

Triple **G**raph **G**rammars in a Nutshell

Felix Klar & Andy Schürr

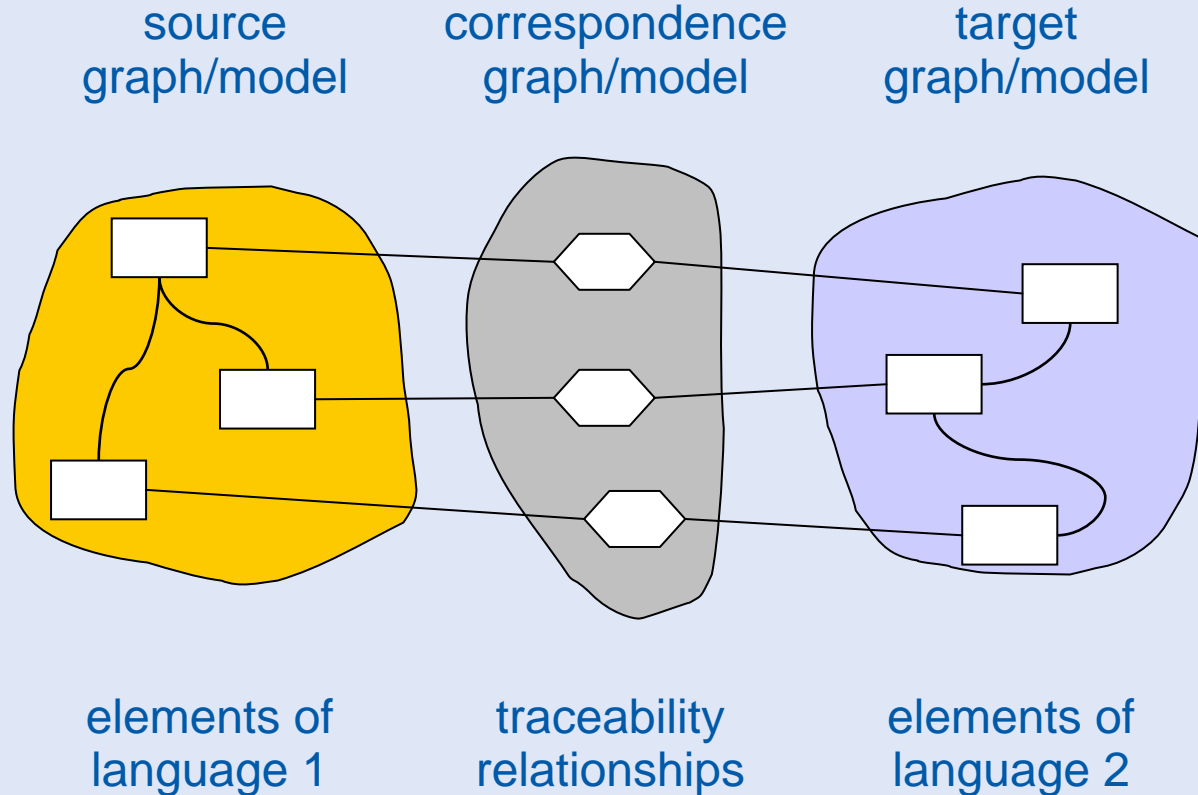
Technische Universität Darmstadt

[felix.klar | andy.schuerr] @es.tu-darmstadt.de



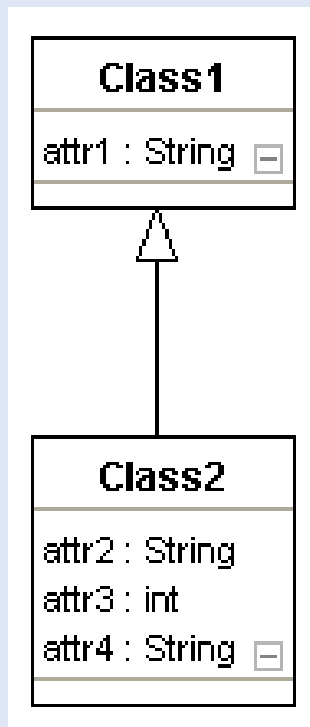


Transformation / integration of modeling languages
(via bidirectional, declarative model transformation language)





Bidirectional transformation of class diagrams \leftrightarrow RDBMS:



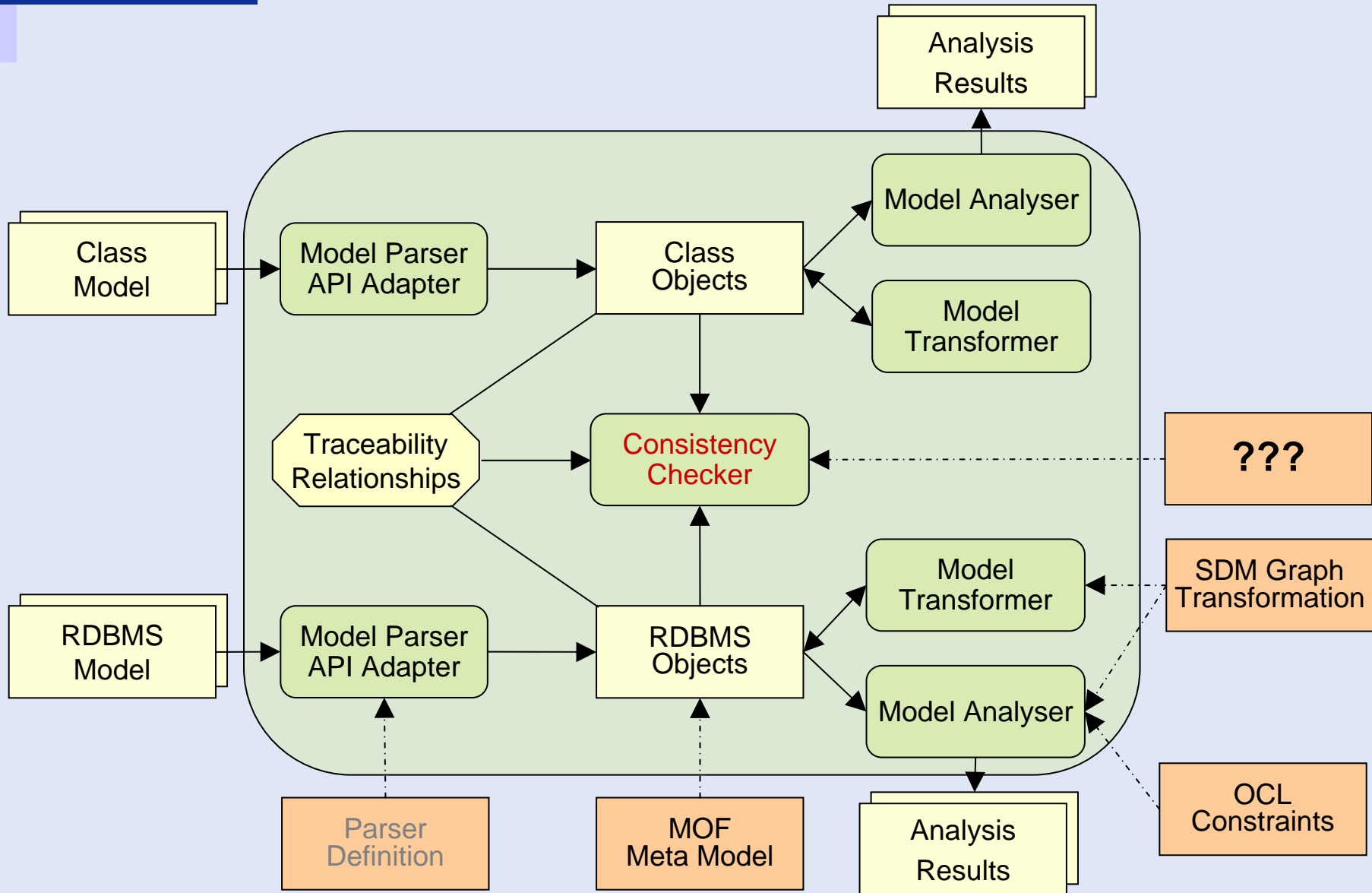
Server: localhost Database: icgt2008 Table: class1

