

# Aspect-Oriented Constraint Management

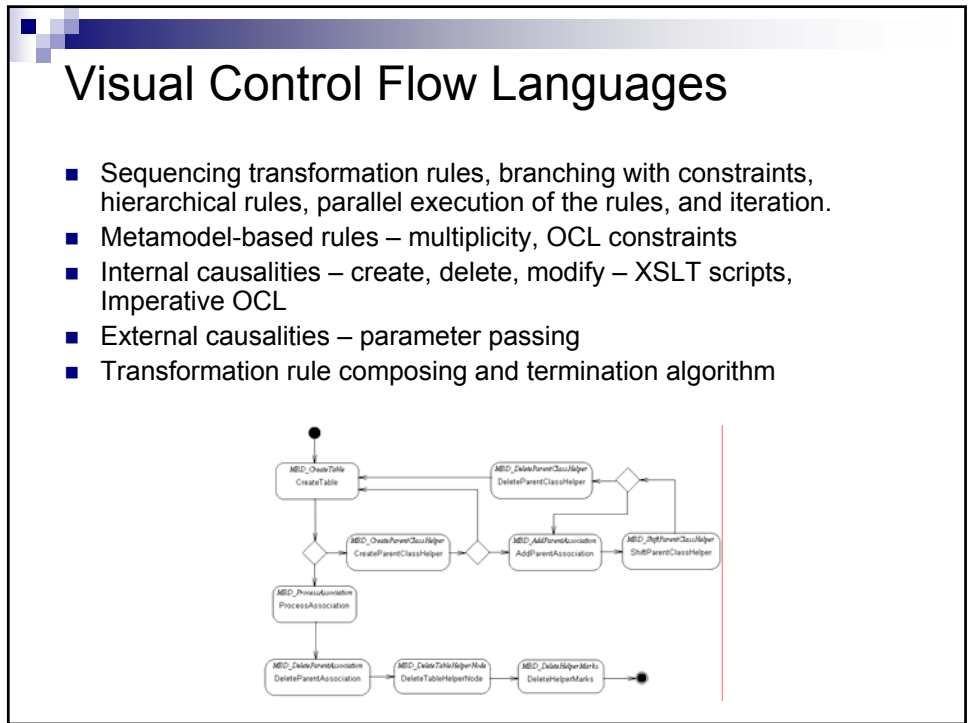
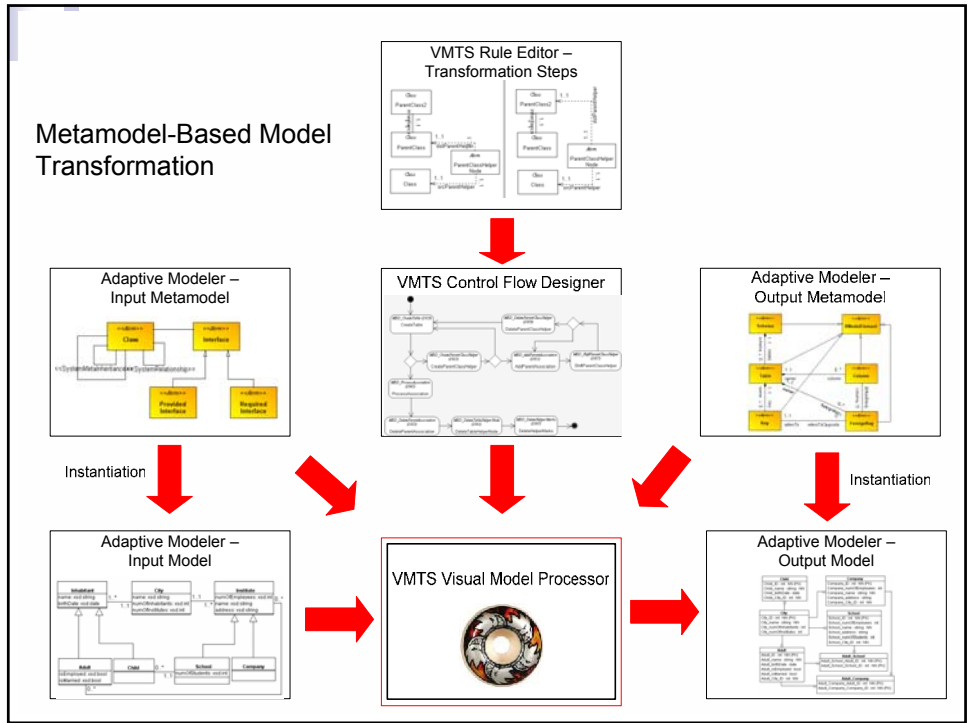
László Lengyel

