Information Preserving Bidirectional Model Transformations*

Hartmut Ehrig[1], Karsten Ehrig[2], Claudia Ermel[1], Frank Hermann[1], and Gabriele Taentzer[3]

[1] Department of Computer Science, Technical University Berlin, Germany
{ehrig,lieske,frank}@cs.tu-berlin.de
[2] Department of Computer Science, University of Leicester, United Kingdom
karsten@mcs.le.ac.uk
[3] Department of Mathematics and Computer Science, Phillips-University Marburg, Germany
taentzer@mathematik.uni-marburg.de

Abstract. Within model-driven software development, model transformation has become a key activity. It refers to a variety of operations modifying a model for various purposes such as analysis, optimization, and code generation. Most of these transformations need to be bidirectional to e.g. report analysis results, or keep coherence between models. In several application-oriented papers it has been shown that triple graph grammars are a promising approach to bidirectional model transformations. But up to now, there is no formal result showing under which condition corresponding forward and backward transformations are inverse to each other in the sense of information preservation. This problem is solved in this paper based on general results for the theory of algebraic graph transformations. The results are illustrated by a transformation of class models to relational data base models which has become a quasi-standard example for model transformation.

1 Introduction

Model transformation is a central activity in model-driven software development as it is used thoroughly for model optimization and other forms of model evolution. Moreover, model transformation is used to map models between different domains for analyzing them or for automatically generating code from them. Often a model transformation is required to be reversible to translate information back to source models. For example, a transformation of a domain-specific model to some formal model for the purpose of validation should be reversible to transform back analysis results stemming from the formal model. Reversible model transformations also play an important role in the presence of system evolution. Having usually a variety of different models around in the engineering process, the evolution of one model depends on the evolution of other models. To keep models coherent to each other, model transformations have to be reversible.

* This work has been partially sponsored by the project SENSORIA, IST-2005-016004.

M.B. Dwyer and A. Lopes (Eds.): FASE 2007, LNCS 4422, pp. 72–86, 2007.

Model transformations have been classified by Czarnecki, Mens et.al. [CH03, MG06]. Mens et.al. distinguish two main classes: endogenous and exogenous model transformations. While the former run within one modeling language, e.g. are used to model refactorings or other kinds of optimizations, the latter are used to translate models between different languages. In the context of this paper we concentrate on exogenous model transformations. A promising approach to reversible transformation is bi-directional model transformation, since only one transformation description is needed to deduce forward and backward transformations automatically.

A bi-directional model transformation can be well described by triple graph transformations as introduced by Schürr et.al. [KS06, Sch94]. The main idea is to relate a source and a target graph by some correspondence graph in between which is mapped to both graphs. In this way, source and target graphs are coupled and a basic structure for consistent co-evolution of the model graphs is established. Triple rules are used to formulate conditions for consistent co-evolution describing the simultaneous transformation of source and target graphs. It is often the case though that these graphs do not develop simultaneously: i.e. one graph evolves and the other one has to be updated accordingly. To capture this situation, Königs and Schürr showed that each triple rule can be split into a so-called source rule which changes the source graph only and a forward rule which updates the target accordingly. Furthermore, they lifted this result to transformation sequences in [KS06]. This means that we obtain for each triple transformation sequence a corresponding forward transformation and dually also a corresponding backward transformation sequence.

But up to now, there is no formal result showing under which conditions a given forward transformation sequence has an inverse backward sequence in the sense that both together are information preserving concerning the source graphs. The main result of this paper solves this problem under the condition that a given forward transformation sequence $G_1 \overset{tr_F^*}{\Longrightarrow} G_2$ is source consistent. Roughly speaking, that means G_1 can be generated by source rules only. This result is based on an extension of the result in [KS06] cited above, which allows to state a bijective correspondence between triple transformation sequences and combined match consistent source and forward transformation sequences. The proof of this extended result is based on the well-known Local Church–Rosser and Concurrency Theorem for graph transformations (see [EEPT06]) which are shown to be valid also for triple graph grammars.

All main concepts and results are illustrated at a running example, which is a model transformation from class models to relational data base models. This quasi-standard model transformation has been originally defined in the specification for QVT [OMG05] by the Object Management Group. Due to space limitations, we present a triple graph grammar for a restricted form of this model transformation.

In Section 2 we start with a review of triple graph grammar for graphs and introduce the running example in Section 3. Section 4 presents the main results concerning information preserving forward and backward transformations. In

Section 5 we discuss how to obtain a general theory for triple graph transformations which can be based also on typed and attributed graphs.

2 Review of Triple Rules and Triple Graph Grammars

Triple graph grammars [Sch94] have been shown to be a promising approach to consistently co-develop two related structures. They provide bidirectional transformation between a pair of graphs representing these structures which are connected using a third so-called correspondence graph together with its embeddings into the source and target graph. In [KS06], Königs and Schürr formalize the basic concepts of triple graph grammars in a set-theoretical way. In this section, we take up this formalization and present further steps of a theory of triple graph grammars in the following sections. We first base this formalization on simple graphs and will discuss the extension to typed, attributed graphs based on concepts from category theory in Section 5.