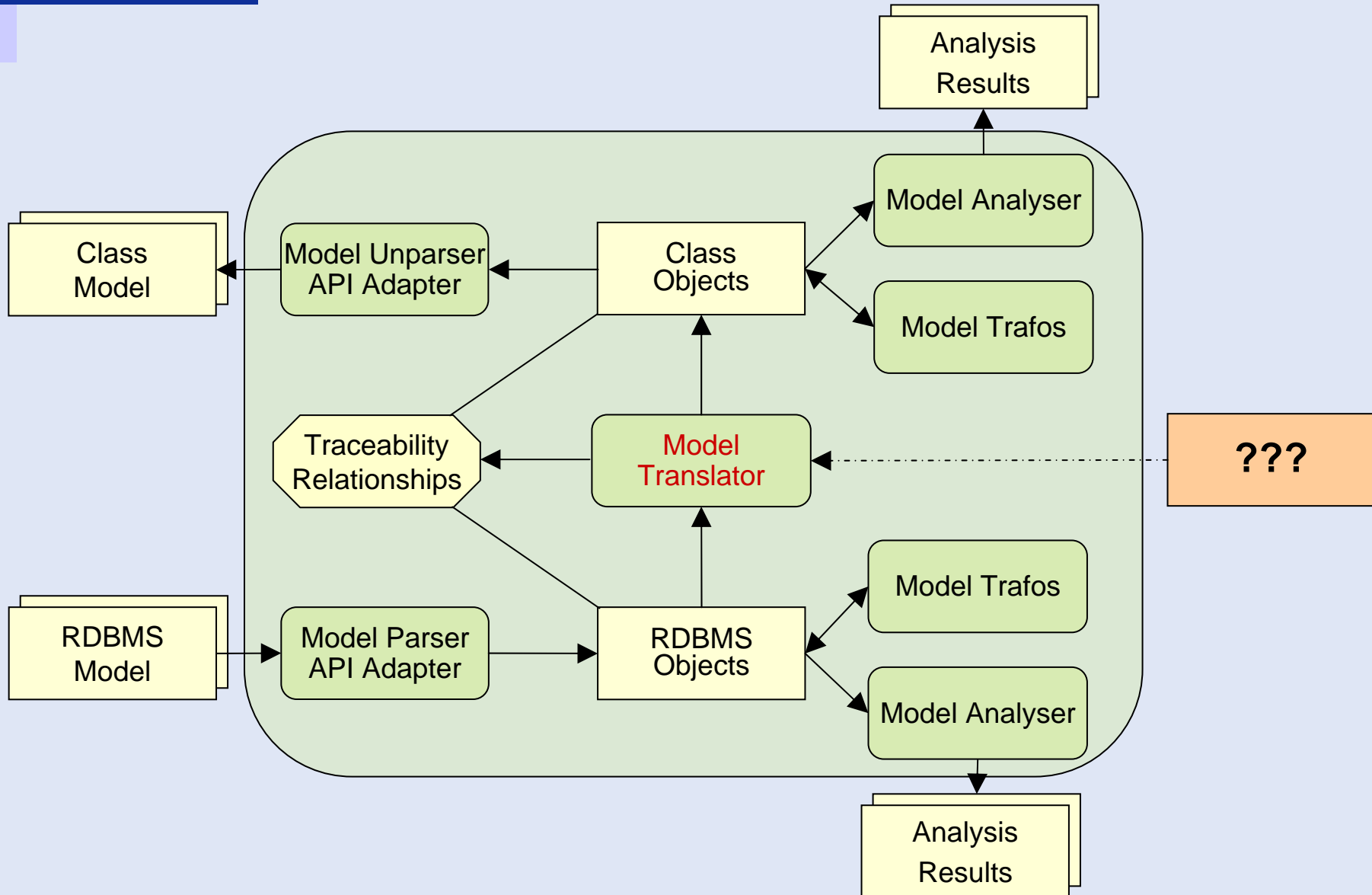
Browse Structure SQL Search Insert

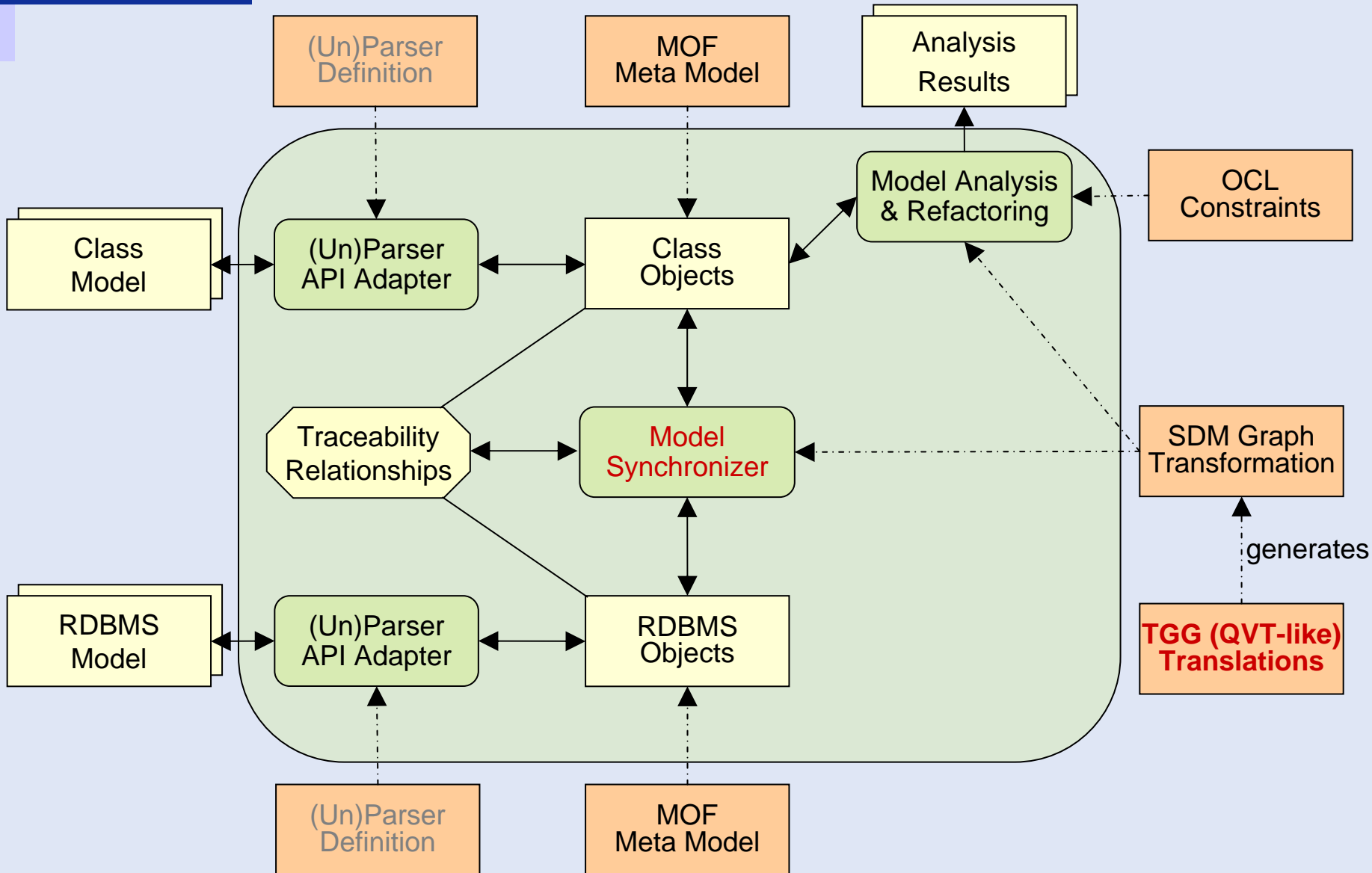
	Field	Type	Collation	Attributes	Null
<input type="checkbox"/>	attr1	varchar(1024)	latin1_general_ci		No
<input type="checkbox"/>	attr2	varchar(1024)	latin1_general_ci		No
<input type="checkbox"/>	attr3	int(11)			No
<input type="checkbox"/>	attr4	varchar(1024)	latin1_general_ci		No