<http://www.vmts.aut.bme.hu>

## Outline

- Backgrounds
- Metamodel-based model transformation
- Graph rewriting, Metamodel-based rewriting
- Validated model transformation
- Crosscutting constraints
- Aspect-oriented software development
- Aspect-oriented constraint management
- Optimized constraint handling and validation
- Constraint separation
- Constraint normalization
- Summary





## Validated Model Transformation

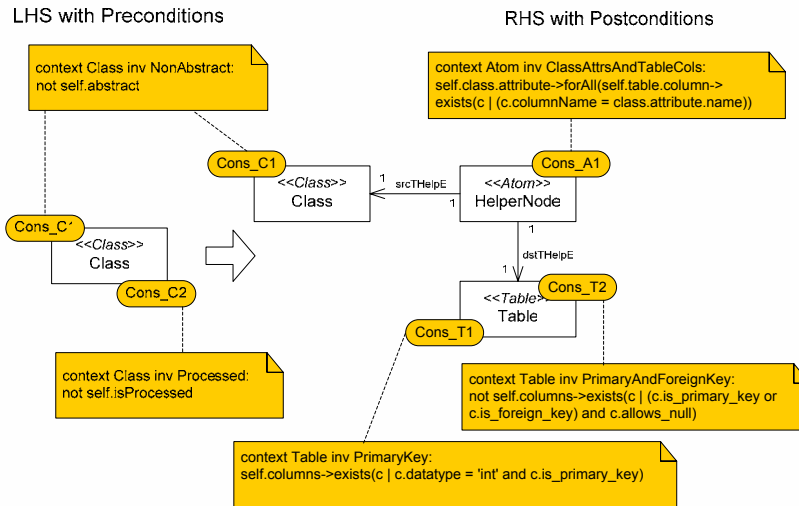
### ■ Motivation

- At the implementation level, system validation can be achieved by testing. There is no real possibility that the testing covers all the possible cases.
- In case of model transformation environments, it is not enough to validate that the transformation engine itself works as it is expected. The transformation specification should also be validated.
- There are several transformation environment, but only few of them supports some (offline) validation method.
- There is a need for a solution that can validate model transformation specifications: online validated model transformation that guarantees if the transformation finishes successfully, the generated output is valid, and it is in accordance with the requirements.

