# From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars

Hartmut Ehrig[1] , Karsten Ehrig[2] and Frank Hermann[1]

[1] [ehrig, frank](at)cs.tu-berlin.de
Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany

[2] karsten@mcs.le.ac.uk
Department of Computer Science
University of Leicester, United Kingdom

Bericht-Nr. 2008/03
ISSN 1436-9915

# From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars

Hartmut Ehrig[1], Karsten Ehrig[2] and Frank Hermann[1]

[1] [ehrig, frank](at)cs.tu-berlin.de
Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany

[2] karsten@mcs.le.ac.uk
Department of Computer Science
University of Leicester, United Kingdom

February 5, 2008

**Abstract**

Success and efficiency of software and system design fundamentally relies on its models. The more they are based on formal methods the more they can be automatically transformed to execution models and finally to implementation code. This paper presents model transformation and model integration as specific problem within bidirectional model transformation, which has shown to support various purposes, such as analysis, optimization, and code generation.

The main purpose of model integration is to establish correspondence between various models, especially between source and target models. From the analysis point of view, model integration supports correctness checks of syntactical dependencies between different views and models.

The overall concept is based on the algebraic approach to triple graph grammars, which are widely used for model transformation. The main result shows the close relationship between model transformation and model integration. For each model transformation sequence there is a unique model integration sequence and vice versa. This is demonstrated by a quasi-standard example for model transformation between class models and relational data base models.

**Keywords:** model transformation, model integration, syntactical correctness

## 1 Introduction

Whenever one can expect benefits out of different modeling languages for the same specific task there is a substantial motivation of combining at least two of the them. For this purpose it is useful to have model transformations between these modeling languages together with suitable analysis and verification techniques. In cases of bidirectional model transformation the support for the modeling process increases, for instance, if results of analysis can be translated backwards to mark the original source of deficiency or defect, respectively.

In [EEE$^+$07] Ehrig et al. showed how to analyze bi-directional model transformations based on triple graph grammars [Sch94, KS06] with respect to information preservation, which is especially important to ensure the benefits of other languages for all interesting parts of models. Triple graph grammars are based on triple rules, which allow to generate integrated models $G$ consisting of a source model $G_S$, a target model $G_T$ and a connection model $G_C$ together with correspondences from $G_C$ to $G_S$ and $G_T$. Altogether $G$ is a triple graph $G = (G_S \leftarrow G_C \rightarrow G_T)$. From each triple rule $tr$ we are able to derive a source rule $tr_S$ and a forward rule $tr_F$, such that the source rules are generating source models $G_S$ and the forward rules allow to transform a source model $G_S$ into its corresponding target model $G_T$ leading to a model transformation from source to target models. On the other hand we can also derive from each triple rule $tr$ a target rule $tr_T$ and a backward rule $tr_B$, such that the target rules are generating target models $G_T$ and backward rules transform target models to source models. The relationship between these forward and backward model transformation sequences was analyzed already in [EEE+07] based on a canonical decomposition and composition result for triple transformations.

In this paper we study the model integration problem: Given a source model $G_S$ and a target model $G_T$ we want to construct a corresponding integrated model $G = (G_S \leftarrow G_C \rightarrow G_T)$. For this purpose, we derive from each triple rule $tr$ an integration rule $tr_I$, such that the integration rules allow to define a model integration sequence from $(G_S, G_T)$ to $G$. Of course, not each pair $(G_S, G_T)$ allows to construct such a model integration sequence. In our main result we characterize existence and construction of model integration sequences sequences from $(G_S, G_T)$ to $G$ by model transformation sequences from $G_S$ to $G_T$. This main result is based on the canonical decomposition result mentioned above [EEE+07] and a new decomposition result of triple transformation sequences into source-target- and model integration sequences.

In Section 2 we review triple rules and triple graph grammars as introduced in [Sch94] and present as example the triple rules for model transformation and integration between class models and relational data base models. Model transformations based on our paper [EEE+07] are introduced in Section 3, where we show in addition syntactical correctness of model transformation. The main new part of this paper is model integration presented in Section 4 including the main results mentioned above and applied to our example. Related and future work are discussed in sections 5 and 6, respectively.

# 2 Review of Triple Rules and Triple Graph Grammars

Triple graph transformation [Sch94] has been shown to be a promising approach to consistently co-develop two related structures. Bidirectional model transformation can be defined using models consisting of a pair of graphs which are connected via an intermediate 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, which was generalized and extended by Ehrig et. el. in [EEE+07] to typed, attributed graphs. In this section, we shortly review main constructions and relevant results for model integration as given in [EEE+07].

**Definition 1** (Triple Graph and Triple Graph Morphism)**.** *Three graphs $SG$, $CG$, and $TG$, called source, connection, and target graphs, together with two graph morphisms $s_G : CG \rightarrow SG$ and $t_G : CG \rightarrow TG$ form a triple graph $G = (SG \overset{s_G}{\leftarrow} CG \overset{t_G}{\rightarrow} TG)$. $G$ is called empty, if $SG$, $CG$, and $TG$ are empty graphs.*

*A triple graph morphism $m = (s, c, t) : G \rightarrow H$ between two triple graphs $G = (SG \overset{s_G}{\leftarrow} CG \overset{t_G}{\rightarrow} TG)$ and $H = (SH \overset{s_H}{\leftarrow} CH \overset{t_H}{\rightarrow} TH)$ consists of three graph morphisms $s : SG \rightarrow SH$, $c : CG \rightarrow CH$ and $t : TG \rightarrow TH$ 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.*

Triple graphs $G$ are typed over a triple graph $TG = (TG_S \leftarrow TG_C \rightarrow TG_T)$ by a triple graph morphism $t_G : G \rightarrow TG$. Type graph of the example is given in Fig. 1 showing the structure of class diagrams in source component and relational databases in target component. Where classes are connected via associations the corresponding elements in databases are foreign keys. Though, the complete structure of correspondence elements between both types of models is defined via the connection component of $TG$. Throughout the example, originating from [EEE+07], elements are arranged left, center, and right according to the component types source, correspondence and target. Morphisms starting at a connection part are given by dashed arrow lines.

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,

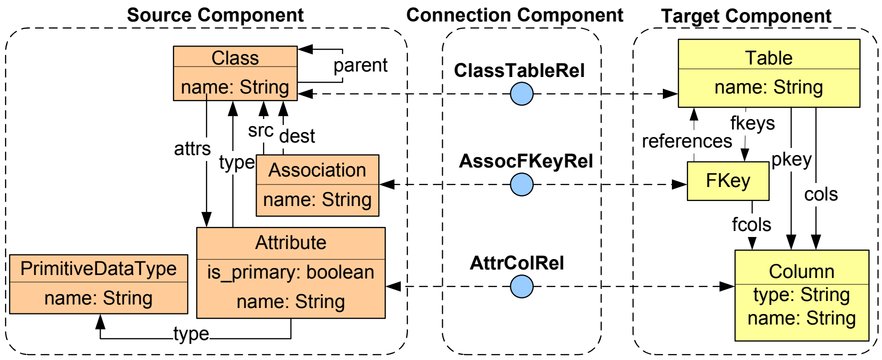



Figure 1: Triple type graph for *CD2RDBM* model transformation

is 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 2** (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 $tr = (s, c, t) : L \to R$.*

$$\begin{tikzcd}
L \arrow[d, "tr"'] & = & (SL \arrow[d, "s"] & CL \arrow[l, "s_L"'] \arrow[r, "t_L"] \arrow[d, "c"] & TL) \arrow[d, "t"] \\
R & = & (SR & CR \arrow[l, "s_R"] \arrow[r, "t_R"'] & TR)
\end{tikzcd}$$