Definition 1 (Graph and Graph Morphism). *A graph $G = (V, E, s, t)$ consists of a set V of nodes (also called vertices), E of edges and two functions $src, tar : E \to V$, the source and target functions. Given graphs G_1, G_2 with $G_i = (V_i, E_i, src_i, tar_i)$ for $i = 1, 2$, a graph morphisms $f : G_1 \to G_2$, $f = (f_V, f_E)$, consists of two functions $f_V : V_1 \to V_2$ and $f_E : E_1 \to E_2$ that preserve the source and target functions, i.e. $f_V \circ src_1 = src_2 \circ f_E$ and $f_V \circ tar_1 = tar_2 \circ f_E$.*

Definition 2 (Triple Graph and Triple Graph Morphism). *Three graphs SG, CG, and TG, called source, connection, and target graphs, together with two graph morphisms $s_G : CG \to SG$ and $t_G : CG \to TG$ form a triple graph $G = (SG \xleftarrow{s_G} CG \xrightarrow{t_G} TG)$. G is called empty, if SG, CG, and TG are empty graphs.*

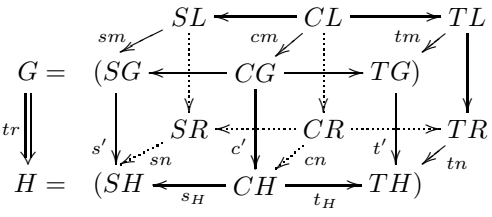
A triple graph morphism $m = (s, c, t) : G \to H$ between two triple graphs $G = (SG \xleftarrow{s_G} CG \xrightarrow{t_G} TG)$ and $H = (SH \xleftarrow{s_H} CH \xrightarrow{t_H} TH)$ consists of three graph morphisms $s : SG \to SH$, $c : CG \to CH$ and $t : TG \to TH$ such that $s \circ s_G = s_H \circ c$ and $t \circ t_G = t_H \circ c$. It is injective, if morphisms s, c and t are injective.

A triple rule is used to build up source and target graphs as well as their connection graph, i.e. to build up triple graphs. Structure filtering which deletes parts of triple graphs, are performed by projection operations only, i.e. structure deletion is not done by rule applications. Thus, we can concentrate our investigations on non-deleting triple rules without any restriction.

Definition 3 (Triple Rule tr and Triple Transformation Step). *A* triple rule *tr consists of triple graphs L and R, called left-hand and right-hand sides, and an injective triple graph morphism $tr = (s, c, t) : L \to R$.*

$$\begin{array}{ccccc} L & = & (SL & \xleftarrow{s_L} & CL \xrightarrow{t_L} TL) \\ {\scriptstyle tr}\downarrow & & {\scriptstyle s}\downarrow & & {\scriptstyle c}\downarrow \quad\quad \downarrow{\scriptstyle t} \\ R & = & (SR & \xleftarrow[s_R]{} & CR \xrightarrow[t_R]{} TR) \end{array}$$

Given a triple rule $tr = (s, c, t) : L \to R$, *a triple graph* G *and a triple graph morphism* $m = (sm, cm, tm) : L \to G$, *called triple match* m, *a triple graph transformation step (TGT-step)* $G \overset{tr,m}{\Longrightarrow} H$ *from* G *to a triple graph* H *is given by three pushouts* (SH, s', sn), (CH, c', cn) *and* (TH, t', tn) *in category* **Graph** *with induced morphisms* $s_H : CH \to SH$ *and* $t_H : CH \to TH$.



Moreover, we obtain a triple graph morphism $d : G \to H$ with $d = (s', c', t')$ called transformation morphism. A sequence of triple graph transformation steps is called triple (graph) transformation sequence, short: TGT-sequence. Furthermore, a triple graph grammar $TGG = (S, TR)$ consists of a triple start graph S and a set TR of triple rules.

Remark 1 (gluing construction). Each of the pushout objects SH, CH, TH in Def. 3 can be constructed as a gluing construction, e.g. $SH = SG +_{SL} SR$, where the S-components SG of G and SR of R are glued together via SL (see [EEPT06] Chapter 2 for more details).

3 Case Study: CD2RDBM Model Transformation

This case study presents a model transformation problem (see [BRST05,OMG05]) which occurs in several variants. It contains the transformation of class models to relational database models. We will use it in this paper to illustrate the triple graph grammar approach and especially, the conditions for information preserving bidirectional transformations. In contrast to [BRST05], we present a slightly restricted variant where the different treatment of persistent and non-persistent classes is omitted, due to space limitations.

The source language consists of class diagrams, while the target language consists of schemes for database tables. A reference structure is established as helper structure for the model transformation which relates classes with tables and subclasses or attributes with columns. Associations are translated to foreign keys. The relationship between the elements of the source and the target language is documented in the TGG type graph in Fig. 1 where dashed edges represent the morphisms s and t connecting the the source and the target graph via a connection graph.

Please note that this case study is given in the framework of triple graphs over typed attributed graphs which is briefly discussed in Section 5. In that section, we also show how to extend the basic theory presented in Sections 2 and 4, to typed attributed graphs.

Fig. 2 shows four of the triple rules for the *CD2RDBM* model transformation. Triple rule *Class2Table* simultaneously creates a class and a table which are related to each other. Since all triple rules are non-deleting, they are depicted in a compact notation not separating the left from the right-hand side. All graph

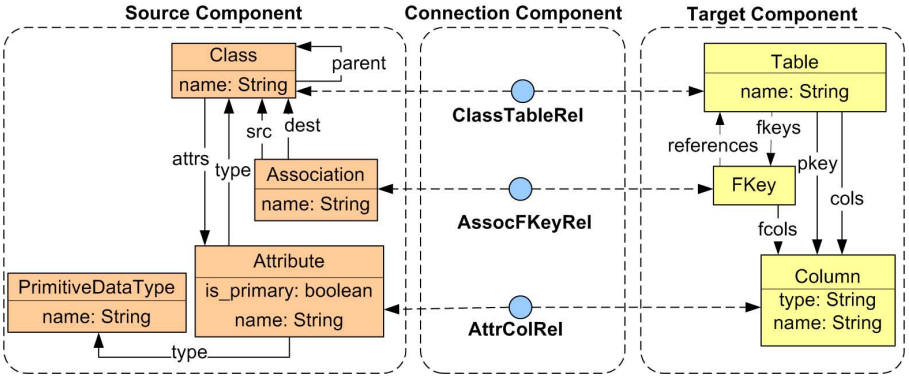


Fig. 1. TGG type graph for *CD2RDBM* model transformation

items which are newly created, are annotated by "{ new}". Those items occur in the right-hand side of a rule only. Triple rule *PrimaryAttribute2Column* creates columns and attributes. Given that a class is already related to some table, an attribute of this class is related to a column of the related table. By triple rule *SetKey*, the corresponding column for each primary attribute is set as primary key . A newly created subclass is related to the same table as its given superclass by triple rule *Subclass2Table*.