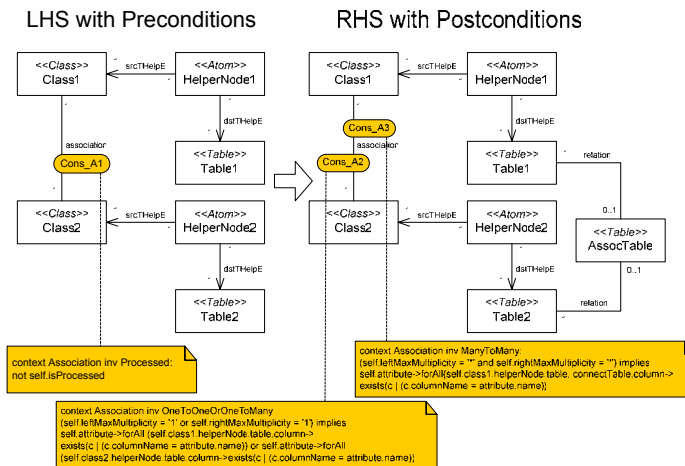
## Validated Model Transformation /2

- A precondition assigned to a transformation step is a boolean expression that must be true at the moment when the transformation step is fired.
- A postcondition assigned to a transformation step is a boolean expression that must be true after the completion of a transformation step.
- If a precondition of a transformation step is not true then the transformation step fails without being fired.
- If a postcondition of a transformation step is not true after the execution of the transformation step then the transformation step fails.
- A direct corollary of this is that an OCL expression in LHS is a precondition to the transformation step, and an OCL expression in RHS is a postcondition to the transformation step.
- A transformation step can be fired if and only if all conditions enlisted in LHS are true. Also, if a transformation step finished successfully then all conditions enlisted in RHS must be true.
- A model transformation is validated if satisfies a set of high-level constructions (e.g. validation, preservation, guarantee type constructs).
- Successful execution of the step guarantees that the output model fulfills the conditions required by high-level constructs.

## Defining Transformation Steps with Constraints



## Defining Transformation Steps with Constraints /2



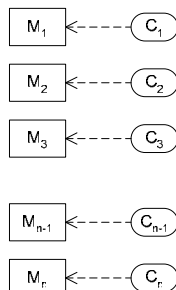
## Motivation

- Transformation consists of several rules, many times not only a transformation rule but a whole transformation is required to validate, preserve or guarantee a certain property
- The same constraint appears numerous times in the transformation → crosscuts the transformation
- Modification and deletion of the constraints is not consistent
- It is difficult to maintain and reason about the effects and purpose of constraints when they are scattered
- We need a mechanism to separate these concerns
- After we separated the constraints from the pattern rule nodes we need a weaver method

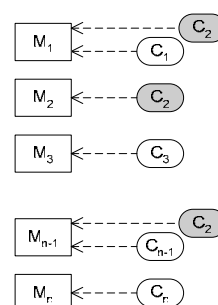
## Aspect-Oriented Software Development

- Separation of concerns
- Concerns
- Modular decomposition
- Crosscutting concerns

(a) Mapping concerns to modules

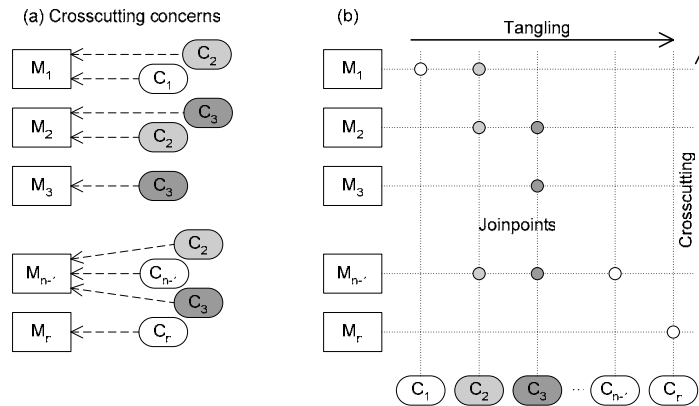


(b) Crosscutting concerns



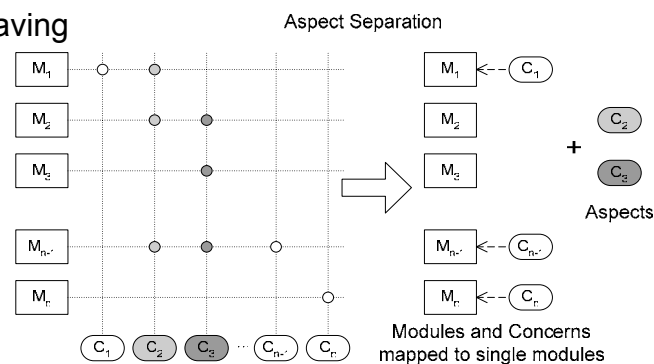
## Aspect-Oriented Software Development – Multiple Crosscutting Concerns