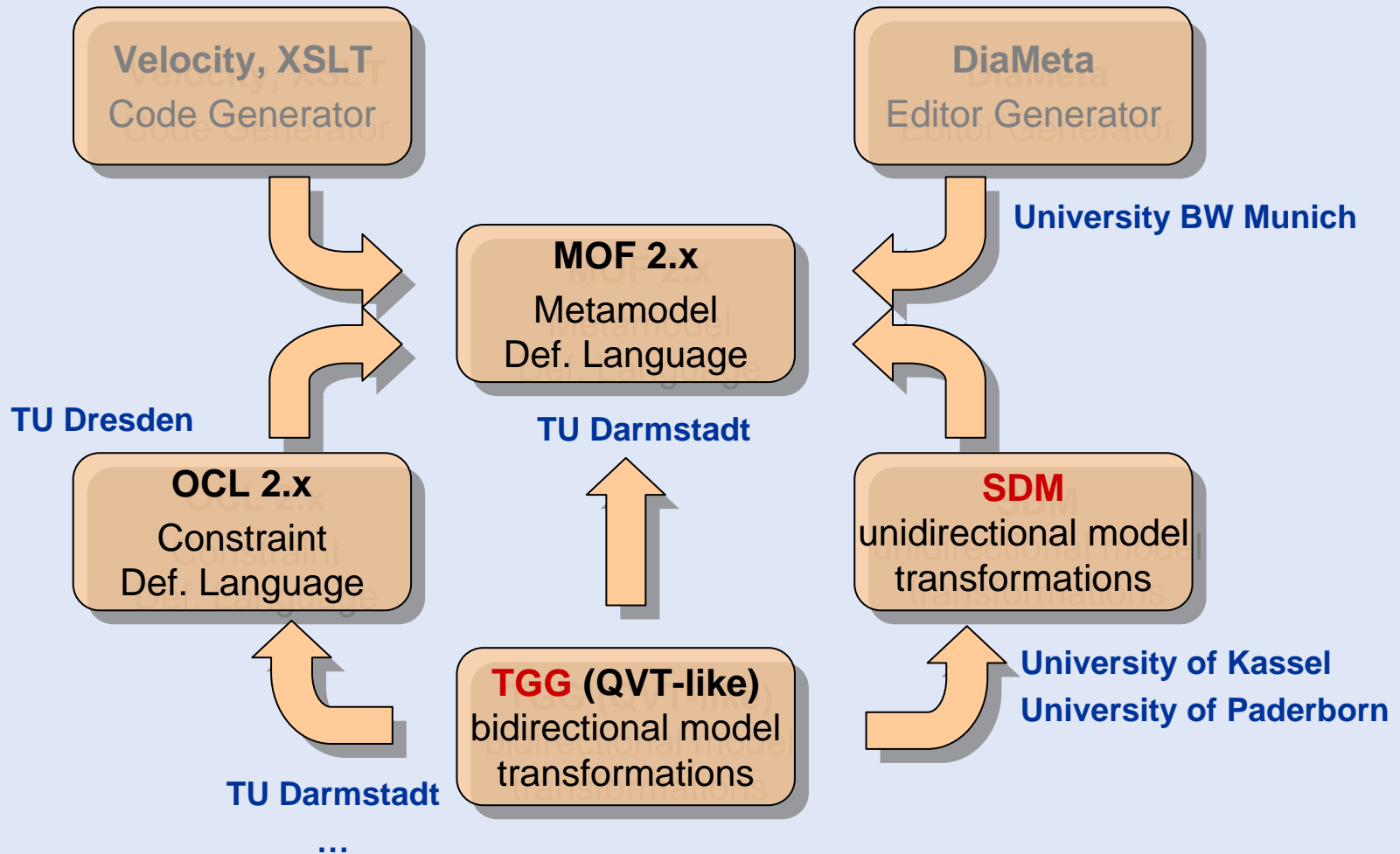
language 1,
e.g. Class Diagrams

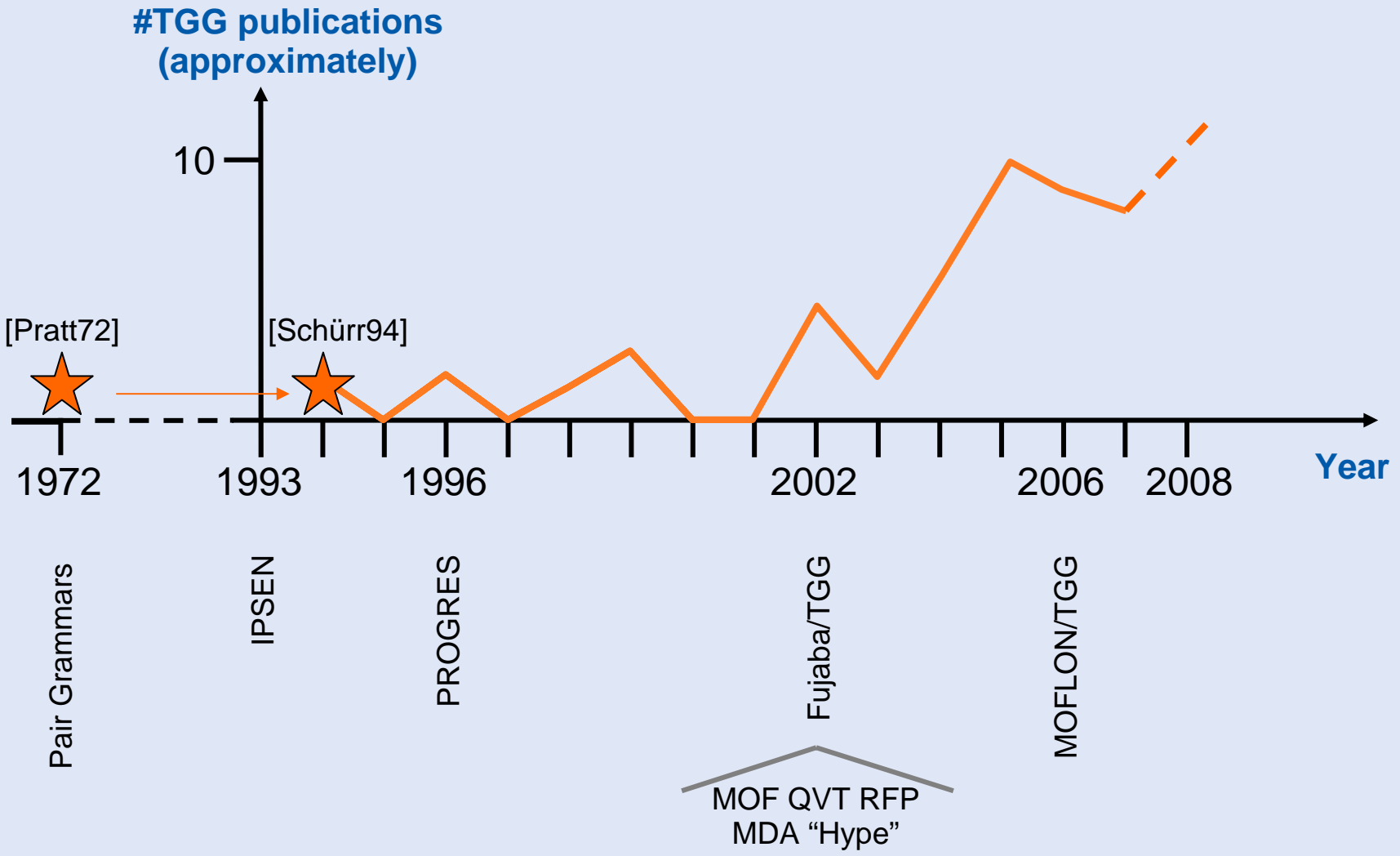
language 2,
e.g. Database Schemata

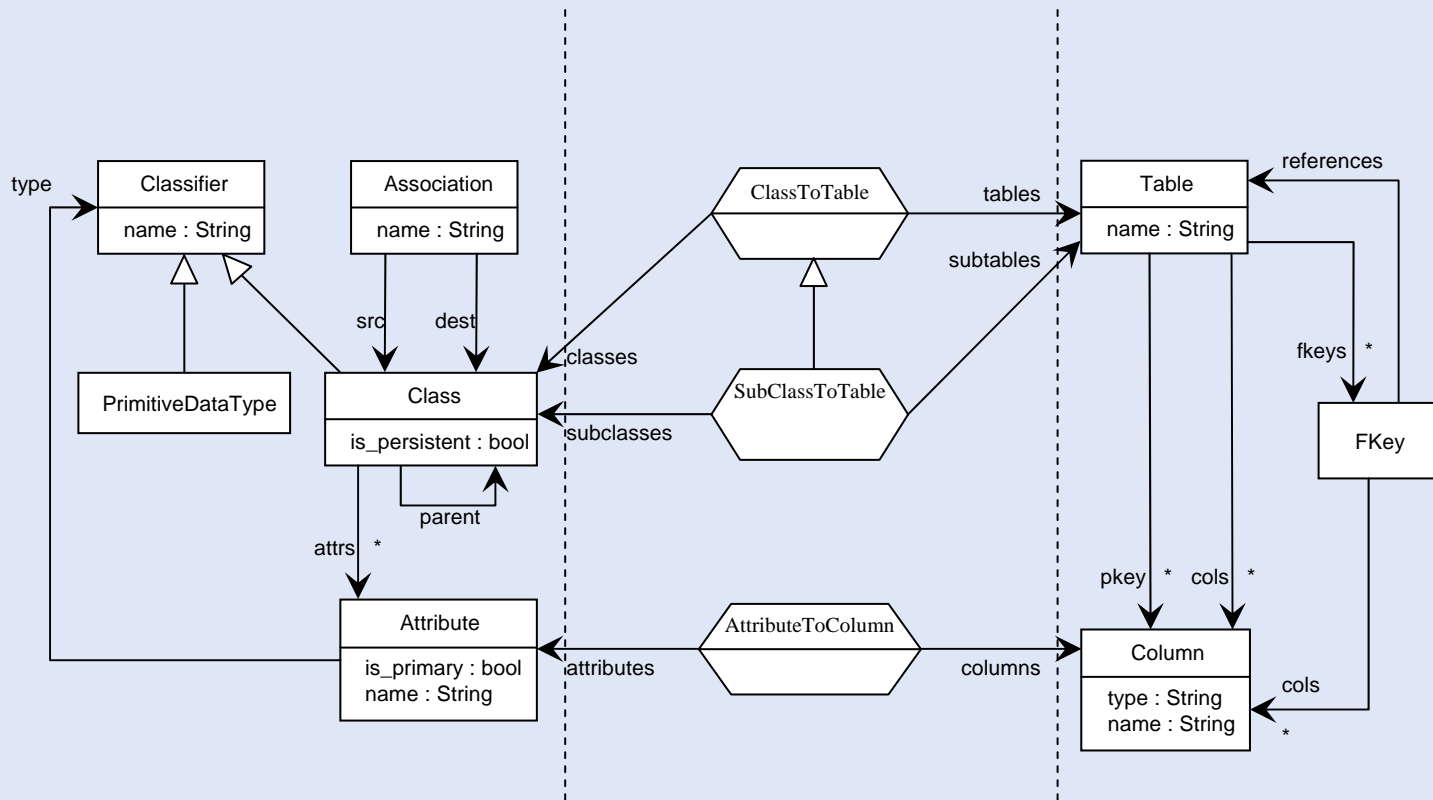








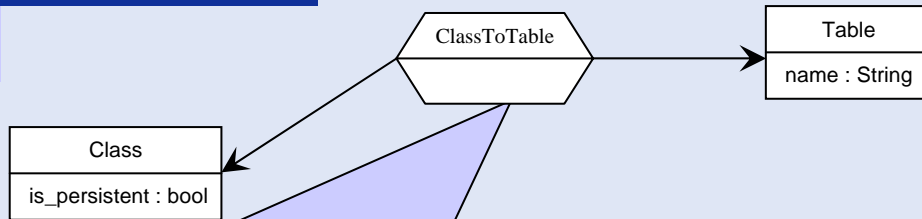




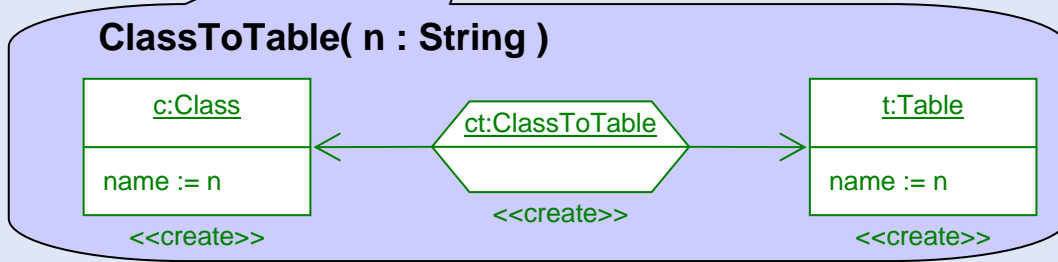
source graph schema
(source meta model)