*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) $G \xRightarrow{tr,m} H$ 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*

$$\begin{tikzcd}
 & & SL \arrow[dl, "sm"'] \arrow[dd, dotted] & & CL \arrow[ll] \arrow[rr] \arrow[dl, "cm"'] \arrow[dd, dotted] & & TL \arrow[dl, "tm"'] \arrow[dd] \\
G = \arrow[dd, Rightarrow, "tr"'] & (SG \arrow[dd, "s'"'] & & CG \arrow[ll] \arrow[rr] \arrow[dd, "c'"'] & & TG) \arrow[dd, "t'"'] & \\
 & & SR \arrow[dl, dotted, "sn"] & & CR \arrow[ll, dotted] \arrow[rr, dotted] \arrow[dl, dotted, "cn"] & & TR \arrow[dl, "tn"] \\
H = & (SH & & CH \arrow[ll, "s_H"] \arrow[rr, "t_H"'] & & TH) &
\end{tikzcd}$$

*morphisms $s_H : CH \to SH$ and $t_H : CH \to TH$. Morphism $n = (sn, cn, tn)$ is called comatch.*

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. Given a triple rule $tr$ we refer by $L(tr)$ to its left and by $R(tr)$ to its right hand side.

**Remark 1** (gluing construction)**.** *Each of the pushout objects $SH, CH, TH$ in Def. 2 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 $SL$.*

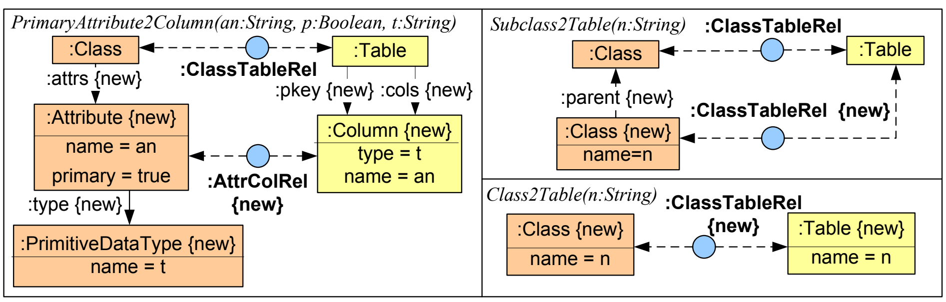



Figure 2: TGT-rules for *CD2RDBM* model transformation

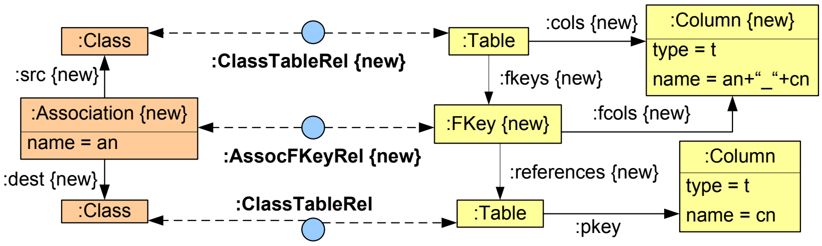



Figure 3: Rule *Association2ForeignKey*($an$ : *String*) for *CD2RDBM* model transformation

Examples for triple rules are given in Fig. 2 and Fig. 3 in short notation. Left and right hand side of a rule are depicted in one triple graph. Elements, which are created by the rule, are labeled with "new" and all other elements are preserved, meaning they are included in the left and right hand side. Rule "*Class2Table*" synchronously creates a class in a class diagram with its corresponding table in the relational database. Accordingly the other rules create parts in all components. For rule "*PrimaryAttribute2Column*" there is an analogous rule "*Attribute2Column*" for translation of non primary attributes, which does not add the edge ":pkey" in the database component.

# 3 Model transformation

The triple rules $TR$ are defining the language $VL = \{G \,|\, \emptyset \Rightarrow^* G \ via \ TR\}$ of triple graphs. As shown already in [Sch94] we can derive from each triple rule $tr = L \to R$ the following source and forward rule. Forward rules are used for model transformations from a model of a source language to models of the target language. Source rules are important for analyzing properties of forward transformations such as information preservation, presented in [EEE$^+$07].

$$
\begin{tikzcd}
L \arrow[d, "tr"'] & = & (SL \arrow[d, "s"'] & CL \arrow[l, "s_L"'] \arrow[r, "t_L"] \arrow[d, "c"'] & TL) \arrow[d, "t"] \\
R & = & (SR & CR \arrow[l, "s_R"] \arrow[r, "t_R"'] & TR)
\end{tikzcd}
$$

triple rule $tr$

$$
\begin{tikzcd}
(SR \arrow[d, "id"'] & CL \arrow[l, "s \circ s_L"'] \arrow[r, "t_L"] \arrow[d, "c"'] & TL) \arrow[d, "t"] \\
(SR & CR \arrow[l, "s_R"'] \arrow[r, "t_R"] & TR)
\end{tikzcd}
$$

forward rule $tr_F$

$$
\begin{tikzcd}
(SL \arrow[d, "s"'] & \emptyset \arrow[l] \arrow[r] \arrow[d] & \emptyset) \arrow[d] \\
(SR & \emptyset \arrow[l] \arrow[r] & \emptyset)
\end{tikzcd}
$$

source rule $tr_S$

For simplicity of notation we sometimes identify source rule $tr_S$ with $SL \xrightarrow{s} SR$ and target rule $tr_T$ with $TL \xrightarrow{t} TR$.

Theses rules can be used to define a model transformation from source graphs to target graphs. Vice versa using backward rules - which are dual to forward rules - it is also possible to define backward transformations from target to source graphs and altogether bidirectional model transformations. In [EEE$^+$07] we have shown that there is an equivalence between corresponding forward and backward TGT sequences. This equivalence is based on the canonical decomposition and composition result (Thm. 1) and its dual version for backward transformations.

**Definition 3** (Match Consistency). *Let $tr_S^*$ and $tr_F^*$ be sequences of source rules $tri_S$ and forward rules $tri_F$, which are derived from the same triple rules $tri$ for $i = 1, \ldots, n$. Let further $G_{00} \xRightarrow{tr_S^*} G_{n0} \xRightarrow{tr_F^*} G_{nn}$ be a* TGT*-sequence with $(mi_S, ni_S)$ being match and comatch of $tri_S$ (respectively $(mi, ni)$ for $tri_F$) then match consistency of $G_{00} \xRightarrow{tr_S^*} G_{n0} \xRightarrow{tr_F^*} G_{nn}$ means that the $S$-component of the match $mi$ is uniquely determined by the comatch $ni_S$ $(i = 1, \ldots, n)$.*

**Theorem 1** (Canonical Decomposition and Composition Result - Forward Rule Case)**.**

1. **Decomposition:** *For each* TGT*-sequence based on triple rules $tr^*$*
   (1) $G_0 \xRightarrow{tr^*} G_n$ *there is a canonical match consistent* TGT*-sequence*

(2) $G_0 = G_{00} \xRightarrow{tr_S^*} G_{n0} \xRightarrow{tr_F^*} G_{nn} = G_n$ *based on corresponding source rules* $tr_S^*$ *and forward rules* $tr_F^*$.