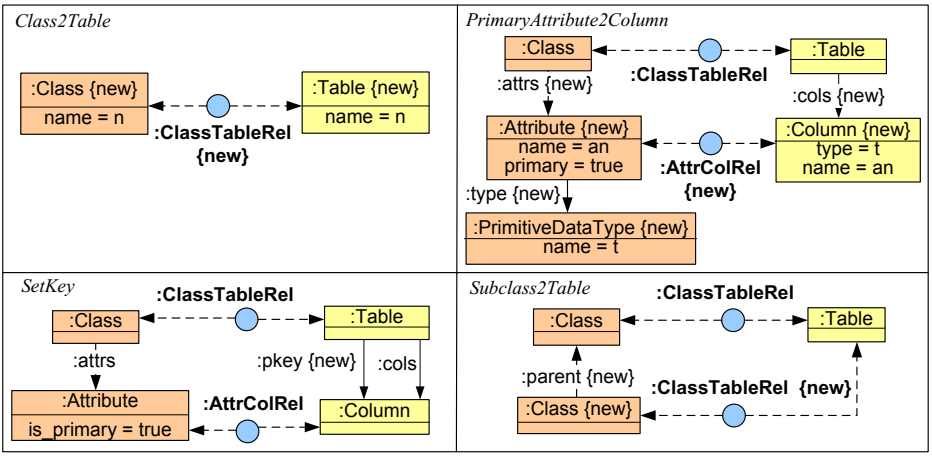


Fig. 2. TGG rules for *CD2RDBM* model transformation

Fig. 3 shows triple rule *Association2FKey* which creates associations related to foreign keys (FKey) pointing to columns of other tables. A similar triple rule *Attribute2FKey* (not depicted) creates class-typed attributes which are also related to foreign keys. Instead of the :*Association*{*new*} node in rule

Association2FKey, rule *Attribute2FKey* has an :*Attribute*{*new*} node, connected by an :*attrs*{*new*} edge to the upper class and a :*type*{*new*} edge to the lower class.

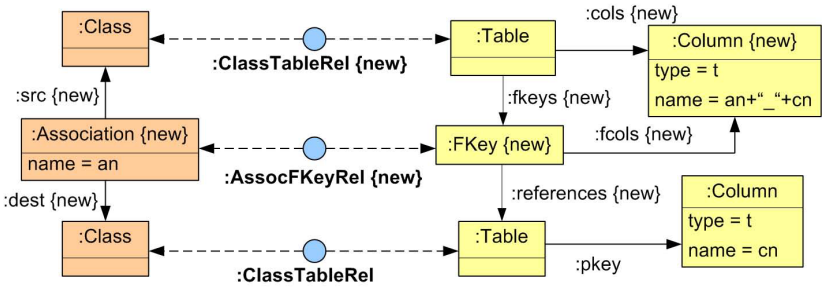


Fig. 3. TGG rule *Association2FKey*

4 Information Preserving Forward and Backward Transformations

The power of bi-directional model transformations is its potential to invert a forward transformation without specifying a new transformation. Deriving rules for forward and backward transformations automatically, we investigate the requirements for such a reversal to be fulfilled by triple graph transformations. A sufficient requirement for reversal is based on the notion of source transformation which is the projection of a triple graph transformation to its source component. It is sufficient to show that a source structure can be constructed by source transformations only. In this case, the forward transformation is called source consistent and we can show that it can be inverted, i.e. there is a backward transformation leading back to the same source structure as the original one.

For updating the changes of the source to the target graph and vice versa *forward* as well as *backward* rules are needed and they can be derived from a triple rule. In addition we can deduce a source rule tr_S and a target rule tr_T with empty connection and target or source component from a triple rule tr.

Definition 4 (Derived Triple Rules). *Given a triple rule tr as in Def. 3, a* source rule tr_S*, a* target rule tr_T*, a* forward rule tr_F *and a* backward rule tr_B *can be constructed as shown below:*

$$\begin{array}{ccccc} L_S = (SL & \longleftarrow & \emptyset & \longrightarrow & \emptyset) \\ {\scriptstyle tr_S}\downarrow & {\scriptstyle s}\downarrow & \downarrow & & \downarrow \\ R_S = (SR & \longleftarrow & \emptyset & \longrightarrow & \emptyset) \end{array}$$

source rule tr_S

$$\begin{array}{ccccc} L_F = (SR & \xleftarrow{s \circ s_L} & CL & \xrightarrow{t_L} & TL) \\ {\scriptstyle tr_F}\downarrow & {\scriptstyle id}\downarrow & {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle t} \\ R_F = (SR & \xleftarrow{s_R} & CR & \xrightarrow{t_R} & TR) \end{array}$$

forward rule tr_F

$$\begin{array}{ccccc} L_T = (\emptyset & \longleftarrow & \emptyset & \longrightarrow & TL) \\ {\scriptstyle tr_T}\downarrow & & \downarrow & & \downarrow{\scriptstyle t} \\ R_T = (\emptyset & \longleftarrow & \emptyset & \longrightarrow & TR) \end{array}$$

target rule tr_T

$$\begin{array}{ccccccc} L_B = (SL & \xleftarrow[s_L]{} & CL & \xrightarrow[tot_L]{} & TR) \\ {\scriptstyle tr_F}\downarrow & {\scriptstyle s}\downarrow & {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle id} \\ R_B = (SR & \xleftarrow{s_R} & CR & \xrightarrow{t_R} & TR) \end{array}$$

backward rule tr_B

Example 1 (derived forward and backward rules for triple rule Class2Table) Fig. 4 shows the forward and backward rules derived from triple rule *Class2Table* in Fig. 2 (a). In the forward rule a new table is created for an existing class. Vice versa, in the backward rule a table exists already and the corresponding class is created.