corr. graph schema
corr. meta model

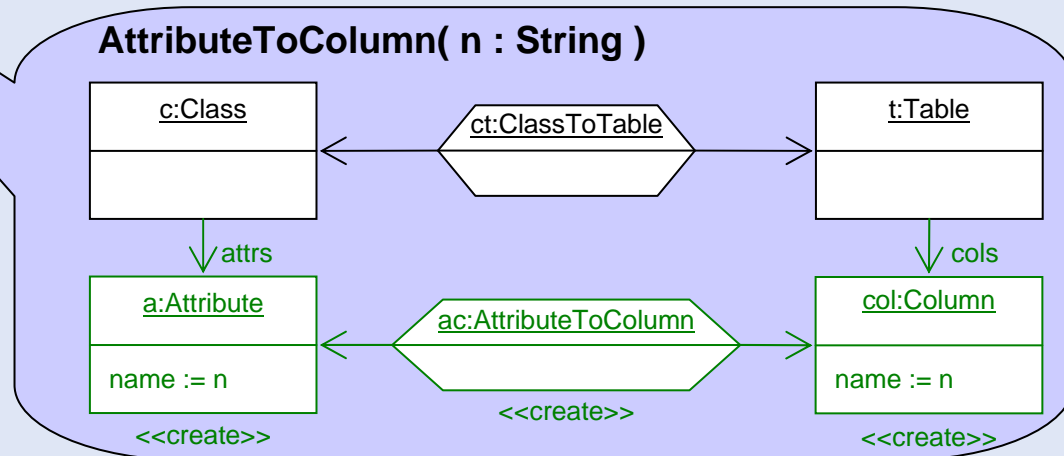
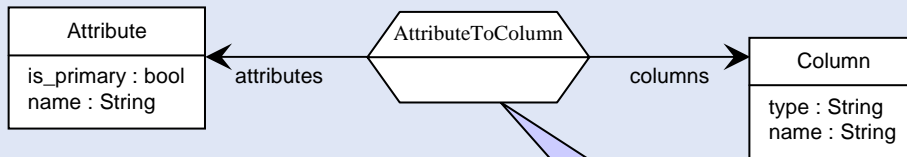
„target“ graph schema
„target“ meta model)



each correspondence link type owns a TGG rule

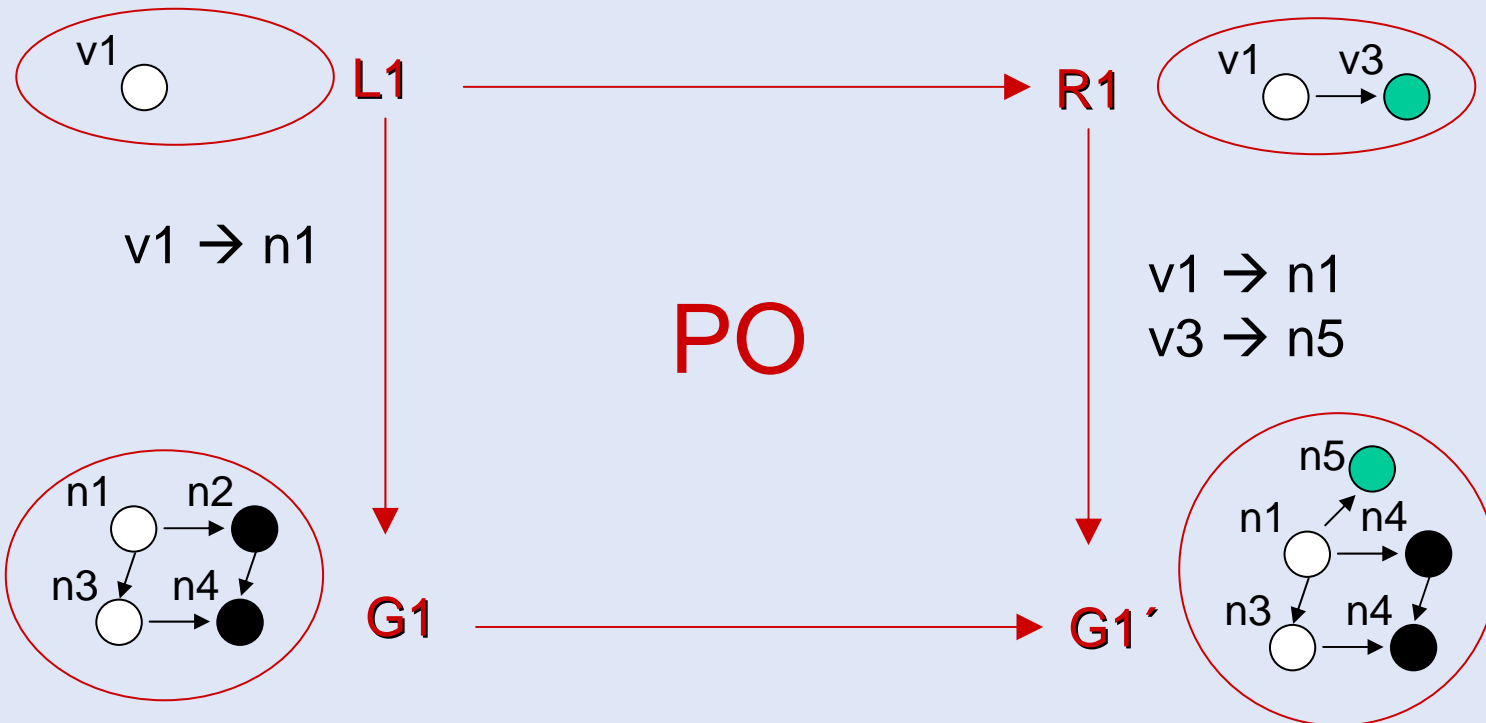


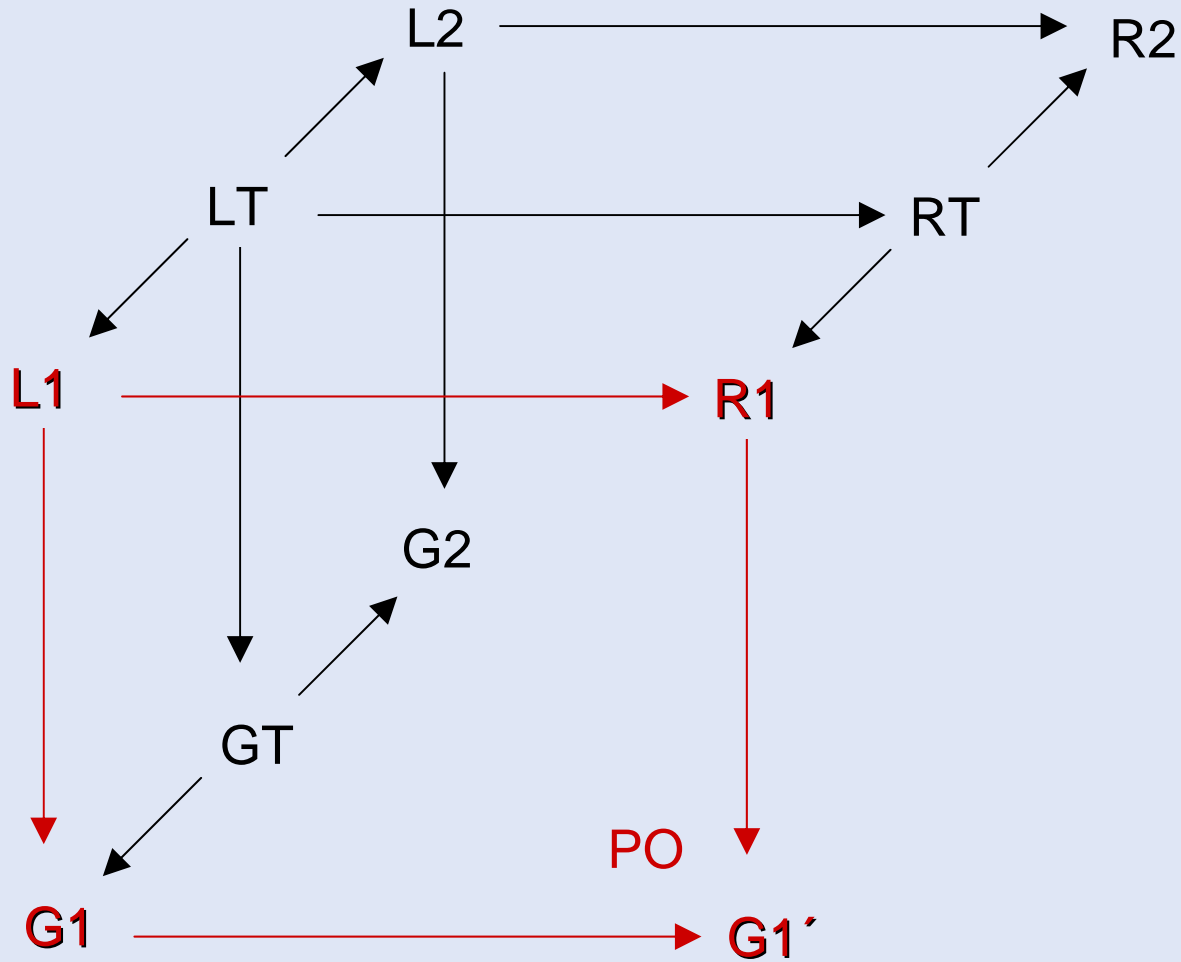
specifies simultaneous evolution of models