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.*

*Proof.* See [EEE+07]. □

Now we want to discuss under which conditions forward transformation sequences $G_1 \xRightarrow{tr_F^*} G_n$ define a model transformation between suitable source and target languages. In fact we have different choices: On the one hand we can consider the projections $VL_S = proj_S(VL)$ and $VL_T = proj_T(VL)$ of the triple graph language $VL = \{G \mid \emptyset \Rightarrow^* G \text{ via } TR\}$, where $proj_X$ is a projection defined by restriction to one of the triple components, i. e. $X \in \{S, C, T\}$. On the other hand we can use the source rules $TR_S = \{tr_S \mid tr \in TR\}$ and the target rules $TR_T = \{tr_T \mid tr \in TR\}$ to define the source language $VL_{S0} = \{G_S \mid \emptyset \Rightarrow^* G_S \text{ via } TR_S\}$ and the target language $VL_{T0} = \{G_T \mid \emptyset \Rightarrow^* G_T \text{ via } TR_T\}$. Since each sequence $\emptyset \Rightarrow^* G \text{ via } TR$ can be restricted to a source sequence $\emptyset \Rightarrow^* G_S$ via $TR_S$ and to a target sequence $\emptyset \Rightarrow^* G_T$ via $TR_T$ we have $VL_S \subseteq VL_{S0}$ and $VL_T \subseteq VL_{T0}$, but in general no equality. In case of typed graphs the rules in $TR$ are typed over $TG$ with $TG = (TG_S \leftarrow TG_C \rightarrow TG_T)$ and rules of $TR_S$ and $TR_T$ typed over $(TG_S \leftarrow \emptyset \rightarrow \emptyset)$ and $(\emptyset \leftarrow \emptyset \rightarrow TG_T)$, respectively. Since $G_S$ and $G_T$ are considered as plain graphs they are typed over $TG_S$ and $TG_T$, respectively.

Given a forward transformation sequence $G_1 \xRightarrow{tr_F^*} G_n$ we want to ensure the source component of $G_1$ corresponds to the target component of $G_n$, i.e. the transformation sequence defines a a model transformation $MT$ from $VL_{S0}$ to $VL_{T0}$, written $MT : VL_{S0} \Rrightarrow VL_{T0}$, where all elements of the source component are translated. Thus given a class diagram as instance of the type graph in Fig. 1 all corresponding tables, columns and foreign keys of the corresponding data base model shall be created in the same way they could have been synchronously generated by the triple rules of $TR$. An example forward transformation is presented in [EEE+07]. Since $G_S \in VL_{S0}$ is generated by $TR_S$-rules we have a source transformation $\emptyset \Rightarrow^* G_S$ via $TR_S$. In order to be sure that $G_1 \xRightarrow{tr_F^*} G_n$ transforms all parts of $G_1$, which are generated by $\emptyset \Rightarrow^* G_S$, we require that $\emptyset \Rightarrow^* G_S$ is given by $\emptyset \xRightarrow{tr_S^*} G_1$ with $G_1 = (G_S \leftarrow \emptyset \rightarrow \emptyset)$, i.e. $proj_S(G_1) = G_S$ based on the same triple rule sequence $tr^*$ as $G_1 \xRightarrow{tr_F^*} G_n$. Finally we require that the TGT-sequence $\emptyset \xRightarrow{tr_S^*} G_1 \xRightarrow{tr_F^*} G_n$ is match consistent, because this implies – by Fact 1 below – that $G_S \in VL_S$ and $G_T \in VL_T$ and that we obtain a model transformation $MT : VL_S \Rrightarrow VL_T$ (see Fact 1).

**Definition 4** (Model Transformation)**.** *A model transformation sequence* $(G_S, G_1 \xRightarrow{tr_F^*} G_n, G_T)$ *consists of a source graph* $G_S$*, a target graph* $G_T$*, and a source consistent forward* TGT*-sequence* $G_1 \xRightarrow{tr_F^*} G_n$ *with* $G_S = proj_S(G_1)$ *and* $G_T = proj_T(G_n)$.
*Source consistency of* $G_1 \xRightarrow{tr_F^*} G_n$ *means that there is a source transformation sequence* $\emptyset \xRightarrow{tr_S^*} G_1$*, such that* $\emptyset \xRightarrow{tr_S^*} G_1 \xRightarrow{tr_F^*} G_n$ *is match consistent. A model transformation* $MT : VL_{S0} \Rrightarrow VL_{T0}$ *is defined by model transformation sequences* $(G_S, G_1 \xRightarrow{tr_F^*} G_n, G_T)$ *with* $G_S \in VL_{S0}$ *and* $G_T \in VL_{T0}$.

**Remark 2.** *A model transformation* $MT : VL_{S0} \Rrightarrow VL_{T0}$ *is a relational dependency and only in special cases a function.*

This allows to show that $MT : VL_{S0} \Rrightarrow VL_{T0}$ defined above is in fact $MT : VL_S \Rrightarrow VL_T$

**Fact 1** (Syntactical Correctness of Model Transformation $MT$)**.** *Given* $G_S \in VL_{S0}$ *and* $G_1 \xRightarrow{tr_F^*} G_n$ *source consistent with* $proj_S(G_1) = G_S$ *then* $G_T = proj_T(G_n) \in VL_T$ *and* $G_S \in VL_S$*, i.e.* $MT : VL_S \Rrightarrow VL_T$.

*Proof.* Given $G_1 \xRightarrow{tr_F^*} G_n$ source consistent, we have $\emptyset \xRightarrow{tr_S^*} G_1 \xRightarrow{tr_F^*} G_n$ match consistent and hence, by Theorem 1 above with $G_0 = \emptyset \xRightarrow{tr^*} G_n$ which implies $G_n \in VL$. Now we have $proj_S(G_n) = proj_S(G_1) = G_S \in VL_S$ and $proj_T(G_n) = G_T \in VL_T$. □

# 4 Model Integration

Given models $G_S \in VL_{S0}$ and $G_T \in VL_{T0}$ the aim of model integration is to construct an integrated model $G \in VL$, such that $G$ restricted to source and target is equal to $G_S$ and $G_T$, respectively, i.e. $proj_S G = G_S$ and $proj_T G = G_T$. Thus, given a class diagram and a data base model as instance of the type graph in Fig. 1 all correspondences between their elements shall be recovered or detected, respectively. Similar to model transformation we can derive rules for model integration based on triple rule $tr$. The derived rules are source-target rule $tr_{ST}$ and integration rule $tr_I$ given by

$$\begin{tikzcd} (SL \arrow[d, "s"'] & CL \arrow[l, "s_L"'] \arrow[r, "t_L"] \arrow[d, "c"] & TL) \arrow[d, "t"] \\ (SR & CR \arrow[l, "s_R"] \arrow[r, "t_R"'] & TR) \end{tikzcd}$$

triple rule $tr$

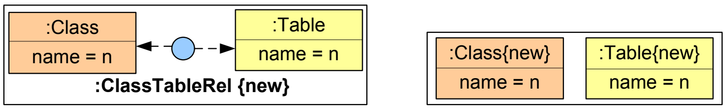
$$\begin{tikzcd} (SL \arrow[d, "s"'] & \emptyset \arrow[l] \arrow[r] \arrow[d] & TL) \arrow[d, "t"'] \\ (SR & \emptyset \arrow[l] \arrow[r] & TR) \end{tikzcd}$$

source-target rule $tr_{ST}$

$$\begin{tikzcd} (SR \arrow[d, "id"'] & CL \arrow[l, "s \circ s_L"'] \arrow[r, "t \circ t_L"] \arrow[d, "c"'] & TR) \arrow[d, "id"] \\ (SR & CR \arrow[l, "s_R"'] \arrow[r, "t_R"] & TR) \end{tikzcd}$$

integration rule $tr_I$

An example for both kinds of rules is given in Fig. 4 for the triple rule *Class2Table* in Fig. 2.




(a) integration rule $Class2Table_I$ (b) source-target rule $Class2Table_{ST}$

Figure 4: Derived rules for $Class2Table()$

Similar to the canonical decomposition of TGT-sequences $G_0 \xRightarrow{tr^*} G_n$ into source and forward transformation sequences we also have a canonical decomposition into source-target and integration transformation sequences of the form $\emptyset \xRightarrow{tr^*_{ST}} G_0 \xRightarrow{tr^*_I} G_n$. Such a sequence is called $S$-$T$-consistent, if the $S$- and $T$-component of the comatch of $tri_{ST}$ is completely determined by that of the match of $tri_I$ for $tr = (tri)_{i=1\ldots n}$.