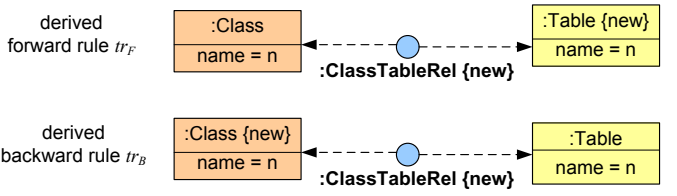


Fig. 4. derived forward and backward rules for triple rule *Class2Table*

Note that the source rule tr_S and the target rule tr_T can be obtained by projection of tr to source and target, respectively.

Definition 5 (Projection). *Given a triple graph* $G = (SG \overset{s_G}{\leftarrow} CG \overset{t_G}{\rightarrow} TG)$*, the* projection $proj_T(G)$ *to the target is triple graph* $G_T = (\emptyset \overset{\emptyset}{\leftarrow} \emptyset \overset{\emptyset}{\rightarrow} TG)$ *and the* projection $proj_S(G)$ *to the source is triple graph* $G_S = (SG \overset{\emptyset}{\leftarrow} \emptyset \overset{\emptyset}{\rightarrow} \emptyset)$.

A first important result shows that each TGT-sequence can be decomposed in transformation sequences by corresponding source and forward rules and vice versa, provided that their matches are consistent. Roughly spoken, match consistency means that the co-matches of source rule applications determine the matches of corresponding forward rule applications.

The following Theorem 1 is partly given as Theorem 4.7 in [KS06] where especially the bijective correspondence between *decomposition* and *composition* is missing which however, is most important in this paper. Essential for this bijective correspondence is the notion of match consistency for specific TGT-sequences applying source and forward rules tri_S and tri_F of the same triple rule tri for $i = 1, \ldots, n$.

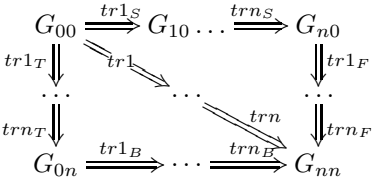
Definition 6 (Match Consistency). *A TGT-sequence* $G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{trn_S}{\Longrightarrow} G_{n0} \overset{tr1_F}{\Longrightarrow} G_{n1} \Rightarrow \cdots \overset{trn_F}{\Longrightarrow} G_{nn}$ *is called match consistent, if the S-component of the match* $m1_F$ *of* $G_{n0} \overset{tr1_F}{\Longrightarrow} G_{n1}$ *is completely determined by the co-match* $n1_S$ *of* $G_{00} \overset{tr1_S}{\Longrightarrow} G_{10}$ *and the transformation morphism* $d_1 : G_{10} \rightarrow G_{n0}$*, i.e.* $(m1_F)_S = d1_S \circ (n1_S)_S$ *and similar for all matches of the forward transformations* tri_F $(i > 1)$*. For* $n = 1$ *this means* $(m1_F)_S = (n1_S)_S$.

Theorem 1 (Decomposition and Composition of TGT-Sequences)

1. **Decomposition:** *For each TGT-sequence*
$$(1)\quad G_0 \overset{tr_1}{\Longrightarrow} G_1 \Longrightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n$$
there is a corresponding match consistent TGT-sequence
$$(2)\ G_0 = G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \Rightarrow \cdots \overset{trn_S}{\Longrightarrow} G_{n0} \overset{tr1_F}{\Longrightarrow} G_{n1} \Rightarrow \cdots \overset{trn_F}{\Longrightarrow} G_{nn} = G_n.$$
2. **Composition:** *For each match consistent transformation sequence (2) there is a canonical transformation sequence (1).*
3. **Bijective Correspondence:** *Composition and Decomposition are inverse to each other.*

The proof is given in Section 5.2.

Remark 2. Moreover, we have $proj_T(G_{00}) = proj_T(G_{n0})$ and $proj_S(G_{n0}) = proj_S(G_{nn})$ in (2), due to the special form of triple rules $tr1_S, .., trn_S$ and $tr1_F, .., trn_F$, respectively. Dual results hold for target rules tr_T and backward rules tr_B (see the lower triangle in the figure on the right).



Theorem 1 and its dual version lead to the following equivalence of forward and backward TGT-sequences which can be derived from the same general TGT-sequence.

Theorem 2 (Equivalence of Forward and Backward TGT-sequences). *Each of the following TGT-sequences implies the other ones assumed that the matches are uniquely determined by each other.*

1. $G_0 \overset{tr_1}{\Longrightarrow} G_1 \overset{tr_2}{\Longrightarrow} G_2 \Longrightarrow ... \overset{tr_n}{\Longrightarrow} G_n$
2. $G_0 = G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \Longrightarrow ... \overset{trn_S}{\Longrightarrow} G_{n0} \overset{tr1_F}{\Longrightarrow} G_{n1} \Longrightarrow ... \overset{trn_F}{\Longrightarrow} G_{nn} = G_n$, *which is match consistent. In this case we have:* $proj_T(G_{00}) = proj_T(G_{n0})$, $proj_S(G_{n0}) = proj_S(G_{nn})$
3. $G_0 = G_{00} \overset{tr1_T}{\Longrightarrow} G_{01} \Longrightarrow ... \overset{trn_T}{\Longrightarrow} G_{0n} \overset{tr1_B}{\Longrightarrow} G_{1n} \Longrightarrow ... \overset{trn_B}{\Longrightarrow} G_{nn} = G_n$, *which is match consistent. In this case we have:* $proj_S(G_{00}) = proj_S(G_{0n})$, $proj_T(G_{0n}) = proj_T(G_{nn})$

Proof. Theorem 2 is a direct consequence of Theorem 1 concerning decomposition and composition of forward TGT-sequences and its dual version for target rules tri_T and backward rules tri_B where match consistency in Part 3 is defined by the T-components of the matches. The projection properties follow from Remark 2.2.