- Tangled Concerns
- Join points

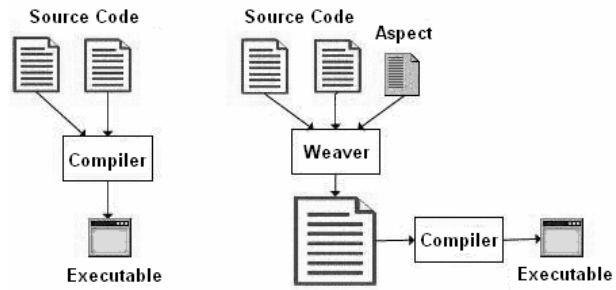


## Aspect-Oriented Software Development – Aspect Decomposition and Weaving

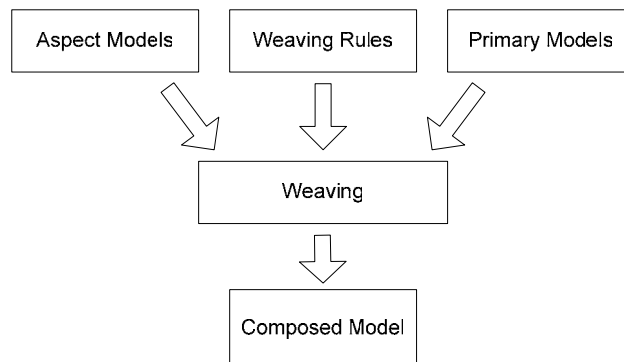
- Aspect
- Pointcut specification
- Advice
- Weaving



## Aspect-Oriented Programming



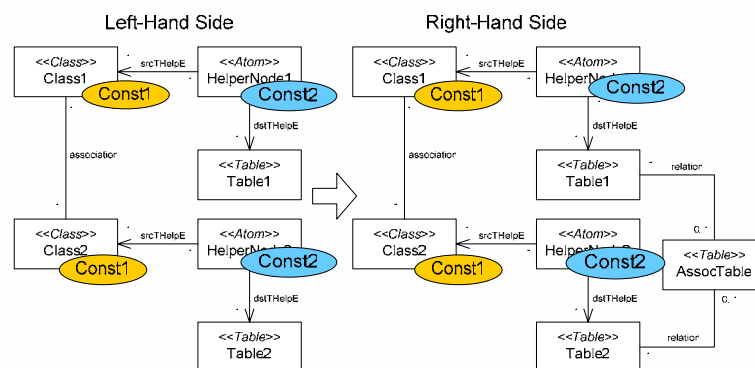
## Aspect-Oriented Modeling



## Aspect-Oriented Constraint Management Overview

- Motivation
  - Transformation consists of several rules, many times not only a transformation rule but a whole transformation is required to validate, preserve or guarantee a certain property
  - The same constraint appears numerous times in the transformation → crosscuts the transformation
- Aspect-oriented constraint management
  - Aspect-oriented constraints
  - Constraint aspects
  - Weaver algorithms
- Results:
  - Consistent constraint management
  - Reusable constraints and transformation rules
  - Weaving algorithms facilitates to require from not only individual rules, but from whole transformations to validate, preserve or guaranty certain properties.

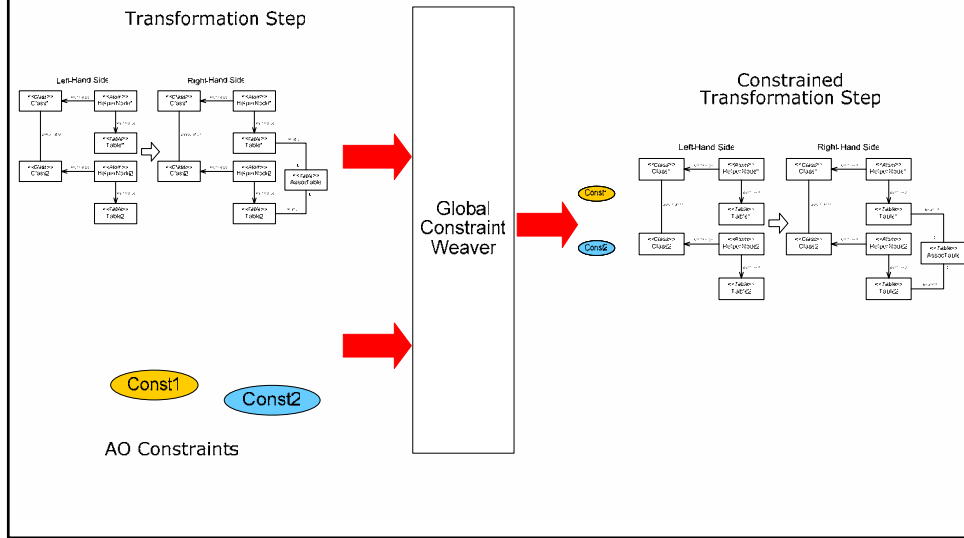
## Removing Crosscutting Constraints



```
context Atom inv Const2:
self.class.attribute->forAll(self.table.column->
exists(c | (c.columnName =
class.attribute.name)))
```