**Theorem 2** (Canonical Decomposition and Composition Result - Integration Rule Case)**.**

1. **Decomposition:** *For each* TGT*-sequence based on triple rules* $tr^*$

   (1) $G_0 \xRightarrow{tr^*} G_n$ *there is a canonical* $S$-$T$*-match consistent* TGT*-sequence*

   (2) $G_0 = G_{00} \xRightarrow{tr^*_{ST}} G_{n0} \xRightarrow{tr^*_I} G_{nn} = G_n$ *based on corresponding source-target rules* $tr^*_{ST}$ *and integration rules* $tr^*_I$.

2. **Composition:** *For each* $S$-$T$*-match consistent transformation sequence (2) there is a canonical transformation sequence (1).*

3. **Bijective Correspondence:** *Composition and Decomposition are inverse to each other.*

In the following we give the proof of Theorem 2 which is based on the Local-Church-Rosser and the Concurrency Theorem for algebraic graph transformations (see [Roz97], [EEPT06]). In Lemma 1 we show that a triple rule $tr$ can be represented as concurrent production $tr_{ST} *_E tr_I$ of the corresponding source-target rule $tr_{ST}$ and integration rule $tr_I$, where the overlapping $E$ is equal to $L(tr_I)$, the left hand side of $tr_I$. Moreover $E$-related sequences in the sense of the Concurrency Theorem correspond exactly to $S$-$T$-match-consistent sequences in Theorem 2. In Lemma 2 we show compatibility of $S$-$T$-match consistency with sequential independence in the sense of the Local-Church-Rosser-Theorem. Using Lemma 1 we can decompose a single TGT-transformation $G_0 \xRightarrow{tr} G_1$ into an $S$-$T$-match consistent sequence $G_0 \xRightarrow{tr_{ST}} G_{10} \xRightarrow{tr_I} G_1$ and vice versa. Lemma 2 allows to decompose TGT-sequences $G_0 \xRightarrow{tr^*} G_n$ into $S$-$T$-match consistent sequences $G_0 \xRightarrow{tr^*_{ST}} G_{n0} \xRightarrow{tr^*_I} G_n$ and vice versa.

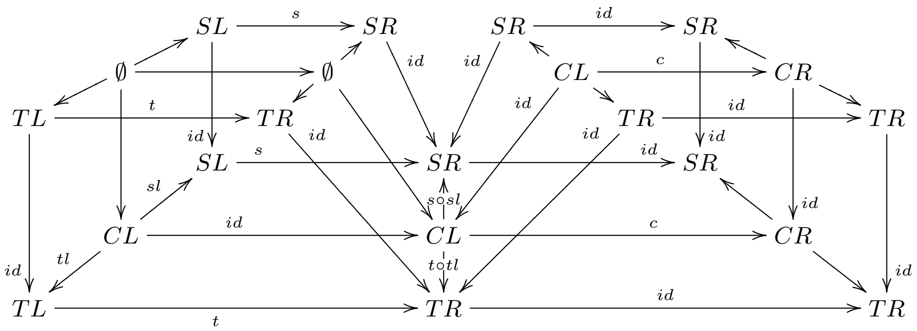
All constructions are done in the category $\mathbf{TripleGraph_{TG}}$ of typed triple graphs and typed triple graph morphisms, which according to Fact 4.18 in [EEPT06] is an adhesive HLR category. This implies that the Local-Church-Rosser and Concurrency Theorem are valid for triple rules with injective morphisms (see Chapter 5 in [EEPT06]).

**Lemma 1** (Concurrent Production $tr = tr_{ST} *_E tr_I$)**.** *Let $E = L(tr_I)$ with $e1 = (id, \emptyset, id) : R(tr_{ST}) \to E$ and $e2 = id : L(tr_I) \to E$ then $tr$ is given by the concurrent production $tr = tr_{ST} *_E tr_I$. Moreover, there is a bijective correspondence between a transformation $G_1 \xRightarrow{tr,m} G_2$ and match-consistent sequences $G_1 \xRightarrow{tr_{ST},m1,n1} H \xRightarrow{tr_I,m2,n2} G_2$, where $S-T$-match consistency means that the $S-$ and $T-$components of the comatch $n1$ and the match $m2$ are equal, i.e. $n1_S = m2_S$ and $n1_T = m2_T$. Construction of concurrent production:*

$$\begin{tikzcd}
L(tr_{ST}) \arrow[r, "tr_{ST}"] \arrow[d, "l"'] \arrow[dr, phantom, "(1)"] & R(tr_{ST}) \arrow[dr, "e1"] & & L(tr_I) \arrow[r, "tr_I"] \arrow[dl, "e2"'] \arrow[dr, phantom, "(2)"] & R(tr_I) \arrow[d, "r"] \\
L(tr) \arrow[rr, "d1"'] & & E \arrow[rr, "d2"'] & & R
\end{tikzcd}$$

*$E-$ concurrent rule*

*Proof.* The pushouts in (1) in $\mathbf{TripleGraph_{TG}}$ are given below showing $d2 \circ d1 = tr : L(tr) \to R(tr)$




$E-$ concurrent rule $tr : L(tr) \to R(tr)$

According to the Concurrency Theorem for $\mathbf{TripleGraph_{TG}}$ there is a bijective correspondence between transformations $G_1 \xRightarrow{tr,m} G_2$ and $E$-related sequences $G_1 \xRightarrow{tr_{ST},m1,n1} H \xRightarrow{tr_I,m2,n2} G_2$. The sequence is $E$-related if there is $h : E \to H$ with $h \circ e1 = n1$ and $h \circ e2 = m2$ and there are $c1 : L(tr) \to G_1$ and $c2 : R(tr) \to G_2$, s.t. $c1 \circ l = m1, c2 \circ r = n2$ and (3) as well as (4) in the following diagram are pushouts.

$$\begin{tikzcd}
L(tr_{ST}) \arrow[rr, "tr_{ST}"] \arrow[dd, "m1"'] \arrow[dr, "l"] & & R(tr_{ST}) \arrow[dr, "e1"] \arrow[d, phantom, "(1)"] & & L(tr_I) \arrow[rr, "tr_I"] \arrow[dl, "e2"'] & & R(tr_I) \arrow[dd, "n2"] \arrow[dl, "r"'] \arrow[dll, phantom, "(2)"] \\
 & L(tr) \arrow[rr, "d1"] \arrow[dl, "c1"'] \arrow[drr, phantom, "(3)"] & & E \arrow[rr, "d2"] \arrow[d, "h"] \arrow[drr, phantom, "(4)"] & & R(tr) \arrow[dr, "c2"] & \\
G_1 \arrow[rrr, "d1"'] & & & H \arrow[rrr, "d2"'] & & & F_2
\end{tikzcd}$$