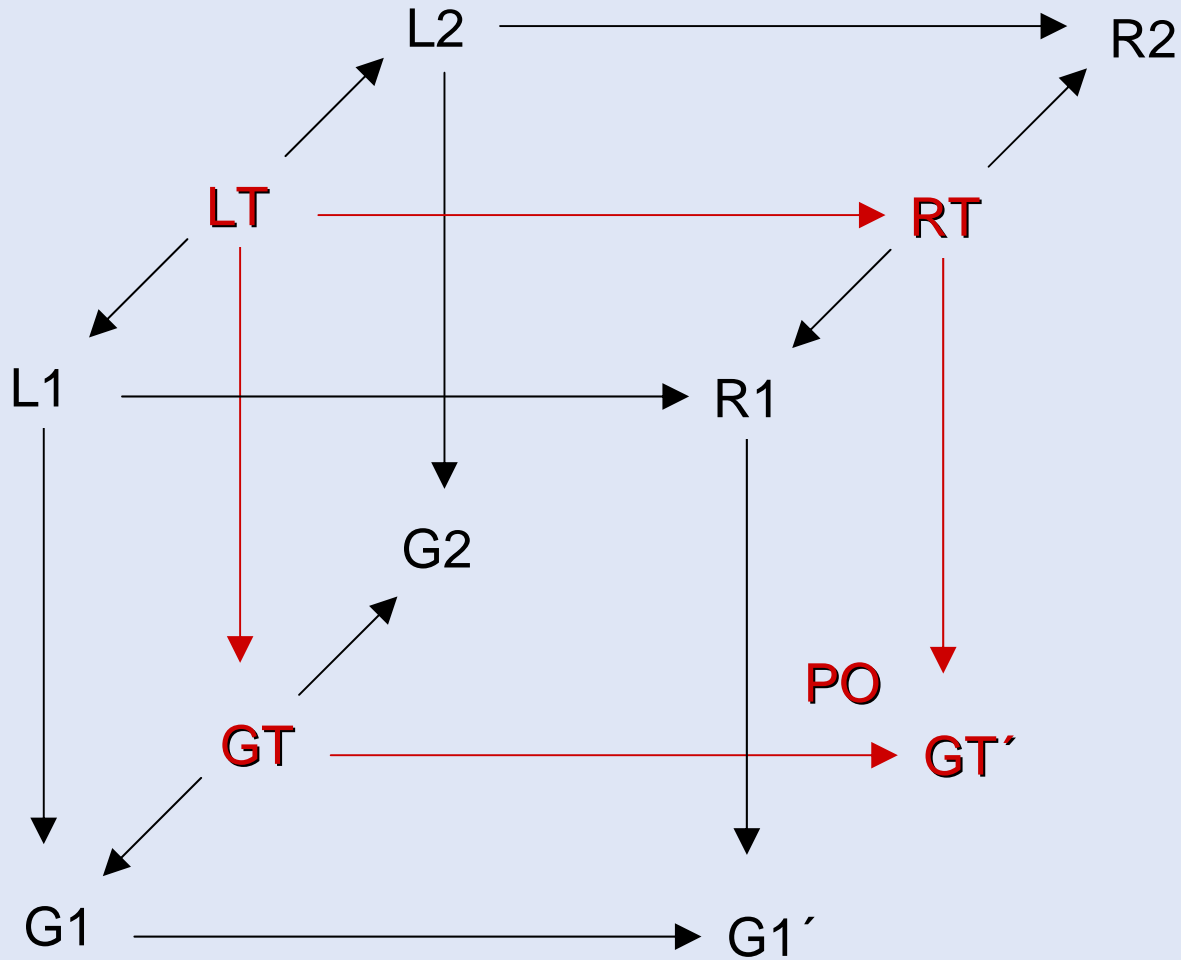


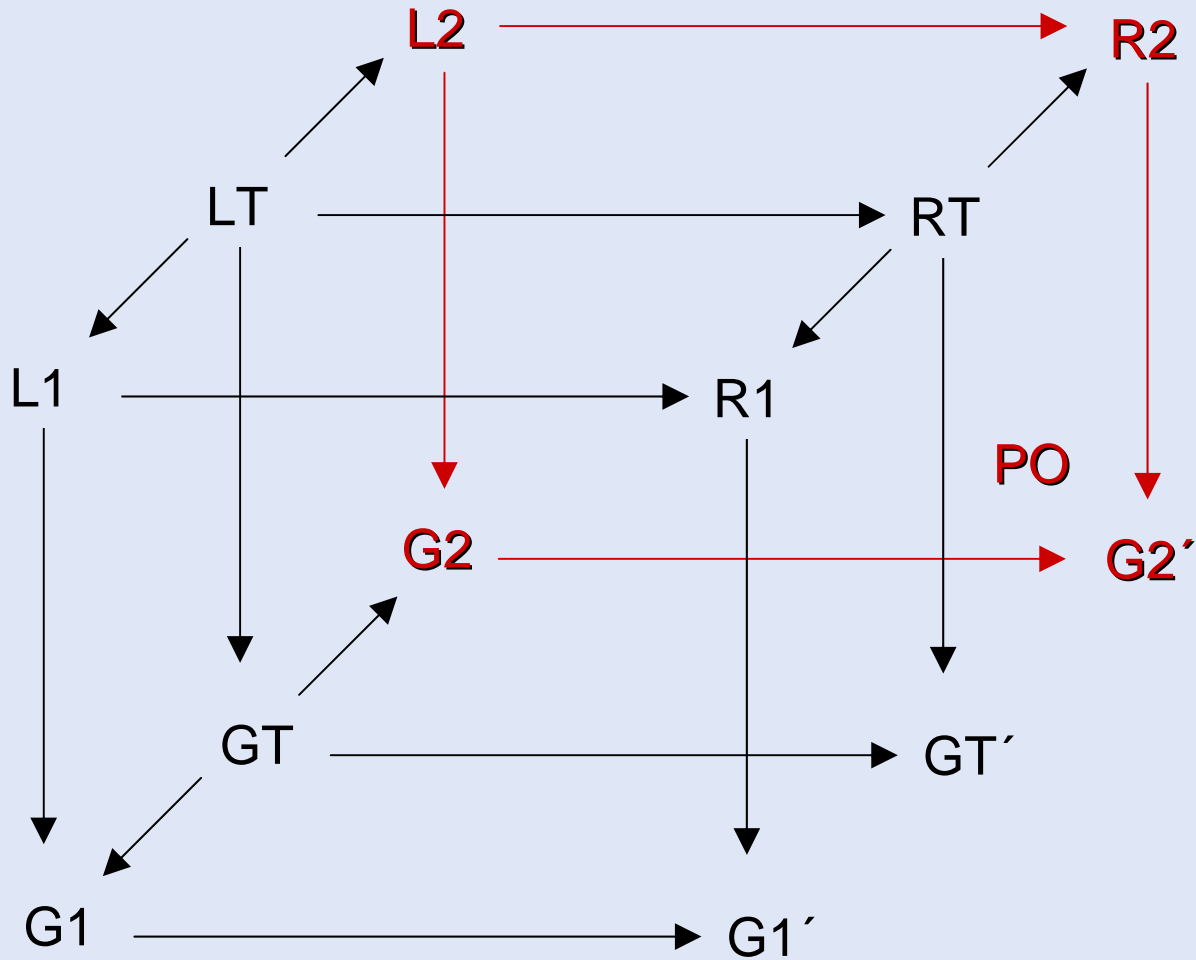


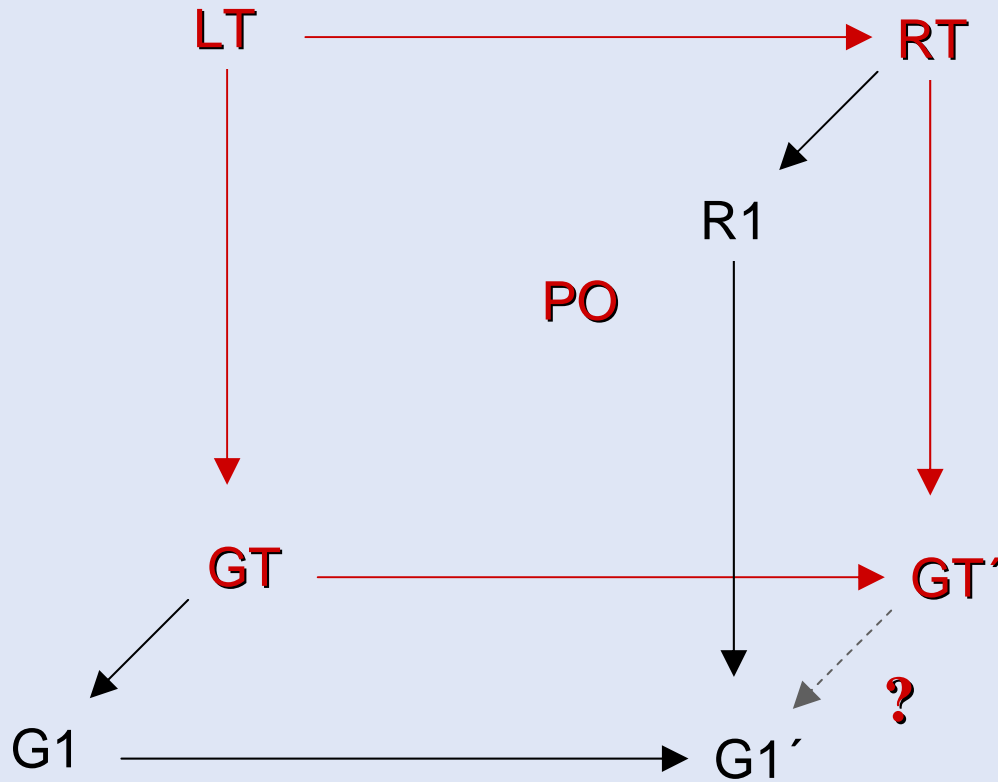
1. L1 and R1 are left- and right-hand side of transformation
($L1 \rightarrow R1$ identifies elements preserved by transformation)
2. G1 is „input“ graph for transformation
($L1 \rightarrow G1$ identifies match of graph transformation rule)
3. G1' is result of rule application to G1 (defined as pushout)