In the following we use the short notations for TGT-sequences introduced in Theorem 2:

1. $F \overset{tr^*}{\Longrightarrow} H$, with $F = G_0, H = G_n$ for sequence (1),

2. $F \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H$, with $F = G_0, G = G_{n0}, H = G_n$ for sequence (2), and
3. $F \overset{tr_T^*}{\Longrightarrow} K \overset{tr_B^*}{\Longrightarrow} H$, with $F = G_0, K = G_{0n}, H = G_n$ for sequence (3).

Now we are able to address the main topic of this paper. We want to analyse under which conditions a forward TGT-sequence $G \overset{tr_F^*}{\Longrightarrow} H$ is information preserving in the sense that there is a backward TGT-sequence starting from $H_T = proj_T(H)$ and leading to H' such that the source graphs of G and H' are equal, i.e. $proj_S(G) = proj_S(H')$. That means sequence $G \overset{tr_F^*}{\Longrightarrow} H \overset{proj_T}{\longrightarrow} H_T \overset{tr_B^*}{\Longrightarrow} H'$ is information preserving concerning the source component. In this case we say that $G \overset{tr_F^*}{\Longrightarrow} H$ is backward information preserving.

The condition under which we obtain backward information preservation is source consistency of $G \overset{tr_F^*}{\Longrightarrow} H$, i.e. G is generated by corresponding source rules tr_S^* such that $\emptyset \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H$ is match consistent.

Definition 7 (Information Preserving Forward Transformation)
A forward TGT-sequence $G \overset{tr_F^*}{\Longrightarrow} H$ is

(1) **backward information preserving**, *if for $H_T = proj_T(H)$ there is a backward TGT-sequence $H_T \overset{tr_B^*}{\Longrightarrow} H'$ with $G_S = proj_S(G) = proj_S(H')$.*

$$\begin{array}{ccccc} G & \overset{tr_F^*}{\Longrightarrow} & H & \xrightarrow{proj_T} & H_T \overset{tr_B^*}{\Longrightarrow} H' \\ & {\scriptstyle proj_S}\searrow & & \swarrow{\scriptstyle proj_S} & \\ & & G_S & & \end{array}$$

(2) **source consistent**, *if there is a source TGT-sequence $\emptyset \overset{tr_S^*}{\Longrightarrow} G$ such that $\emptyset \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H$ is match consistent.*

Remark 3. For backward transformations the terms forward information preserving and target consistency are defined dually.

Theorem 3 (Information Preserving Forward Transformation)
A forward TGT-sequence $G \overset{tr_F^*}{\Longrightarrow} H$ is backward information preserving, if it is source consistent.

Proof. $G \overset{tr_F^*}{\Longrightarrow} H$ is source consistent which implies the existence of (2) $\emptyset \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H$ with $proj_S(G) = proj_S(H)$ being match consistent. By Theorem 2 with $G_0 = \emptyset$, $G_{n0} = G$, $G_{0n} = K$ and $G_n = H$ we obtain (3) $\emptyset \overset{tr_T^*}{\Longrightarrow} K \overset{tr_B^*}{\Longrightarrow} H' = H$ with $proj_T(K) = proj_T(H)$ being match consistent. Moreover, $proj_S(K) = proj_S(\emptyset) = \emptyset$ and the C-component of K is $\emptyset$ which implies $K = proj_T(H) = H_T$ leading to the diagram in Def. 7(1). Hence, $G \overset{tr_F^*}{\Longrightarrow} H$ is backward information preserving.

Remark 4. If $G \overset{tr_F^*}{\Longrightarrow} H$ is source consistent, then there is already a canonical backward transformation $H_T \overset{tr_B^*}{\Longrightarrow} H'$ with $H' = H$ and $H_T = proj_T(H)$

which is target consistent, i.e. $\emptyset \overset{tr_T^*}{\Longrightarrow} H_T \overset{tr_B^*}{\Longrightarrow} H'$ is match consistent. Vice versa, given a target consistent backward transformation there is a source consistent forward transformation according to Theorem 2. Similar results hold for backward TGT-sequences $K \overset{tr_B^*}{\Longrightarrow} H$.

Example 2 (backward information preserving CD2RDBM *model transformation sequence).* We consider a concrete forward transformation $G \Longrightarrow H$ from a given class model G to its extension H by the corresponding data base model (Fig. 5). The small class model in G (left part of Fig. 5) consists of two classes *Company* and *Person* with an association in between, and a third class *Customer* inheriting from class *Person*. Class *Customer* is equipped with an attribute. The transformation is performed by applying first the forward rules of rule *Class2Table* twice $(1, 2)$, and afterwards the forward rules of *SubClass2Table* (3), *PrimaryAttribute2Column* (4), *SetKey* (5), and *Association2FKey* (6) each once. In Fig. 5, the corresponding matches (1..6) of this sequence are indicated by contours.