$E-$ concurrent rule

First of all we observe that the sequence is $S$-$T$-match consistent, i.e. $n1_S = m2_S$ and $n1_T = m2_T$ iff there is $h : E \to H$ with $h \circ e1 = n1$ and $h \circ e2 = m2$. In case of $n1_S = m2_S$ and $n1_T = m2_T$ we define $h$ by $h = m2$, which implies $h \circ e2 = h \circ id = m2$, but also $h \circ e1 = h \circ n1$, because $h_S \circ e1_S = m2_S \circ id = n1_S, h_T \circ e1_T = m2_T \circ id = n1_T$ and $h_C \circ e1_C = \emptyset = n1_C$. Vice versa, given $h$ with $h \circ e1 = n1$, $h \circ e2 = m2$ we have $n1_S = h_S \circ e1_S = h_S \circ id = h_S \circ e2_S = m2_S$ and similar $n1_T = m2_T$. In order to have an $E$-related sequence we have to show that $h$ induces $c1$ and $c2$ with $c1 \circ l = n1$ and $c2 \circ r = n2$, st. (3) and (4) become pushouts. This follows for $c2$ and (4) directly from pushout (2) and pushout decomposition. For $c1$ and (3), however, we need pushout-pullback decomposition, which would require that $h$ is injective. In order to avoid the assumption $h$ injective we give now a direct construction for $c1$ with $c1 \circ l = n1$ and pushout (3) by $c1 = (m1_S, h_C, m1_T)$ according to the following diagrams

$$
\begin{tikzcd}
SL \arrow[rr, "s"] \arrow[dr, "id"] \arrow[ddd, "m1_S"'] & & SR \arrow[dr, "id"] & \\
& SL \arrow[rr, "s"] \arrow[ddl, "c1_S=m1_S"'] \arrow[r, phantom, "(1)_S"] \arrow[dr, phantom, "(3)_S"] & & SR \arrow[dd, "h_S"] \\
& & & \\
SG_1 \arrow[rrr, "g1_S"'] & & & SH
\end{tikzcd}
\qquad
\begin{tikzcd}
\emptyset \arrow[rr] \arrow[dr] \arrow[ddd, "m1_C"'] & & \emptyset \arrow[dr] & \\
& CL \arrow[rr, "id"] \arrow[ddl, "c1_C=h_C"'] \arrow[r, phantom, "(1)_C"] \arrow[dr, phantom, "(3)_C"] & & CL \arrow[dd, "h_C"] \\
& & & \\
CG_1 \arrow[rrr, "g1_C=id"'] & & & CH = CG_1
\end{tikzcd}
\qquad
\begin{tikzcd}
TL \arrow[rr, "t"] \arrow[dr, "id"] \arrow[ddd, "m1_T"'] & & TR \arrow[dr, "id"] & \\
& TL \arrow[rr, "t"] \arrow[ddl, "c1_T=m1_T"'] \arrow[r, phantom, "(1)_T"] \arrow[dr, phantom, "(3)_T"] & & TR \arrow[dd, "h_T"] \\
& & & \\
TG_1 \arrow[rrr, "g1_T"'] & & & TH
\end{tikzcd}
$$

Note that the outer diagrams are the $S$-, $C$- and $T$- component of the pushout according to $G_1 \xRightarrow{tr_{ST},m1,n1} H$ with $n1 = h \circ e1$, where we assume w.l.o.g. $g1_C = id$ induced by $\emptyset \xrightarrow{id} \emptyset$ in the $C$-component. The outer diagrams in the $S$- and $T$-component are pushouts and equal to $(3)_S$ and $(3)_T$ respectively. $(3)_C$ is a trivial pushout. It remains to show that $c1$ is a morphism in $\mathbf{TripleGraph_{TG}}$. This follows from componentwise commutativity of (3) and the fact that $d1, h$ and $g1$ are in $\mathbf{TripleGraph_{TG}}$ by construction and $g1_S, g1_T$ injective, because $tr_{ST}$ injective implies $g$ injective. In more detail, $g1_S, g1_T$ injective implies commutativity of the left squares below showing that $c1$ is in $\mathbf{TripleGraph_{TG}}$.

$$
\begin{tikzcd}
& & SL \arrow[rrr, "s"] \arrow[ddd, "m1_S"'] & & & SR \arrow[ddd, "h_S"] \\
& CL \arrow[ur, "sl"] \arrow[dl, "tl"'] \arrow[rrr, "id"] \arrow[ddd, "h_C"'] & & & CL \arrow[ur, "s \circ sl"'] \arrow[dl, "t \circ tl"] \arrow[ddd, "h_C"'] & \\
TL \arrow[rrr, "t"] \arrow[ddd, "m1_T"'] & & & TR \arrow[ddd, "h_T"'] & & \\
& & SG_1 \arrow[rrr, "g1_S"] & & & SH \\
& CG_1 \arrow[ur] \arrow[dl] \arrow[rrr, "g1_C=id"] & & & CH = CG_1 \arrow[ur] \arrow[dl] & \\
TG_1 \arrow[rrr, "g1_T"] & & & TH & &
\end{tikzcd}
$$

□

**Lemma 2** (Compatibility of $S-T$-match consistency with independence).
*Given the* TGT*-sequences on the right with independence in* (4) *and matches $m_i, m'_i$ and comatches $n_i, n'_i$. Then we have:*

$$
\begin{tikzcd}
& & G_{20} \arrow[dr, Rightarrow, "tr1_I", "m1',n1'"'] & & \\
G_{00} \arrow[r, Rightarrow, "tr1_{ST}", "m0,n0"'] & G_{10} \arrow[ur, Rightarrow, "tr2_{ST}", "m2',n2'"'] \arrow[dr, Rightarrow, "tr1_I", "m1,n1"'] & & G_{21} \arrow[r, Rightarrow, "tr2_I", "m3,n3"'] & G_{22} \\
& & G_{11} \arrow[ur, Rightarrow, "tr2_{ST}", "m2,n2"'] & &
\end{tikzcd}
$$

(1) $G_{00} \xRightarrow{tr1_{ST}} G_{10} \xRightarrow{tr1_I} G_{11}$ *$S-T$-match consistent* $\Leftrightarrow$
(2) $G_{00} \xRightarrow{tr1_{ST}} G_{10} \xRightarrow{tr2_{ST}} G_{20} \xRightarrow{tr1_I} G_{21}$ *$S-T$-match consistent*
*and*
(3) $G_{11} \xRightarrow{tr2_{ST}} G_{21} \xRightarrow{tr2_I} G_{22}$ *$S-T$-match consistent* $\Leftrightarrow$
(4) $G_{10} \xRightarrow{tr2_{ST}} G_{20} \xRightarrow{tr1_I} G_{21} \xRightarrow{tr2_I} G_{22}$ *$S-T$-match consistent*

*Proof.* By independence we have $d : L(tr2_{ST}) \to G_{10}$ with $g2 \circ d = n2$ leading to $g3 \circ n2' = n2$ and $m1' = g1 \circ n1$.

$$
\begin{tikzcd}
 & L(tr2_{ST}) \arrow[r, "tr2_{ST}"] \arrow[d, "m2'=d"'] \arrow[dd, bend left, "m2"] & R(tr2_{ST}) \arrow[d, "n2'"'] \arrow[dd, bend left, "n2"] \\
L(tr1_I) \arrow[r, "m1"'] \arrow[d, "tr1_I"'] \arrow[rr, bend left, "m1'"] \arrow[dr, phantom, "(PO)"] & G_{10} \arrow[r, "g1"'] \arrow[d, "g2"'] \arrow[dr, phantom, "(PO)"] & G_{20} \arrow[d, "g3"'] \\
R(tr1_I) \arrow[r, "n1"] \arrow[rr, bend right, "n1'"'] & G_{11} \arrow[r, "g4"] & G_{21}
\end{tikzcd}
$$

