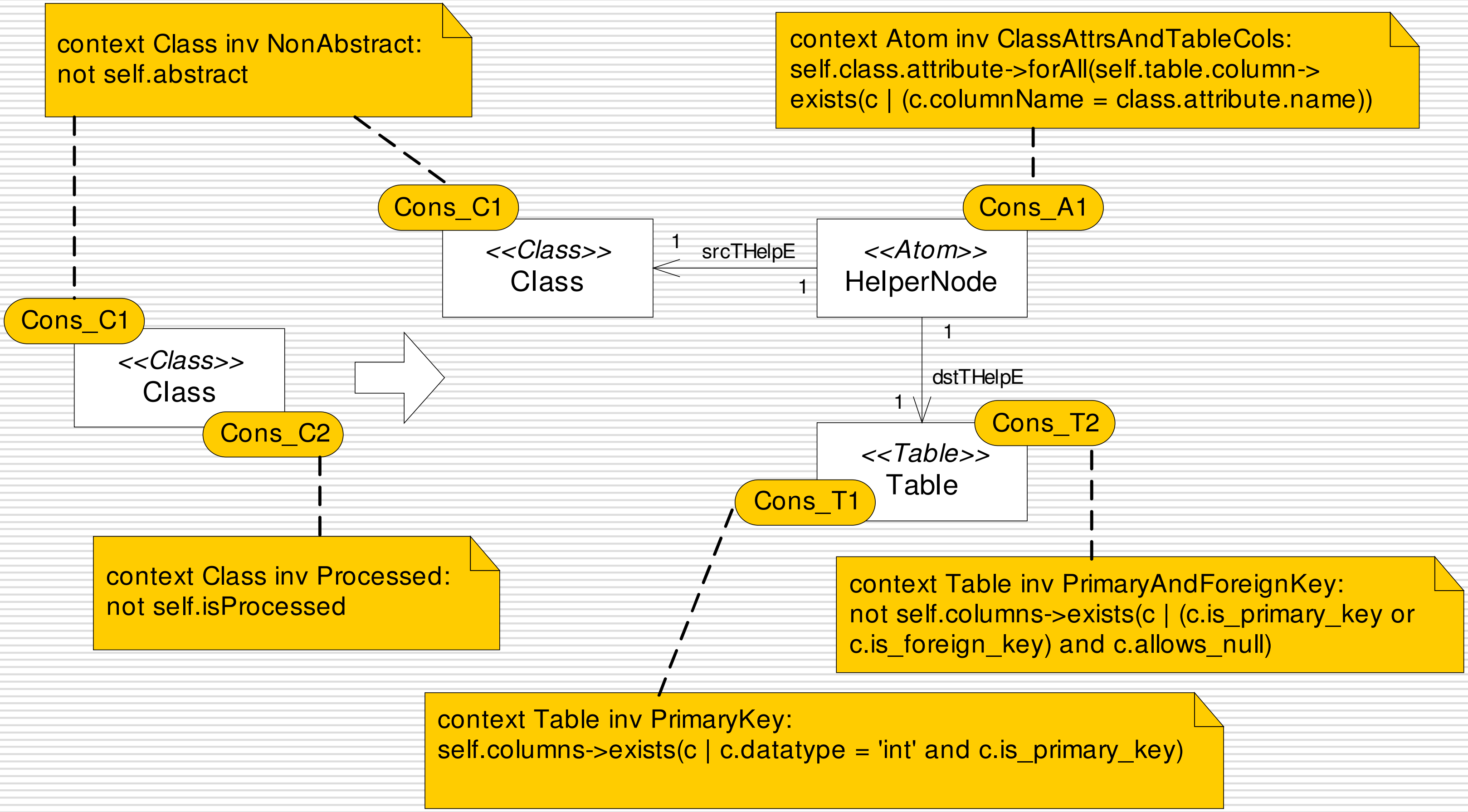
Online Validation

- Requirements as input
- Formalizing with semantically motivated constraints to the rules
- Algorithmically feasible

Online Validation

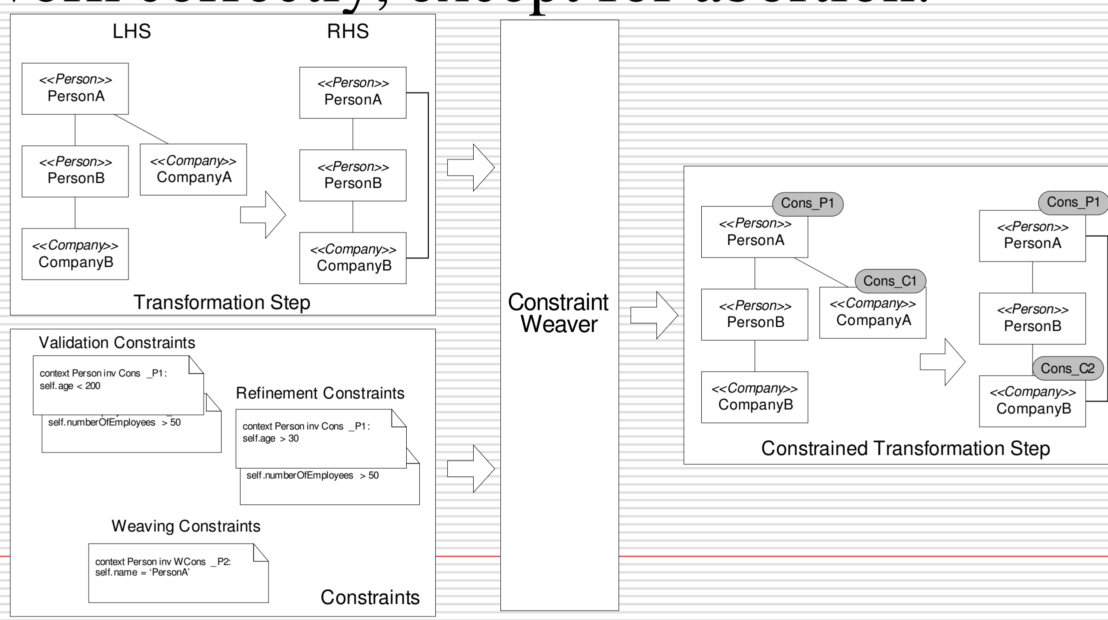
- Requirements
  - Each table has primary key,
  - Each class attribute is part of a table,
  - Each parent class attribute is part of a table created for its inherited class,
  - Each many-to-many association has a distinct table,
  - Each one-to-many and one-to-one association has merged into the appropriate tables,
  - Foreign keys not allow NULL value, and
  - Each association class attribute buried into the appropriate table based on the multiplicities of its association.

Online Validated Model Transformation



Separating Constraints in Validated Model Transformation

- A *refining constraint* complete the conditions required by the structure of LHS of a transformation step.
- A *validation constraint* expresses a semantically motivated constraint without which the transformation would work correctly, except for abortion.



On Formal Semantics

- Mapping to a mathematical domain
  - Mostly executed only on human brains
  - Needs a purpose:
    - complexity depends on abstraction level and
    - data representation
  - Encoding into mathematical domain can be error prone: formality does not save us
    - Validation vs. verification

Summary

- Mostly offline validation methods are not feasible
- Online validation tests the actual model only
- Combined solutions
- Errors are possible in both methods: formality (i.e. mathematics) helps, but is not the key – engineering solution is needed with tests

Thanks for your Attention!

Any Questions?