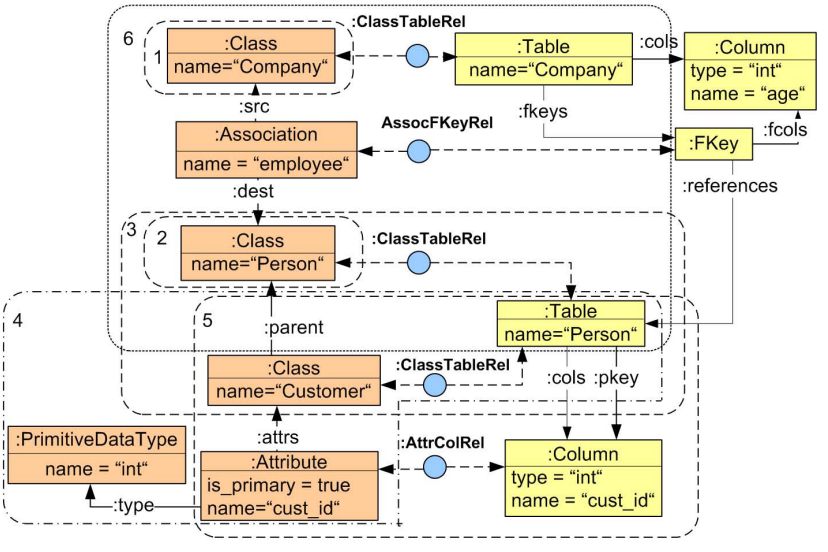


Fig. 5. Result graph of $C2RDBM$ forward transformation

This forward transformation is source consistent, since there is a transformation sequence $\emptyset \Longrightarrow G$. The co-matches of this source transformation sequence correspond to the matches of the forward transformation in Fig. 5, restricted to the source elements. It is easy to check that both transformation sequences are match consistent, i.e. the co-match of each source transformation step is not altered by forthcoming steps and is used again in its corresponding forward transformation step. Thus we can conclude from Theorem 3 that transformation $G \Longrightarrow H$ is backward information preserving, i.e. there is a backward

transformation from $proj_T(H) \Longrightarrow H'$ and the source graph of H' is equal to G. The backward transformation with the matches of the corresponding backward rules is shown in Fig. 6 where $proj_T(H)$ is given in the right part of Fig. 6, and H' is the complete result graph in Fig. 6.

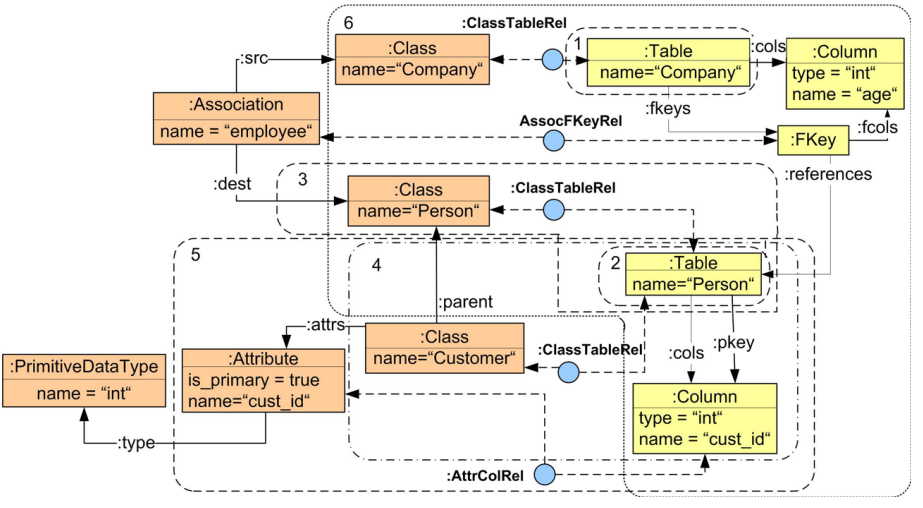


Fig. 6. Result graph of $C2RDBM$ backward transformation

5 General Theory of Triple Graph Transformations

In Section 2 we have introduced triple graphs and triple graph transformations based on simple graphs and graph morphisms (see Definition 1 - 3). In this section, we extend the concept to triple graphs based on typed, attributed and typed attributed graphs in the sense of [EEPT06]. WE can show that he corresponding categories are adhesive HLR categories. Thus, the general theory of adhesive HLR-systems in [EEPT06] can be instantiated by all these variants of triple graph transformations. That fact allows to obtain the well-known Local Church–Rosser and Concurrency Theorem for triple graph transformations which are used in a special case in the proof of Theorem 1. Further concepts and results which are presented in [EEPT06] and can be instantiated by triple graph transformation, include negative application conditions and critical pair analysis.

5.1 Triple Graph Transformations as Instantiation of Adhesive HLR Categories

Adhesive HLR categories and systems which are based on adhesive categories presented in [LS05], are a general categorical framework for several variants of graphs and graph transformation systems. One important instantiation of this framework are graph transformations based on simple graphs and graph

morphisms (as in Section 2) leading to the category **Graphs**. Another important instantiation are attributed graphs and attributed graph morphisms (as in [EPT04]) leading to the category **AGraphs**. Roughly speaking, attributed graphs $AG = (G, D)$ are pairs of graphs G and data type algebras D where some of the domains of D are carrying the attributes of graphs G. Rule graphs are attributed by a common term algebra such that left and right hand side graph items may have arbitrary terms as attributes. Category **TripleGraphs** consisting of triple graphs and triple graph morphisms (as in Section 2), can be constructed as a diagram category over **Graphs** and also becomes an adhesive HLR category (see Fact 4.18 in [EEPT06]).

Analogously, category **TripleAGraphs** of attributed triple graphs is a diagram category over **AGraphs**. Moreover, given type graphs TG in **TripleGraphs** (resp. ATG in **TripleAGraphs**) we obtain category $\mathbf{TripleGraphs_{TG}}$ consisting of typed triple graphs (resp. $\mathbf{TripleAGraphs_{ATG}}$ consisting of typed attributed triple graphs) as slice categories over **TripleGraphs** (resp. **TripleAGraphs**) leading again to adhesive HLR-categories.