(1) $S-T$-match consistent $\Leftrightarrow n0_S = m1_S$ and $n0_T = m1_T$
(2) $S-T$-match consistent $\Leftrightarrow g1_S \circ n0_S = m1'_S$ and $g1_T \circ n0_T = m1'_T$
(1) $\Rightarrow$ (2) : $g1_S \circ n0_S \overset{(1)}{=} g1_S \circ m1_S = m1'_S$ (and similar for $T$-component)
(2) $\Rightarrow$ (1) : $g1_S \circ m1_S = m1'_S \overset{(2)}{=} g1_S \circ n0_S$ (and similar for $T$-component)
(3) $S-T$-match consistent $\Leftrightarrow n2_S = m3_S$ and $n2_T = m3_T$
(4) $S-T$-match consistent $\Leftrightarrow g3_S \circ n2'_S = m3_S$ and $g3_T \circ n2_T = m3'_T$
(3) $\Rightarrow$ (4) : $g3_S \circ n2'_S \overset{(3)}{=} m3_S$ (and similar for $T$-component)
(4) $\Rightarrow$ (3) : $m3_S \overset{(4)}{=} g3_S \circ n2'_S$ (and similar for $T$-component)

$\square$

*Proof of Theorem 2.*

*1. Decomposition:* Given (1) we obtain (for $n = 3$) by Leamma 1 a decomposition into triangles $(1), (2), (3)$, where the corresponding transformation sequences are $S-T$-match consistent.

$$
\begin{tikzcd}[column sep=small]
 & & & G_{30} \arrow[dr, Rightarrow, "tr1_I"] \arrow[d, phantom, "(6)"] & & & \\
 & & G_{20} \arrow[ur, Rightarrow, "tr3_{ST}"] \arrow[dr, Rightarrow, "tr1_I"] \arrow[d, phantom, "(4)"] & & G_{31} \arrow[dr, Rightarrow, "tr2_I"] \arrow[d, phantom, "(5)"] & & \\
 & G_{10} \arrow[ur, Rightarrow, "tr2_{ST}"] \arrow[dr, Rightarrow, "tr1_I"] \arrow[d, phantom, "(1)"] & & G_{21} \arrow[ur, Rightarrow, "tr3_{ST}"] \arrow[dr, Rightarrow, "tr2_I"] \arrow[d, phantom, "(2)"] & & G_{32} \arrow[dr, Rightarrow, "tr3_I"] \arrow[d, phantom, "(3)"] & \\
G_0 = G_{00} \arrow[ur, Rightarrow, "tr1_{ST}"] \arrow[rr, Rightarrow, "tr1"'] & & G_{11} \arrow[ur, Rightarrow, "tr2_{ST}"] \arrow[rr, Rightarrow, "tr2"'] & & G_{22} \arrow[ur, Rightarrow, "tr3_{ST}"] \arrow[rr, Rightarrow, "tr3"'] & & G_{33} = G_3
\end{tikzcd}
$$

In the next step we show that $G_{10} \xRightarrow{tr1_I} G_{11} \xRightarrow{tr2_{ST}} G_{21}$ is sequentially independent leading by the Local Church Rosser Theorem to square (4) sequential independence in this case means existence of $d : L(tr2_{ST}) \to G_{10}$ with $g \circ d = m2$.

$$
\begin{tikzcd}
L(tr1_I) \arrow[r, "tr1_I"] \arrow[d, "m_1"'] & R(tr1_I) \arrow[dr] & & L(tr2_{ST}) \arrow[r] \arrow[dl, "m_2"] \arrow[dlll, dashed, "d"'] & R(tr2_{ST}) \arrow[d] \\
G_1 \arrow[rr, "g"'] & & G_2 \arrow[rr] & & G_3
\end{tikzcd}
$$

The diagram beneath shows that $d = (d_S, d_C, d_T) = (m2_S, \emptyset, m2_T)$ satisfies this property. $(1) - (4)$ leads to the following transformation sequence $G_{00} \xRightarrow{tr1_{ST}} G_{10} \xRightarrow{tr2_{ST}} G_{20} \xRightarrow{tr1_I} G_{21} \xRightarrow{tr2_I} G_{22} \xRightarrow{tr3_{ST}} G_{32} \xRightarrow{tr3_I} G_{33}$ which is again $S-T$-match consistent due to shift equivalence of corresponding matches in the Local Church Rosser Theorem (see Lemma 2). Similar to above we can show that $G_{21} \xRightarrow{tr2_I} G_{22} \xRightarrow{tr3_{ST}} G_{32}$ are sequentially independent leading to (5) and in the next step to (6) with corresponding $S-T$-match consistent sequences.

$$\begin{tikzcd}
& & SL2 \arrow[dddll, "d_S=m2_S"'] \arrow[dddrr, "m2_S"] & & & \\
& & \emptyset \arrow[u] \arrow[d] \arrow[dddll, "d_C"'] \arrow[dddrr, "m2_C"] & & & \\
& & TL2 \arrow[dddll, "d_T"'] \arrow[dddrr, "m2_T"] & & & \\
G_{10,S} \arrow[rrrr, "id"'] & & & & G_{11,S} & = G_{10,S} \\
G_{10,C} \arrow[u] \arrow[d] \arrow[rrrr, "g_C"'] & & & & G_{11,C} \arrow[u] \arrow[d] & \\
G_{10,T} \arrow[rrrr, "id"'] & & & & G_{11,T} & = G_{10,T}
\end{tikzcd}$$

*2. Composition:* Vice versa, each $S-T$-match consistent sequence (2) leads to a canonical $S-T$-match consistent sequence of triangles $(1),(2),(3)$ and later by Lemma 1 to TGT-sequence (1). We obtain the triangles by inverse shift equivalence, where subsequence 1 as above is $S-T$-match consistent. In fact $S-T$-match consistency of (2) together with Lemma 2 implies that the corresponding sequences are sequentially independent in order to allow inverse shifts according to the Local Church Rosser Theorem. Sequential independence for (6) is shown below

$$\begin{tikzcd}
& & & & SR_1 \arrow[d, equal] & \\
SR_1 = R(tr1_{ST}) \arrow[d, "n1"'] & L(tr3_{ST}) \arrow[r, "tr1_I"] \arrow[d, "m_3"'] & R(tr3_{ST}) \arrow[dr] & & L(tr1_I) \arrow[r] \arrow[dl, "m1_I"] \arrow[dlll, dashed, "d"'] & R(tr1_I) \arrow[d] \\
G_{10} \arrow[r, "g_1"'] & G_{20} \arrow[rr, "g_2"'] & & G_{30} \arrow[rr] & & G_{31}
\end{tikzcd}$$

By $S-T$-match consistency we have $m1_{I,S} = g2_S \circ g1_S \circ n1_S$. Define $d_S = g1_S \circ n1_S$, then $g2_S \circ d_S = g2_S \circ g1_S \circ n1_S = m1_{I,S}$ and similar for the $T$-component, while $d_C = m1_{I,C}$ using $g2_C = id$. *3. Bijective Correspondence:* by that of the Local Church Rosser Theorem and Concurrency Theorem. $\square$

Given an integration transformation sequence $G_0 \xRightarrow{tr_I^*} G_n$ with $proj_S(G_0) = G_S$, $proj_T(G_0) = G_T$ and $proj_C(G_0) = \emptyset$, we want to make sure that the unrelated pair $(G_S, G_T) \in VL_{S0} \times VL_{T0}$ is transformed into an integrated model $G = G_n$ with $proj_S(G) = G_S, proj_T(G) = G_T$. Of course this is not possible for all pairs $(G_S, G_T) \in VL_{S0} \times VL_{T0}$, but only for specific pairs. In any case $(G_S, G_T) \in VL_{S0} \times VL_{T0}$ implies that we have a source-target transformation sequence $\emptyset \Rightarrow^* G_0$ via $TR_{ST} = \{tr_{ST} \mid tr \in TR\}$. In order to be sure that $G_0 \xRightarrow{tr_I^*} G_n$ integrates all parts of $G_S$ and $G_T$, which are generated by $\emptyset \Rightarrow^* G_0$, we require that $\emptyset \Rightarrow^* G_0$ is given by $\emptyset \xRightarrow{tr_{ST}^*} G_0$ based on the same triple rule sequence $tr^*$ as $G_0 \xRightarrow{tr_I^*} G_n$. Moreover, we require that the TGT-sequence $\emptyset \xRightarrow{tr_{ST}^*} G_0 \xRightarrow{tr_I^*} G_n$ is $S$-$T$-match consistent because this implies - using Theorem 2 - that $G_S \in VL_S, G_T \in VL_T$ and $G \in VL$ (see Theorem 2).