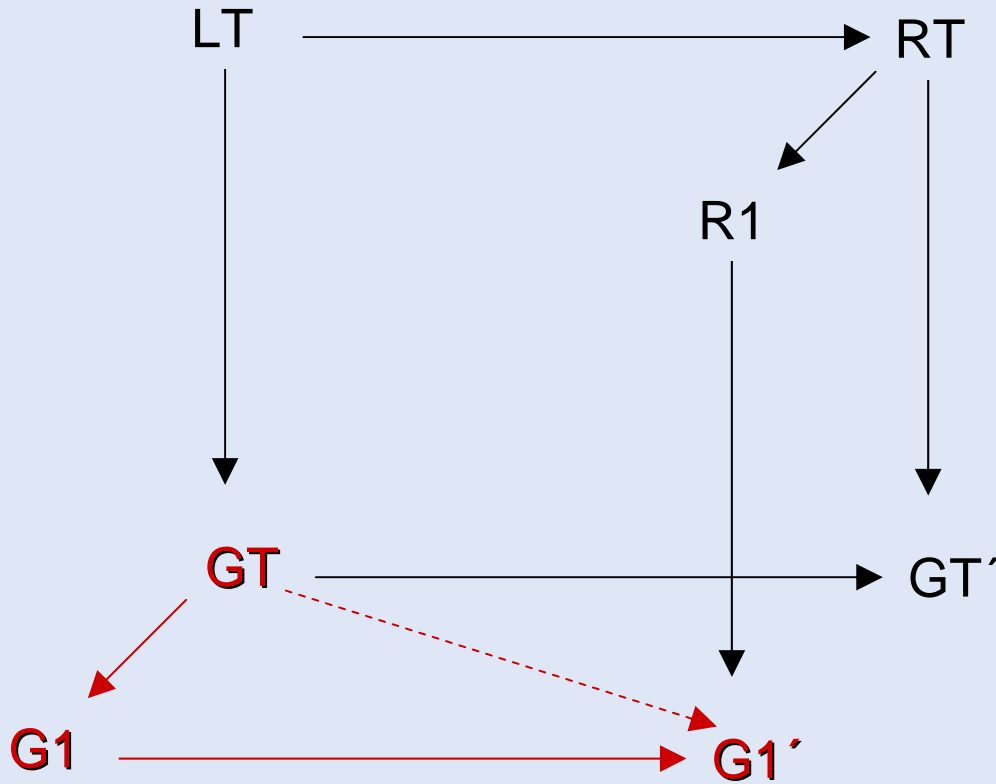


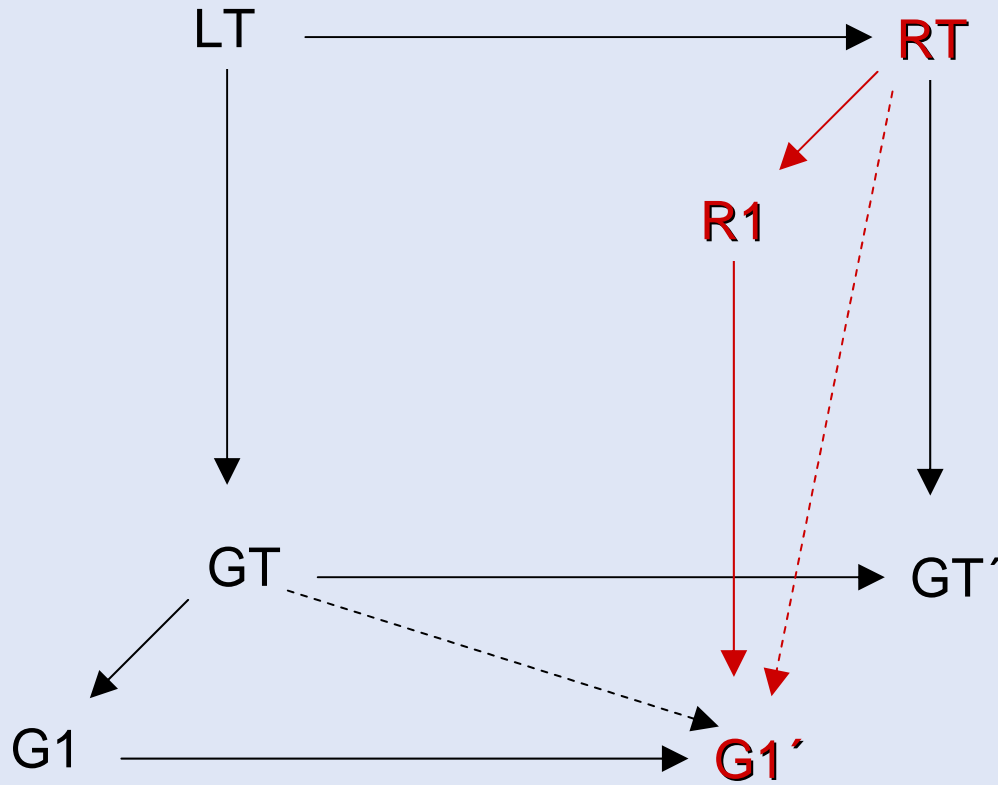


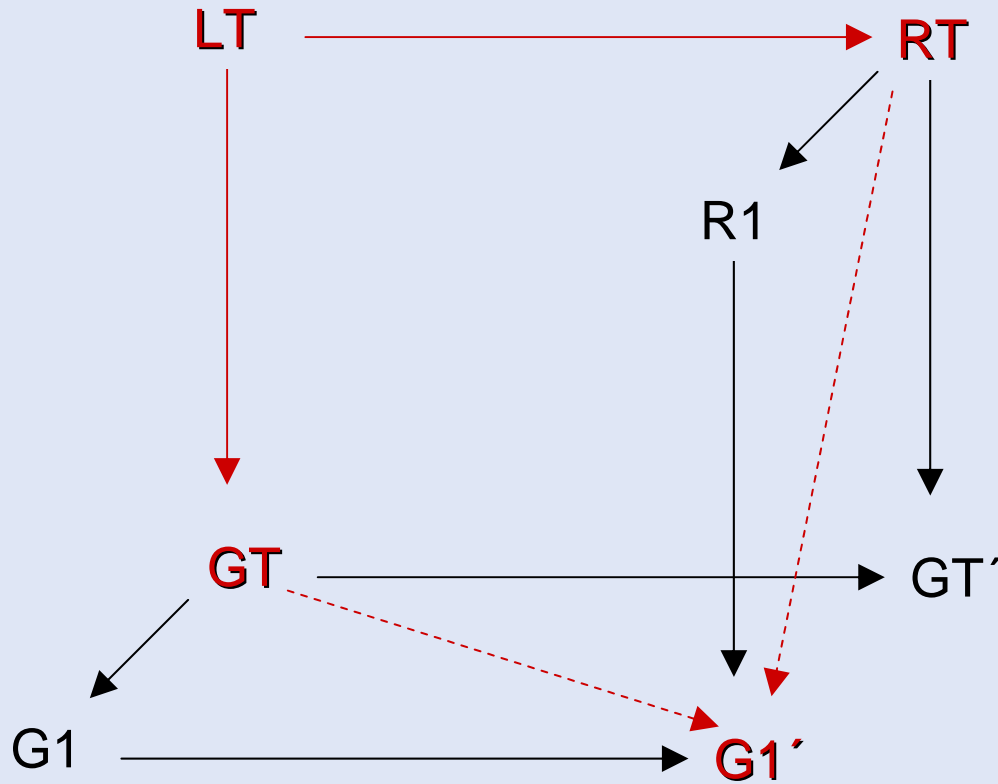


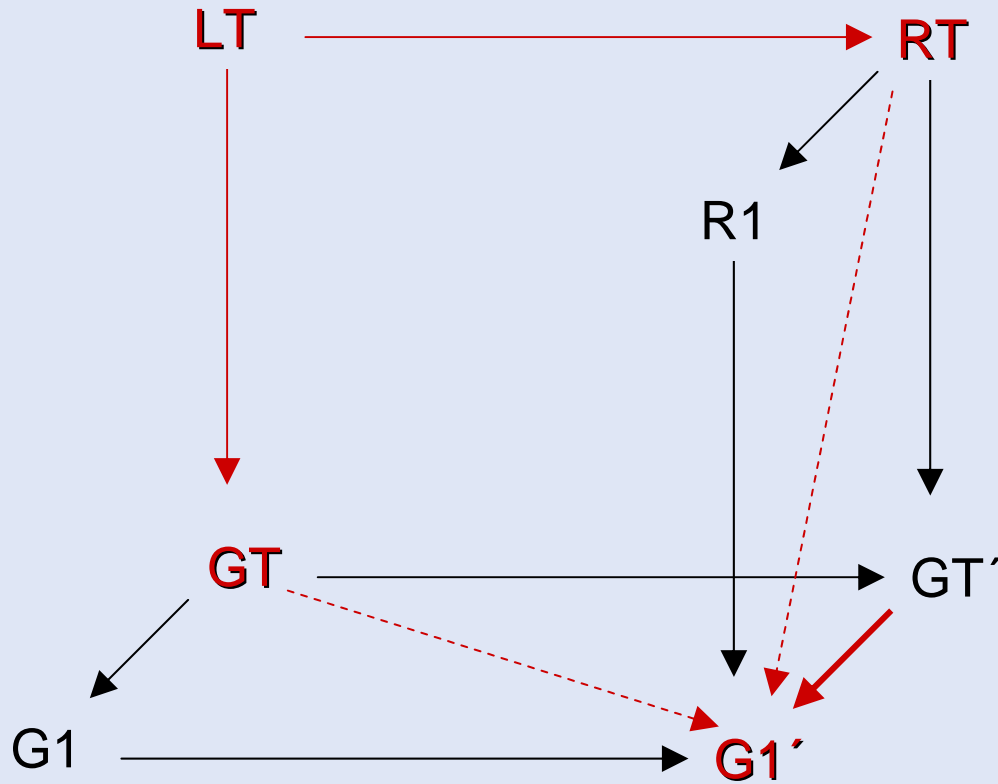














MOFLON [Classdiagram DB Integration] - C:\Do...urces\Classdiagram DB Integration.ctr