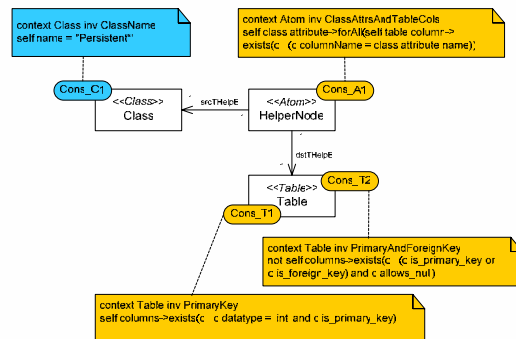
```
context Class inv Const1:
not Abstract
```

# Global Constraint Weaver

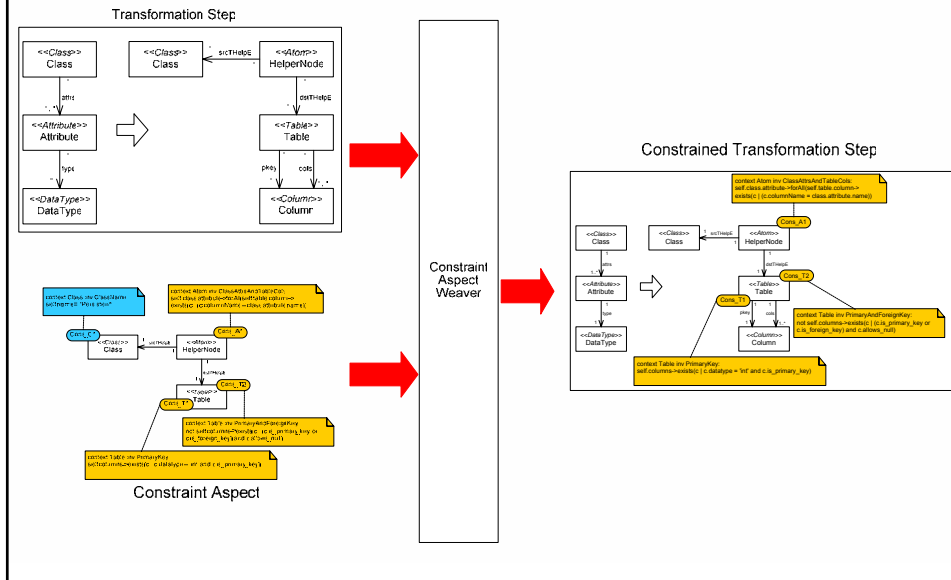


# Constraint Aspect

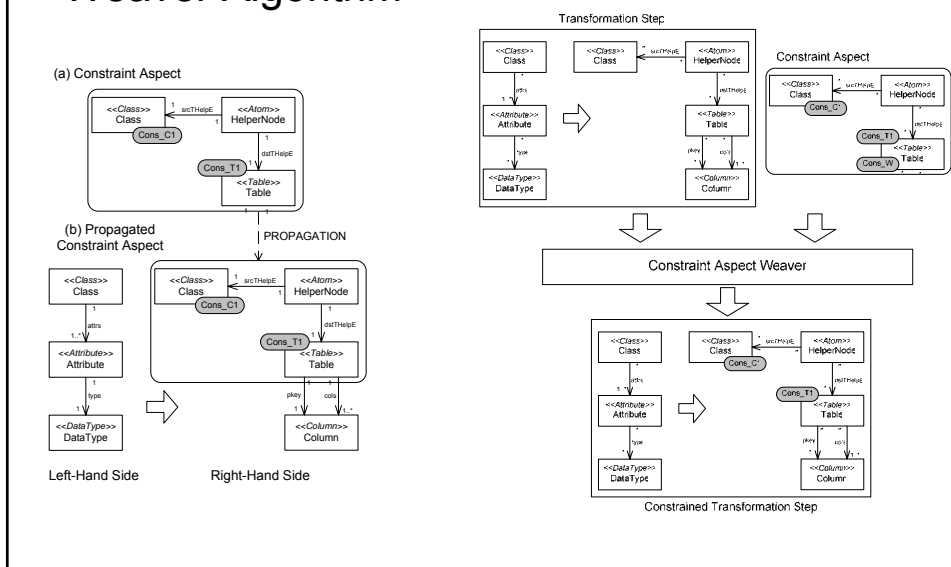
- A constraint aspect is a pattern (structure) built from metamodel elements to which OCL constraints are assigned. A constraint aspect contains not only textual conditions described by the OCL constraints but it has:
  - Structure, type and multiplicity conditions,
  - OCL constraints, and
  - Weaving constraints.