**Definition 5** (Model Integration)**.** *A model integration sequence* $((G_S, G_T), G_0 \xRightarrow{tr_I^*} G_n, G)$ *consists of a source and a target model* $G_S$ *and* $G_T$*, an integrated model* $G$ *and a source-target consistent* TGT*-sequence* $G_0 \xRightarrow{tr_I^*} G_n$ *with* $G_S = proj_S(G_0)$ *and* $G_T = proj_T(G_0)$.
*Source-target consistency of* $G_0 \xRightarrow{tr_I^*} G_n$ *means that there is a source-target transformation sequence* $\emptyset \xRightarrow{tr_{ST}^*} G_0$*, such that* $\emptyset \xRightarrow{tr_{ST}^*} G_0 \xRightarrow{tr_I^*} G_n$ *is match consistent. A model integration* $MI : VL_{S0} \times VL_{T0} \Rrightarrow VL$ *is defined by model integration sequences* $((G_S, G_T), G_0 \xRightarrow{tr_I^*} G_n, G)$ *with* $G_S \in VL_{S0}$*,* $G_T \in VL_{T0}$ *and* $G \in VL$.

**Remark 3.** *Given model integration sequence* $((G_S, G_T), G_0 \xRightarrow{tr_I^*} G_n, G)$ *the corresponding source-target* TGT*-sequence* $\emptyset \xRightarrow{tr_{ST}^*} G_0$ *is uniquely determined. The reason is that each* *comatch of $tri_{ST}$ is completely determined by S- and T-component of the match of $tri_I$, because of embedding $R(tri_{ST}) \rightarrowtail L(tri_I)$. Furthermore, each match of $tri_{ST}$ is given by uniqueness of pushout complements along injective morphisms with respect to non-deleting rule $tri_{ST}$ and its comatch. Moreover, the source-target* TGT*-sequence implies $G_S \in VL_{S0}$ and $G_T \in VL_{T0}$.*

**Fact 2** (Model Integration is syntactically correct)**.** *Given model integration sequence $((G_S, G_T),\ G_0 \xRightarrow{tr_I^*} G_n,\ G)$ then $G_n = G \in VL$ with $proj_S(G) = G_S \in VL_S$ and $proj_T(G) = G_T \in VL_T$.*

*Proof.* $G_0 \xRightarrow{tr_I^*} G_n$ source-target consistent
$\Rightarrow\ \exists\ \emptyset \xRightarrow{tr_{ST}^*} G_0$ s.t. $\emptyset \xRightarrow{tr_{ST}^*} G_0 \xRightarrow{tr_I^*} G_n$ $S$-$T$-match consistent
$\overset{Thm2}{\Rightarrow}\ \emptyset \xRightarrow{tr^*} G_n$ , i.e. $G_n = G \in VL$ □

Finally we want to analyze which pairs $(G_S, G_T) \in VL_S \times VL_T$ can be integrated. Intuitively those which are related by the model transformation $MT : VL_S \Rrightarrow VL_T$ in Theorem 1. In fact, model integration sequences can be characterized by unique model transformation sequences.

**Theorem 3** (Characterization of Model Integration Sequences)**.** *Each model integration sequence $((G_S, G_T), G_0 \xRightarrow{tr_I^*} G_n, G)$ corresponds uniquely to a model transformation sequence $(G_S, G_0' \xRightarrow{tr_F^*} G_n, G_T)$, where $tr_I^*$ and $tr_F^*$ are based on the same rule sequence $tr^*$.*

*Proof.* $((G_S, G_T), G_0 \xRightarrow{tr_I^*} G_n, G)$ is model integration sequence
$\overset{def}{\Leftrightarrow}$ source-target consistent $G_0 \xRightarrow{tr_I^*} G_n$ with $proj_S(G_0) = proj_S(G_n) = G_S$, $proj_C(G_0) = \emptyset$, $proj_T(G_0) = proj_T(G_n) = G_T$ and $G_n = G$
$\overset{def}{\Leftrightarrow}\ \emptyset \xRightarrow{tr_{ST}^*} G_0 \xRightarrow{tr_I^*} G_n$ $S$-$T$-match consistent with $proj_S(G_n) = G_S$ and $proj_T(G_n) = G_T$
$\overset{Thm2}{\Leftrightarrow}\ \emptyset \xRightarrow{tr^*} G_n$ with $proj_S(G_n) = G_S$ and $proj_T(G_n) = G_T$
$\overset{Thm1}{\Leftrightarrow}\ \emptyset \xRightarrow{tr_S^*} G_0' \xRightarrow{tr_F^*} G_n$ match consistent with $proj_S(G_n) = G_S$ and $proj_T(G_n) = G_T$
$\overset{def}{\Leftrightarrow}\ G_0' \xRightarrow{tr_F^*} G_n$ source consistent with $proj_S(G_0') = proj_S(G_n) = G_S$ and $proj_T(G_n) = G_T$
$\overset{def}{\Leftrightarrow}\ (G_S, G_0' \xRightarrow{tr_F^*} G_n, G_T)$ is model transformation sequence. □

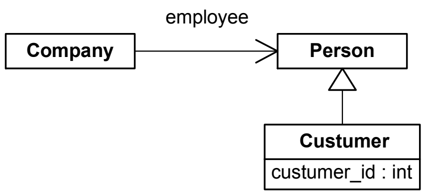



Figure 5: Source component of Fig. 6 in concrete syntax

Coming back to the example of a model transformation from class diagrams to database models the relevance and value of the given theorems can be described from the more practical view. Fig. 6 shows a triple graph, which defines a class diagram in its source component, database tables in its target component and the correspondences in between. Since this model is already fully integrated, it constitutes the resulting graph $G$ of example model integration sequence $((G_S, G_T), G_0 \xRightarrow{tr_I^*} G_n, G)$. The starting point is given by $G_S$ as restriction of $G$ to elements of the class diagram, indicated by pink, and $G_T$ containing the elements of the database part, indicated by yellow colour. Now, the blue nodes for correspondence as well as the morphisms between connection component to source and target component are created during the integration process. All elements are labeled with a number to specify matches and created objects for each transformation step. The sequence of applied rules is

$$G_0 \xRightarrow{Class2table} G_1 \xRightarrow{Class2table} G_2 \xRightarrow{Subclass2Table} G_3 \xRightarrow{PrimaryAttribute2Column} G_4 \xRightarrow{Association2ForeignKey} G_5 = G_n.$$