File Edit Diagrams TGG Rule Import/Export Tools Options Window Help

Classdiagram DB Integration

- Classdiagram Schema
 - DB Schema
 - model
 - T0
 - ClassToTable
 - CtT Rule
 - CtT Rule rule diagram
 - Class
 - Table
 - T1
 - ClassToTable
 - CtT Rule
 - CtT Rule rule diagram
 - Class
 - Table
 - T2
 - ClassToTable
 - CtT Rule
 - CtT Rule rule diagram
 - Class
 - Table
 - T3
 - AttributeToColumn
 - AtC Rule
 - AtC Rule rule diagram
 - Attribute
 - Column
- Classdiagram Schema
 - Classdiagram
 - Association
 - Attribute
 - Class
 - Classifier
 - PrimitiveDataType
 - Types
 - Types
- DB Schema
 - DB
 - Column
 - FKey
 - Table
 - Types

AtC Rule rule diagram [Classdiagram DB Integration]

CtT Rule rule diagram [Classdiagram DB Integration]

CtT Rule rule diagram [Classdiagram DB Integration]

T2 [Classdiagram DB Integration]

T0 [Classdiagram DB Integration]

TGG rule

TGG rule

TGG rule

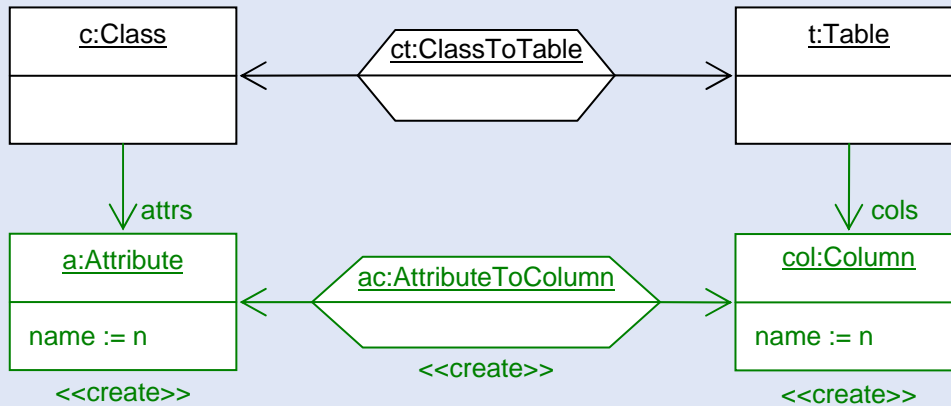
TGG schema

package dependencies

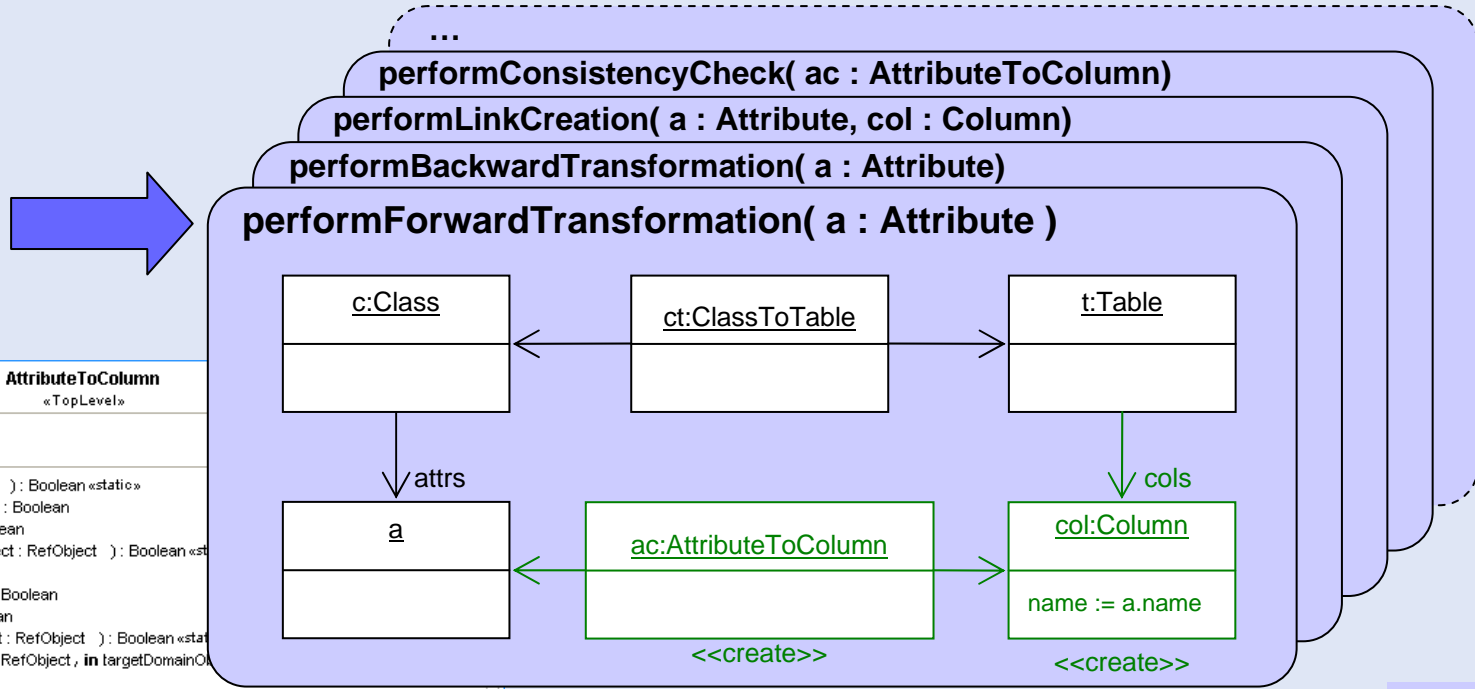
49 MByte of 53 MByte allocated



25 **AttributeToColumn(n : String)**

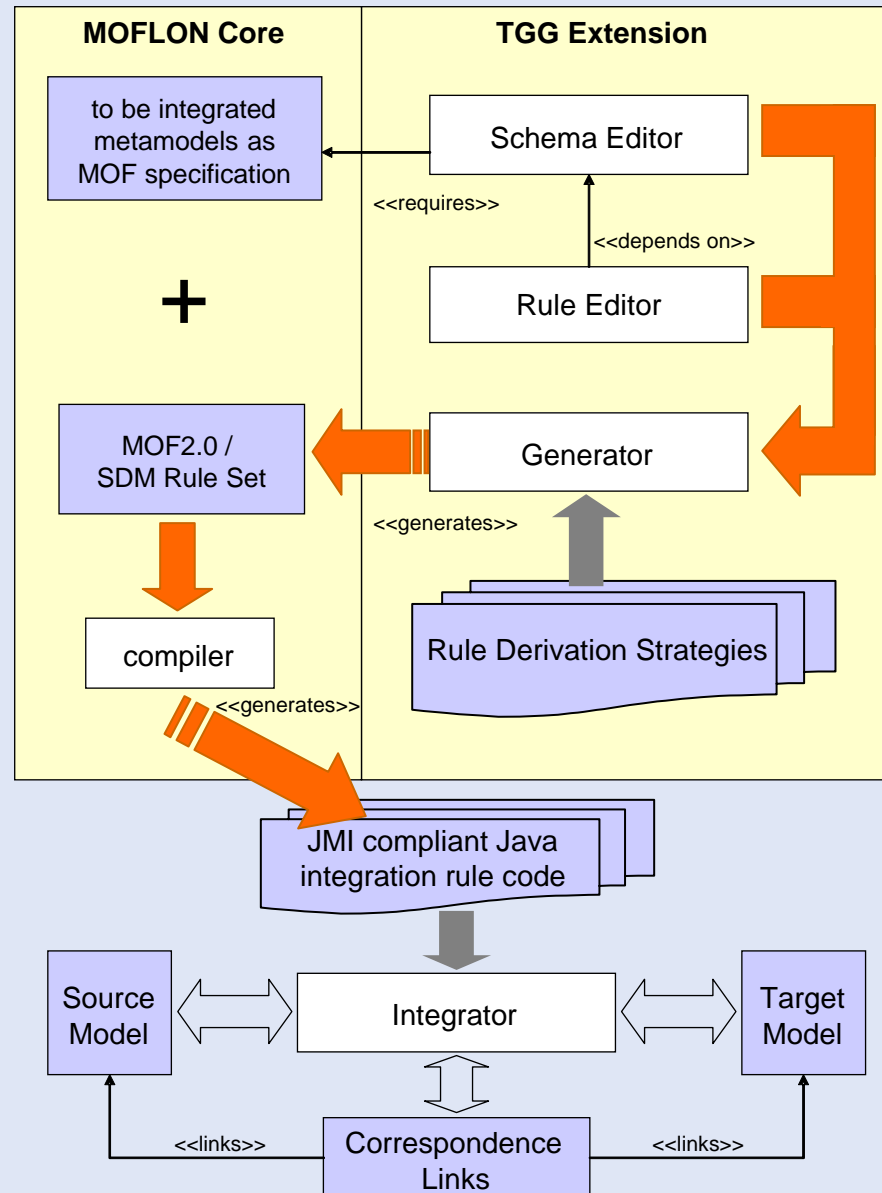


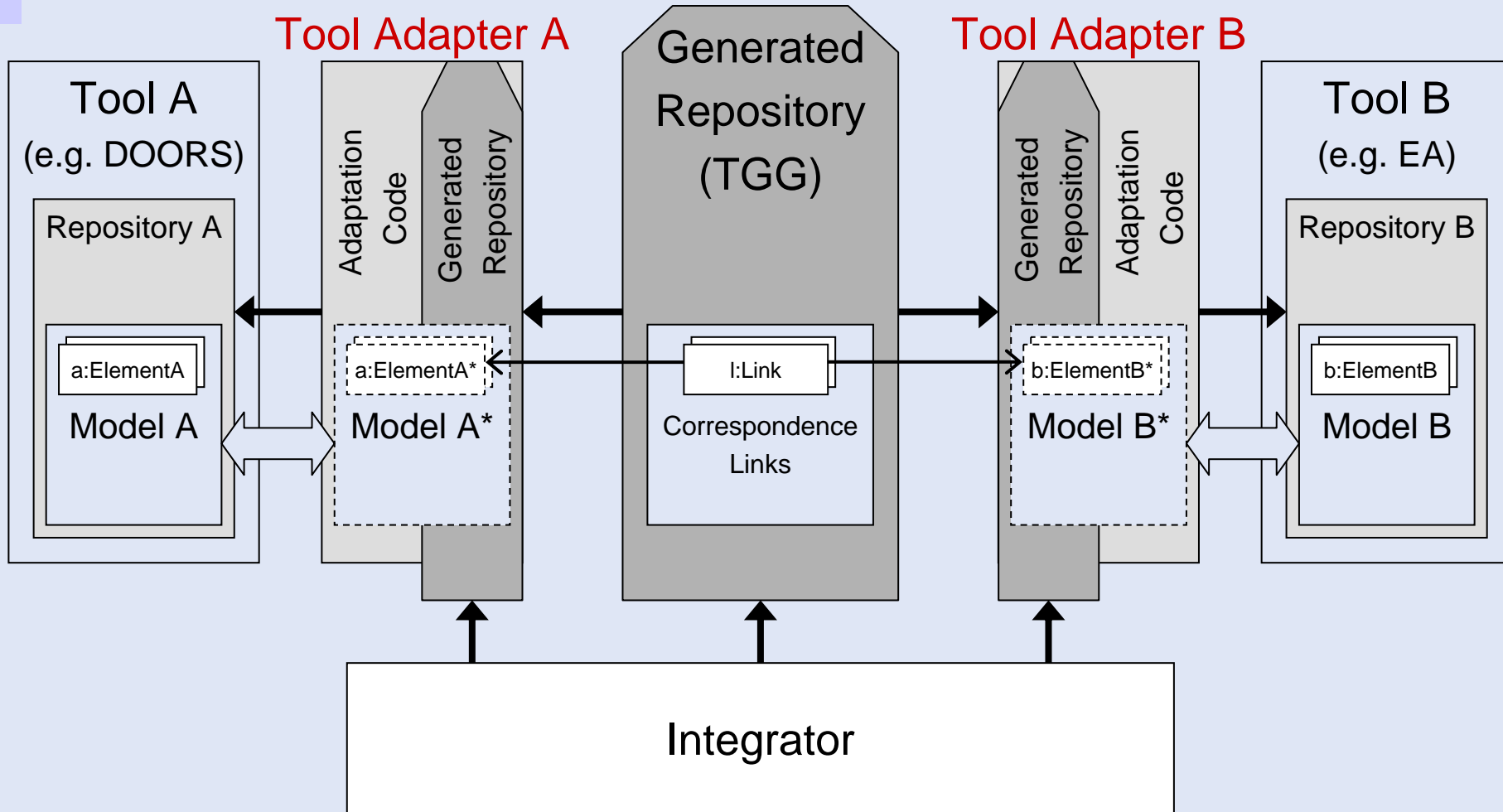
operational rules are derived from a TGG rule



```

AttributeToColumn
«TopLevel»
attribute : Attribute
column : Column
isAppropriateFor ( in sourceObject : RefObject ) : Boolean «static»
performBackwardAttributeValuePropagation ( ) : Boolean
performBackwardDeletionPropagation ( ) : Boolean
performBackwardTransformation ( in inputObject : RefObject ) : Boolean «static»
performConsistencyChecking ( ) : Boolean
performForwardAttributeValuePropagation ( ) : Boolean
performForwardDeletionPropagation ( ) : Boolean
performForwardTransformation ( in inputObject : RefObject ) : Boolean «static»
performLinkCreation ( in sourceDomainObject : RefObject , in targetDomainObject : RefObject ) : Boolean
performLinkDeletion ( ) : Boolean
    
```







for all *affected* elements e of input model in „*appropriate*“ order:
check whether e and related output model elements are consistent
(* with derived consistency checking rules *)
if not consistent then
 if structurally consistent then
 propagate attribute values only
 (* with derived attribute propagation rules *)
 else
 delete *no longer needed* inconsistent output model elements;
 (* with derived deletion rules if these elements exist *)
 (re-)create consistent output model elements;
 (* with „*appropriate*“ derived translation rule *)
 remember *new affected* input model elements
 end
end
end



- **1st generation Aachen/Paderborn TGG implementations:**
 - batch transformation
 - deterministic TGG
 - requires hierarchy on models
 - negative conditions not supported
 - no means for modularization / reuse
- **2nd generation Aachen TGG implementation (just finished):**
 - sophisticated (pseudo-)incremental repair actions
 - nondeterminism resolved via user interaction (defaults, priorities)
 - requires hierarchy on models
 - negative conditions not supported
 - no means for modularization / reuse