# Constraint Aspect Weaver

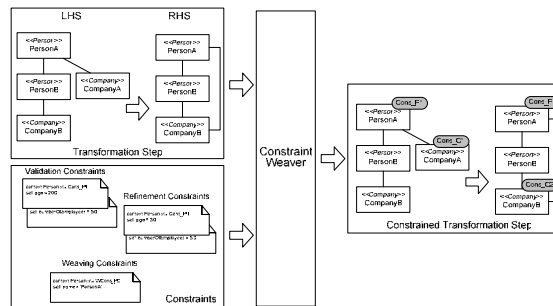


# Constraint Aspect and the Constraint Aspect Weaver Algorithm



## 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.



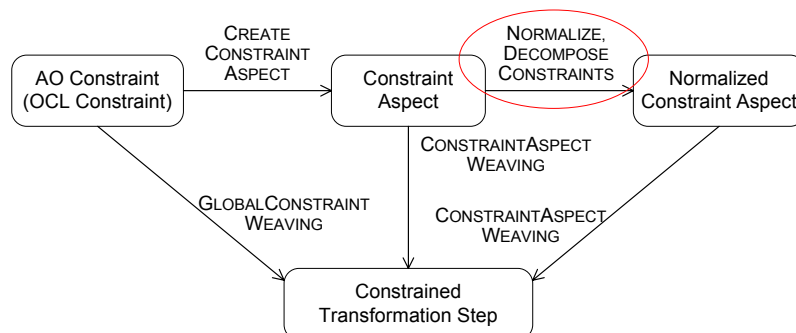
## Aspect-Oriented Constraint Management Summary

- We have found that the source of our transformation problems (maintainability, reusability) is often related to the lack of support for modularizing crosscutting concerns.
- Constraints are defined separately from transformation rules, and they are propagated to the pattern rule nodes.
- The same constraint does not appear repetitiously in many different places: consistent constraint management.
- Aspect-oriented constraint management makes the transformation rules and the constraints reusable:
  - Transformation rules can be executed with a different set of constraints, and
  - Constraints can be propagated to different transformation rules.

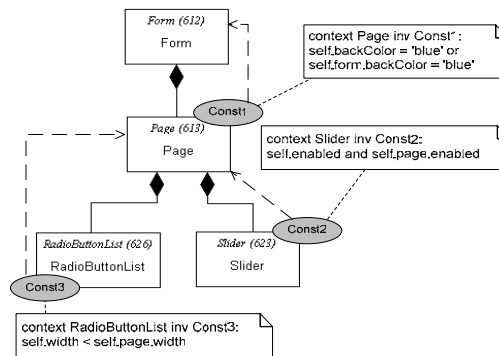
## Constrain Normalization in Rewriting Rules / UML Class Diagrams

- Motivation - Evaluating a constraint that contains navigations requires more computational complexity than a constraint without any navigation steps.
- Our solution: constraint normalization
  - Constraint decomposition and relocation
  - AND/OR Clauses