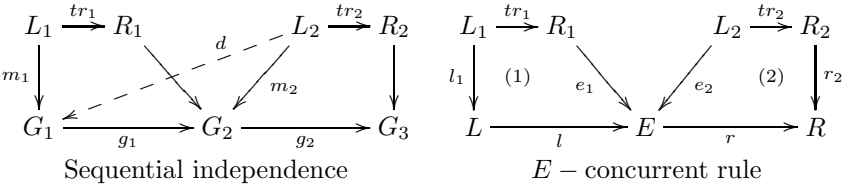
Theorem 4 (Adhesive HLR Categories for Triple Graph Transformations). *Categories* **TripleGraphs**, $\mathbf{TripleGraphs_{TG}}$, **TripleAGraphs**, *and* $\mathbf{TripleAGraphs_{ATG}}$ *together with suitable classes* $\mathcal{M}$ *of monomorphisms are adhesive HLR categories.*

Proof. According to Theorem 4.15 in [EEPT06], diagram and slice categories over adhesive HLR categories **Graphs** and **AGraphs** are again adhesive HLR categories.

This result implies that the general theory of adhesive HLR systems can be applied to triple graph transformations based on categories **TripleGraphs**, $\mathbf{TripleGraphs_{TG}}$, **TripleAGraphs**, and $\mathbf{TripleAGraphs_{ATG}}$. In the following, we use the abbreviation **Triple**, if we mean one of these categories.

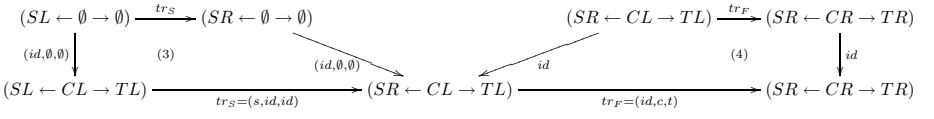
5.2 Proof of Theorem 1

Before proving Parts 1-3 of Theorem 1, we draw some conclusions from Theorem 4 above. From Theorem 5.12 in [EEPT06] for adhesive HRL-categories and Theorem 4 above we can conclude that the Local Church-Rosser Theorem is valid for each category **Triple**. We will use this result to show that "sequentially independent" steps $G_1 \overset{tr_1,m_1}{\Longrightarrow} G_2 \overset{tr_2,m_2}{\Longrightarrow} G_3$ can be commuted leading to $G_1 \overset{tr_2,m_2'}{\Longrightarrow} G_2' \overset{tr_1,m_1'}{\Longrightarrow} G_3$. Sequential independence means that there is a triple morphism $d : L_2 \to G_1$ with $g_1 \circ d = m_2$.



From Theorem 5.23 in [EEPT06] and Theorem 4 above we can conclude that the Concurrency Theorem is valid. This result is used for the construction of E-concurrent rule $tr = tr_1 *_E tr_2$ for triple rules tr_1 and tr_2 given. Triple graph E with triple graph morphisms e_1 and e_2 is constructed by pushouts (1) and (2) above and $tr = r \circ l$.

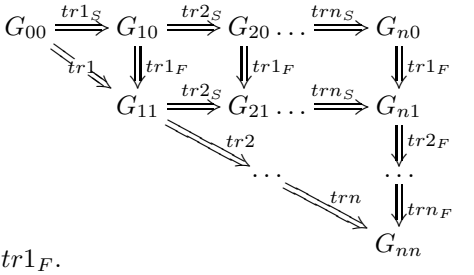
We will use the following construction: Given triple rule $tr : L \to R$ with source rule $tr_S : L_1 \to R_1$, forward rule $tr_F : L_2 \to R_2$, $E = L_2$, $e_1 = (id, \emptyset, \emptyset)$ and $e_2 = id$, we obtain $tr_S *_E tr_F = tr$, because diagrams (3) and (4) below are pushouts in **Triple** and $tr_F \circ tr_S = tr$. Hence, tr is equal to the E-concurrent rule $tr_S *_E tr_F$.



Proof of Theorem 1

1. Decomposition: Given TGT-sequence (1) $G_0 \overset{tr_1}{\Longrightarrow} G_1 \Longrightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n$ we first consider case $n = 1$. TGT-step $G_0 \overset{tr_1}{\Longrightarrow} G_1$ can be decomposed uniquely into a match consistent TGT-sequence $G_0 = G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \overset{tr1_F}{\Longrightarrow} G_{11} = G_1$. In fact we have shown above that $tr1$ can be represented as E-concurrent rule $tr1 = tr1_S *_E tr1_F$. Using the Concurrency Theorem the TGT-step $G_0 \overset{tr_1}{\Longrightarrow} G_1$ can be decomposed uniquely into an E-related sequence as given above. In this special case an E-relation is equivalent to the fact that the S-components of the co-match of $G_{00} \overset{tr1_S}{\Longrightarrow} G_{10}$ and the match of $G_{10} \overset{tr1_F}{\Longrightarrow} G_{11}$ coincide which corresponds exactly to match consistency.

Using this construction for $i = 1, \ldots, n$ the transformation sequence (1) can be decomposed canonically to an intermediate version between (1) and (2) called (1.5): $G_0 = G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \overset{tr1_F}{\Longrightarrow} G_{11} \overset{tr2_S}{\Longrightarrow} G_{21} \overset{tr2_F}{\Longrightarrow} G_{22} \Rightarrow \ldots \overset{trn_S}{\Longrightarrow} G_{n(n-1)} \overset{trn_F}{\Longrightarrow} G_{nn}$ where each subsequence $G_{(i-1)(i-1)} \overset{tri_S}{\Longrightarrow} G_{i(i-1)} \overset{tri_F}{\Longrightarrow} G_{ii}$ is match consistent. Moreover, $G_{10} \overset{tr1_F}{\Longrightarrow} G_{11} \overset{tr2_S}{\Longrightarrow} G_{21}$ is sequentially independent, because we have a morphism $d : L_2 \to G_{10}$, with $L_2 = (SL_2 \leftarrow \emptyset \to \emptyset)$ and $d = (m2_S, \emptyset, \emptyset)$. Morphism $m2 : L_2 \to G_{11}$ is the match of $G_{11} \overset{tr2_S}{\Longrightarrow} G_{21}$, because the S-components of G_{10} and G_{11} are equal according to forward rule $tr1_F$.