- **2nd generation Potsdam implementation (ongoing w.):**
 - first **really incrementally working** approach
 - relies on **event listeners** (for models)
 - requires **hierarchy of rules**
 - negative conditions not supported
 - no means for modularization / reuse
- **2nd generation Darmstadt implementation (ongoing work):**
 - naive (pseudo-)incremental transformation
 - relies on **model diff** algorithm (or event listener)
 - lazy on-demand computation **rule context** dependencies
 - **negative application conditions will be supported**
 - **modularization, decomposition, and refinement of rules**

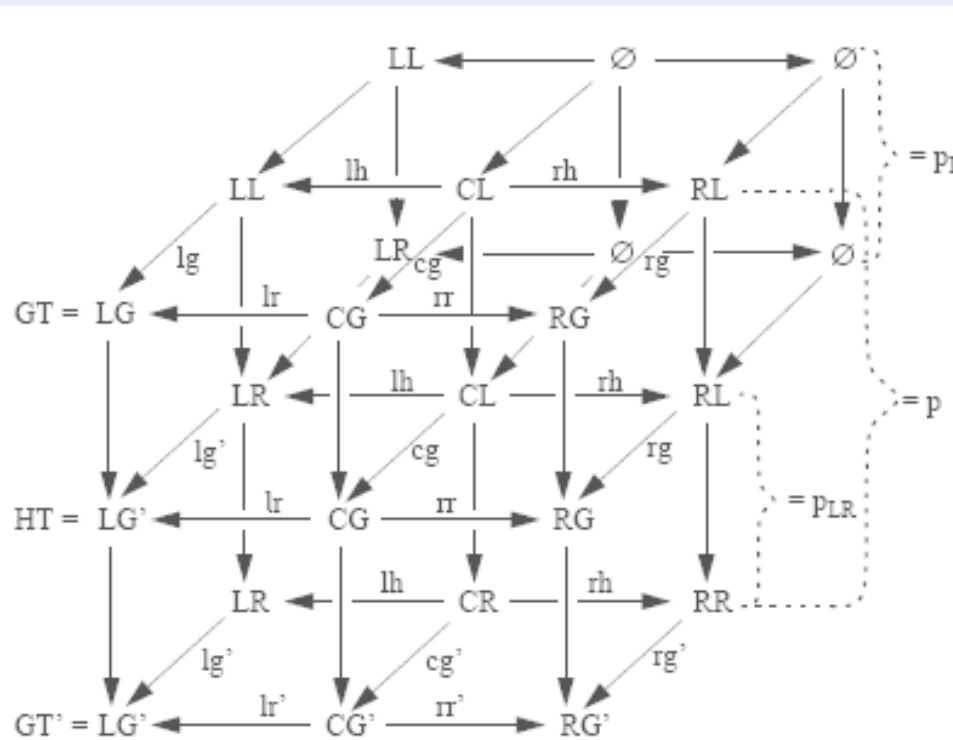


- Rule priorities are harmful (as well as other solutions):
 - **Correctness violation**: forward/backward translation create pairs of models not generated by TGG grammar itself
 - **Completeness violation**: forward/backward translation does not create pairs of models generated by TGG grammar itself
- Characterization of sublanguage of TGGs for which correct, complete, and efficient translation algorithms do exist
- More sophisticated synchronisation / reconciliation algorithms
- Automatic generation of tool adapters / (virtual) model views
- Comparison of practical usefulness of TGG and QVT philosophy
 - **TGG**: select one applicable rule (for overlapping matches)
 - **QVT**: apply all matching mappings and merge results (keys)



Resumée:

We do need a family of model transformation approaches and the TGG approach is just one interesting option:





- From Darmstadt:

<http://www.es.tu-darmstadt.de/forschung/publikationen/>
(select project TGG publication list drop down menu)

- World-wide: <http://www.es.tu-darmstadt.de/forschung/tgg/>

- Selected publications:

- A. Schürr, F. Klar: "**15 Years of Triple Graph Grammars - Research Challenges, New Contributions, Open Problems**", *4th International Conference on Graph Transformation*, Heidelberg: Springer Verlag, 09 2008; *Lecture Notes in Computer Science (LNCS)*, Vol. 5214, 411-425.
- A. Schürr: "**Specification of Graph Translators with Triple Graph Grammars**", in: G. Tinhofer (ed.), *WG'94 20th Int. Workshop on Graph-Theoretic Concepts in Computer Science*, Heidelberg: Springer Verlag, 1994; *Lecture Notes in Computer Science (LNCS)*, Vol. 903, 151-163.
- F. Klar, A. Königs, A. Schürr: "**Model Transformation in the Large**", *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering*, New York: ACM Press, 2007; *ACM Digital Library Proceedings*, 285-294.
- Giese, H. & Wagner, R.: "**Incremental Model Synchronization with Triple Graph Grammars**", in: Nierstrasz, O.; Whittle, J.; Harel, D. & Reggio, G. (eds.): *Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, Heidelberg: Springer Verlag, Oct. 2006; *Lecture Notes in Computer Science (LNCS)*, Vol. 4199, 543-557
- Greenyer, J. & Kindler, E.: "**Reconciling TGGs with QVT**", *Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, Heidelberg: Springer Verlag, Oct. 2007; *Lecture Notes in Computer Science (LNCS)*, Vol. 4735, 16-30