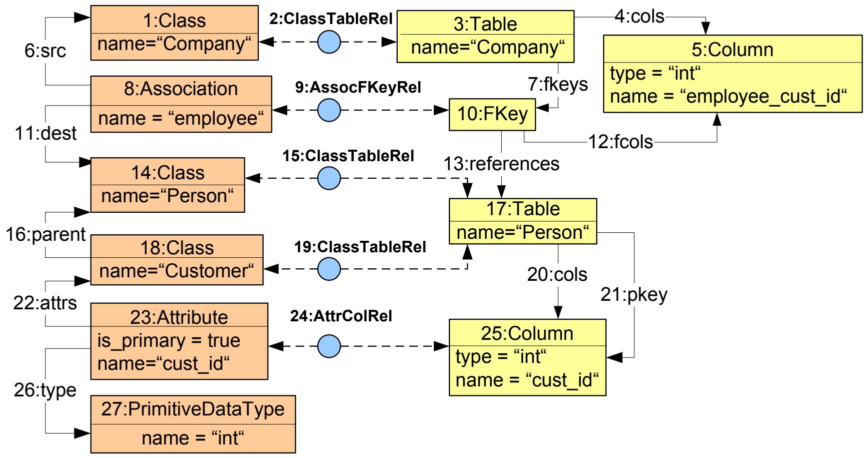



Figure 6: Example of model integration for model transformation $Class2Table$

| | Integration Sequence Elements | | Forward Sequence Elements | |
|---|---|---|---|---|
| Step | Matched | Created | Matched | Created |
| 1 | 1,3 | 2 | 1 | 2,3 |
| 2 | 14,17 | 15 | 14 | 15,17 |
| 3 | 14-18 | 19 | 14-18 | 19 |
| 4 | 17-20, 22,23, 25-27 | 24 | 17-19, 22,23, 26,27 | 20,21, 24,25 |
| 5 | 1-8, 10-15, 17,21,25 | 9 | 1-3,6,8, 11,14,15, 17,21,25 | 4,5,7,9,10,12,13 |

Table 1: Steps of example integration sequence

Now, Table 1 shows all matches of this sequence for both cases of Theorem 3 being the model integration sequence $G_0 \xRightarrow{tr_I^*} G_n$ and the forward transformation sequence $G'_0 \xRightarrow{tr_I^*} G_n$, where $G_0$ contains the elements of $G$ except correspondence parts and $G'_0$ is $G$ leaving out all elements of target and connection component. The column "Created" in the table lists the elements which are created at each transformation step. According to the numbers for the elements, the correspondence component is completely created during the model integration sequence and the elements of each match are created by the corresponding source-target rule application in $\emptyset \xRightarrow{tr_{ST}^*} G_0$. Therefore, $\emptyset \xRightarrow{tr_{ST}^*} G_0 \xRightarrow{tr_I^*} G_n$ is match consistent. Analogously $\emptyset \xRightarrow{tr_S^*} G'_0$ consists of the specified steps in Table 1, where comatches are given by the elements of the match in the forward transformation sequence implying $\emptyset \xRightarrow{tr_S^*} G'_0 \xRightarrow{tr_F^*} G_n$ being match consistent. Both integration and forward transformation sequence can be recaptured by analyzing the other, which corresponds to Theorem 3.

# 5 Related Work

Various approaches for model transformation in general are discussed in [MB03] and [OMG07] using BOTL and QVT respectively. For a taxonomy of model transformation based on graph transformation we refer to [MG06]. Triple Graph Grammars have been proposed by A. Schürr in [Sch94] for the specification of graph transformations. A detailed discussion of concepts, extensions, implementations and applications scenarios is given by E. Kindler and R. Wagner in [KW07]. The main application scenarios in [KW07] are model transformation, model integration and model synchronization. These concepts, however, are discussed only on an informal level using a slightly different concept of triple graphs compared with [Sch94].

In this paper we use the original definition of triple graphs, triple rules, and triple transformations of [Sch94] based on the double pushout approach (see [Roz97], [EEPT06]). In our paper [EEE+07] we have extended the approach of [Sch94] concerning the relationship between TGT-sequences based on triple rules $G_0 \stackrel{tr^*}{\Rightarrow} G_n$ and match consistent TGT-sequences $G_0 \stackrel{tr_S^*}{\Rightarrow} G_{n0} \stackrel{tr_F^*}{\Rightarrow} G_m$ based on source and forward rules leading to the canonical Decomposition and Composition Result 1 (Thm 1). This allows to characterize information preserving bidirectional model transformations in [EEE+07].

In this paper the main technical result is the Canonical Decomposition and Composition Result 2 (Thm 2) using source-target rules $tr_{ST}$ and integration rules $tr_I$ instead of $tr_S$ and $tr_F$. Both results are formally independent, but the same proof technique is used based on the Local Church–Rosser and Concurrency Theorem for graph transformations. The main result of [EEPT06] is based on these two decomposition and composition results. For a survey on tool integration with triple graph grammars we refer to [KS06].

# 6 Future Work and Conclusion

Model integration is an adequate technique in system design to work on specific models in different languages, in order to establish the correspondences between these models using rules which can be generated automatically. Once model transformation triple rules are defined for translations between the involved languages, integration rules can be derived automatically for maintaining consistency in the overall integrated modelling process.

Main contributions of this paper are suitable requirements for existence of model integration as well as composition and decomposition of source-target and integration transformations to and from triple transformations. Since model integration may be applied at any stage and several times during the modelling process, results of model integrations in previous stages can be used as the starting point for the next incremental step.

All concepts are explained using the well known case study for model transformation between class diagrams and relational data bases. While other model transformation approaches were applied to the same example for translation between source and target language, triple graph grammars additionally show their general power by automatic and constructive derivation of an integration formalism. Therefore, model integration in the presented way can scale up very easily, only bounded by the effort to build up general triple rules for parallel model evolution.

Usability extends when regarding partly connected models, which shall be synchronized as discussed on an informal level in [KW07]. On the basis of model integration rules model synchronization can be defined in future work as model integration using inverse source and target rules, standard source and target rules as well as integration rules in a mixed way, such that the resulting model is syntactically correct and completely integrated. Another interesting aspect for future work is the extension of triple graph rules and corresponding transformation and integration rules by negative application conditions (see [HHT96]), or by more general graph constraints (see [HP05]).

# References


[EEE+07] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, G. Taentzer. Information Preserving Bidirectional Model Transformations. In Dwyer and Lopes (eds.), *Fundamental Approaches to Software Engineering*. LNCS 4422, pp. 72–86. Springer, 2007.
`http://tfs.cs.tu-berlin.de/publikationen/Papers07/EEE+07.pdf`

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer Verlag, 2006.
`http://www.springer.com/3-540-31187-4`

[HHT96] A. Habel, R. Heckel, G. Taentzer. Graph Grammars with Negative Application Conditions. *Special issue of Fundamenta Informaticae* 26(3,4):287–313, 1996.

[HP05] A. Habel, K.-H. Pennemann. Nested Constraints and Application Conditions for High-Level Structures. In Kreowski et al. (eds.), *Formal Methods in Software and Systems Modeling*. Lecture Notes in Computer Science 3393, pp. 293–308. Springer, 2005.
`http://dx.doi.org/10.1007/b106390`

[KS06] A. König, 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.
`http://dx.doi.org/10.1016/j.entcs.2005.12.015`

[KW07] E. Kindler, R. Wagner. Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios. Technical report tr-ri-07-284, Software Engineering Group, Department of Computer Science, University of Paderborn, June 2007.
`http://www.uni-paderborn.de/cs/ag-schaefer/Veroeffentlichungen/Quellen/Papers/2007/tr-ri-07-284.pdf`

[MB03] F. Marschall, P. 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*. Pp. 25–36. 2003.
`http://citeseer.ist.psu.edu/marschall03model.html`

[MG06] T. Mens, P. V. Gorp. A Taxonomy of Model Transformation. In *Proc. International Workshop on Graph and Model Transformation (GraMoT'05), number 152 in Electronic Notes in Theoretical Computer Science, Tallinn, Estonia, Elsevier Science*. 2006.
`http://tfs.cs.tu-berlin.de/gramot/Gramot2005/FinalVersions/PDF/MensVanGorp.pdf`

[OMG07] OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Final Adopted Specification (07-07-2007). 2007.
`http://www.omg.org/docs/ptc/07-07-07.pdf`

[Roz97] G. Rozenberg (ed.). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[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.
`http://dx.doi.org/10.1007/3-540-59071-4_45`