Now, the Local Church–Rosser Theorem mentioned above leads to an equivalent sequentially independent sequence $G_{10} \overset{tr2_S}{\Longrightarrow} G_{20} \overset{tr1_F}{\Longrightarrow} G_{21}$ such that $G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \overset{tr2_S}{\Longrightarrow} G_{20} \overset{tr1_F}{\Longrightarrow} G_{21} \overset{tr2_F}{\Longrightarrow} G_{22}$ is match consistent. The iteration of this shift between tri_F and trj_S leads to a shift-equivalent transformation sequence (2) $G_0 = G_{00} \overset{tr1_S}{\Longrightarrow} G_{10} \Rightarrow \cdots \overset{trn_S}{\Longrightarrow} G_{n0} \overset{tr1_F}{\Longrightarrow} G_{n1} \Rightarrow \cdots \overset{trn_F}{\Longrightarrow} G_{nn} = G_n$, which is still match consistent.

2. Composition: Vice versa, each match consistent transformation sequence (2) leads to a canonical sequence (1.5) by inverse shift equivalence where each subsequence as above is match consistent. In fact, match consistency of (2) implies that the corresponding subsequences are sequentially independent in order to allow inverse shifts in an order opposite to that in Part 1 using again the Local Church-Rosser Theorem. Match consistent subsequences of (1.5) are E-related as discussed in Part 1 which allows to apply the Concurrency Theorem to obtain the TGT-sequence (1).
3. Bijective Correspondence: The bijective correspondence of composition and decomposition is a direct consequence of the bijective correspondence in the Local Church–Rosser and the Concurrency Theorem where the bijective correspondence for the Local Church–Rosser Theorem is not explicitly formulated in Theorem 5.12 of [EEPT06], but is a direct consequence of the proof in analogy to Theorem 5.18. □

6 Related Work and Conclusion

In this paper we dealt with bi-directional model transformations, a promising technique in model-driven software development to keep related models consistent or to evolve them into other models or executable code. Bi-directional transformations can be defined using triple graph grammars which were introduced by Schürr [Sch94]. In [KS06], Königs and Schürr considered a set-theoretical formalization of triple graph transformations. We took up this formalization and extended it on the basis of category theory. To cope with the situation that tools do not necessarily keep object identifiers while changing models, we consider projections to source and target graphs in between. This makes the reversal of transformations more complex, but also more flexible due to less requirements on tools. We have shown that forward transformations are backward information preserving, if their source graph can be created by corresponding source rules.

In [KS06], a comprehensive comparison with related model transformation approaches, especially with bi-directional ones, is given. For example, BOTL [MB03] and QVT [OMG05] are discussed and compared to triple graph grammars. Although offering the concept of bi-directional transformation, sufficient conditions for the existence of information preserving transformations have not been given for these approaches.

In Section 5, we considered the extension of triple graph grammars to types and attributes. While these extensions are straightforward, the addition of application conditions to triple rules requires future investigations. From the practical point of view, there is also a request for multiple source and target models, considering activities like multi-model requirement engineering and the creation of a platform independent design model. The idea of triple graphs can be extended in a straightforward way by replacing the span by some arbitrary network of graphs. This approach has already been followed for defining viewpoint-oriented specifications on the basis of distributed graph transformation in [GEMT00].

References

[BRST05] Jean Bézivin, Bernhard Rumpe, Andy Schürr, and Laurence Tratt. Model transformations in practice workshop. In Jean-Michel Bruel, editor, *MoDELS Satellite Events*, volume 3844 of *Lecture Notes in Computer Science*, pages 120–127. Springer, 2005.

[CH03] K. Czarnecki and S. Helsen. Classification of model transformation approaches. In *On-line Proc. of the 2nd Workshop on Generative Techniques in the context of Model-Driven Architecture, Anaheim*, 2003.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer, 2006.

[EPT04] H. Ehrig, U. Prange, and G. Taentzer. Fundamental theory for typed attributed graph transformation. In F. Parisi-Presicce, P. Bottoni, and G. Engels, editors, *Proc. 2nd Int. Conference on Graph Transformation (ICGT'04), Rome, Italy*, volume 3256 of *Lecture Notes in Computer Science*. Springer, 2004.

[GEMT00] M. Goedicke, B. Enders, T. Meyer, and G. Taentzer. Tool Support for ViewPoint-Oriented Software Development: Towards Integration of Multiple Perspectives by Distributed Graph Transformation. In *Int. Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE'99), LNCS 1779*, pages 369 – 378. Springer, 2000.

[KS06] A. König and A. Schürr. Tool Integration with Triple Graph Grammars - A Survey. In *Heckel, R. (eds.): Elsevier Science Publ. (pub.), Proceedings of the SegraVis School on Foundations of Visual Modelling Techniques, Vol. 148, Electronic Notes in Theoretical Computer Science pp. 113-150, Amsterdam*, 2006.

[LS05] Stephen Lack and Pawel Sobociński. Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications*, 39(2):511–546, 2005.

[MB03] Frank Marschall and Peter Braun. Model transformations for the mda with botl. In *Proc. of the Workshop on Model Driven Architecture: Foundations and Applications (MDAFA 2003), Enschede, The Netherlands*, pages 25–36, 2003.

[MG06] T. Mens and P. Van Gorp. A taxonomy of model transformation. In *Proc. International Workshop on Graph and Model Transformation (GraMoT05), number 152 in Electronic Notes in Theoretical Computer Science, Tallinn, Estonia, Elsevier Science*, 2006.

[OMG05] OMG. *MOF QVT Final Adopted Specification (05-11-01). http://www.omg.org/docs/ptc/05-11-01.pdf*, 2005.

[Sch94] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In *G. Tinhofer, editor, WG94 20th Int. Workshop on Graph-Theoretic Concepts in Computer Science, volume 903 of Lecture Notes in Computer Science, pages 151–163, Springer Verlag, Heidelberg*, 1994.