## Overview of Different Aspect-Oriented Constraint Notions

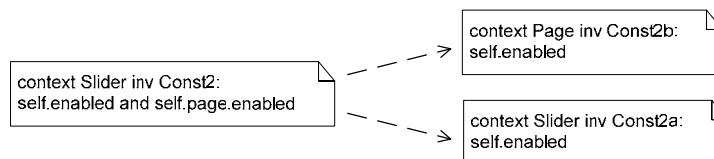


## Example LHS/UML Class Diagram with Constraints



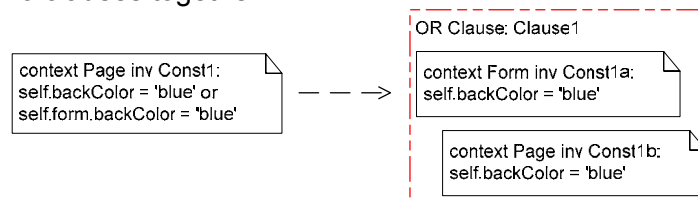
## Constraint Decomposition and Relocation

- If the sub-terms of a constraint are connected only with **and** operations, then it is enough to create new constraint for each sub-term, set the context information for them, propagate this new constraint to the model, and finally remove the original constraint from the model.

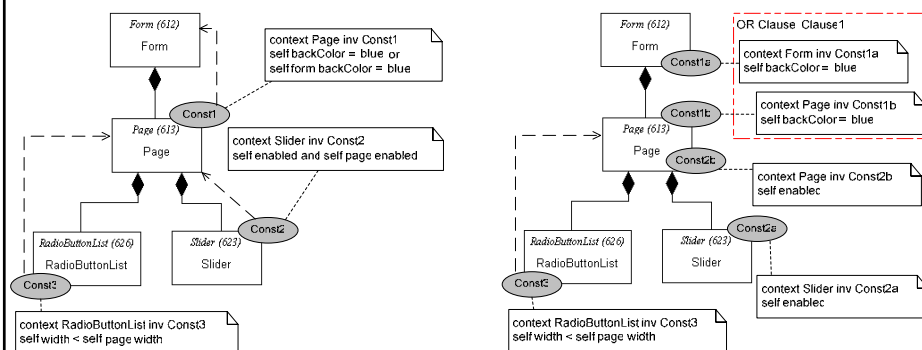


## AND/OR Clauses

- During the evaluation of constraints with **or/xor** it is not required that all the sub-terms be true
- In the case of separately propagated sub-terms all condition must be true to make the whole expression true → AND/OR Clauses
- An additional reference is maintained in order to check the two clauses together

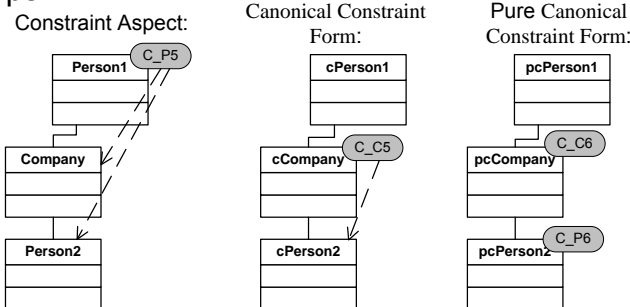


## Example for AND/OR Clauses



## Normalized Constraint Aspect

- The *Canonical Constraint Form* of a constraint aspect is the form which contains the fewest possible navigation steps.
- The *Pure Canonical Constraint Form* of a constraint aspect is the form which does not contain navigation steps



## Summary

- The metamodel-based specification of the rewriting rules allows assigning OCL constraints to pattern rule nodes
- Optimized constraint handling and validation
  - Aspect-oriented constraint management (consistency, reuse)
  - More efficient constraint propagation and validation methods (Constraint Aspects)
- Validated model transformation
  - Preconditions, postconditions – OCL constraints enlisted in model transformation steps
  - Using the weaving algorithms we can require from not only individual steps, but from whole transformations to validate, preserve or guaranty certain properties



## Summary /2

- **Constraint separation**
  - This method facilitates to reduce the complexity of validation type constraints that makes them understandable and working with them becomes easier.
- **Constraint Normalization**
  - Constraint decomposition and relocation
  - AND/OR Clauses