Technische Universität Berlin

Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik

Formal Analysis of Functional Behaviour for Model Transformations Based on Triple Graph Grammars - Extended Version

Frank Hermann, Hartmut Ehrig,
Fernando Orejas, and Ulrike Golas

Bericht-Nr. 2010-08
ISSN 1436-9915

Formal Analysis of Functional Behaviour for Model Transformations Based on Triple Graph Grammars - Extended Version

Frank Hermann[1], Hartmut Ehrig[1], Fernando Orejas[2], and Ulrike Golas[1]

[1] {frank,ehrig,golas}@cs.tu-berlin.de, Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, Germany

[2] orejas(at)lsi.upc.edu, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain,

Abstract

Triple Graph Grammars (TGGs) are a well-established concept for the specification of model transformations. In previous work we have formalized and analyzed already crucial properties of model transformations like termination, correctness and completeness, but functional behaviour - especially local confluence - is missing up to now.

In order to close this gap we generate forward translation rules, which extend standard forward rules by translation attributes keeping track of the elements which have been translated already. In the first main result we show the equivalence of model transformations based on forward resp. forward translation rules. This way, an additional control structure for the forward transformation is not needed. This allows to apply critical pair analysis and corresponding tool support by the tool AGG. However, we do not need general local confluence, because confluence for source graphs not belonging to the source language is not relevant for the functional behaviour of a model transformation. For this reason we only have to analyze a weaker property, called translation confluence. This leads to our second main result, the functional behaviour of model transformations, which is applied to our running example, the model transformation from class diagrams to database models.

Keywords: Model Transformation, Triple Graph Grammars, Confluence, Functional Behaviour

1 Introduction

Model transformations based on triple graph grammars (TGGs) have been introduced by Schürr in [21]. TGGs are grammars that generate languages of graph triples, consisting of source and target graphs, together with a correspondence graph "between" them. Since 1994, several extensions of the original TGG definitions have been published [22, 15, 10] and various kinds of applications have been presented [23, 11, 14]. For source-to-target model transformations, so-called *forward* transformations, we derive rules which take the source graph as input and produce a corresponding target graph. Major properties expected to be fulfilled for model transformations are termination, correctness and completeness, which have been analyzed in [2, 4, 5, 1, 7].

In addition to these properties, functional behaviour of model transformations is an important property for several application domains. Functional behaviour means that for each graph in the source language the model transformation yields a unique graph (up to isomorphism) in the target language. It is well-known that termination and local confluence implies confluence and hence functional behaviour. Since termination has been analyzed already in [5] the main aim of this paper is to analyze local confluence in the view of functional behaviour for model transformations based on general TGGs. Our new technique is implicitly based on our constructions in [5], where the "on-the-fly" construction uses source and forward rules, which can be generated automatically from the triple rules. In this paper, we introduce forward translation rules which combine the source and forward rules using additional translation attributes for keeping track of the source elements that have been translated already. The first main result of this paper shows that there is a bijective correspondence between model transformations based on source consistent forward sequences and those based on forward translation sequences. Furthermore, we introduce an equivalent concept based on triple graphs with interfaces for handling the translation attributes by separating them from the source model in order to keep the source model unchanged.

In contrast to non-deleting triple rules, the corresponding forward translation rules are deleting and creating on the translation attributes. This means that some transformation steps can be parallel dependent. In this case we can apply the well-known critical pair analysis techniques to obtain local confluence. Since they are valid for all $\mathcal{M}$-adhesive systems (called weak adhesive HLR systems in [3]), they are also valid for typed attributed triple graph transformation systems. In fact, our model transformations based on forward translation rules can be considered as special case of the latter. However, we do not need general local confluence, because local confluence for transformations of all those source graphs, which do not belong to the source language, is not relevant for the functional bahaviour of a model transformation. In fact, we only analyze a weaker property, called translation confluence. This leads to our second main result, the functional behaviour of model transformations based on translation confluence. We have applied this technique for showing functional behaviour of our running example, the model transformation from class diagrams to database models, using our tool AGG [24] for critical pair analysis. Note that standard techniques are not applicable to show functional behaviour based on local

confluence.

This paper is organized as follows: In Sec. 2 we review the basic notions of TGGs and model transformations based on forward rules. In Sec. 3 we introduce forward translation rules and characterize in our first main result model transformations in the TGG approach by forward translation sequences. In Sec. 4 we show in our second main result how functional behaviour of model transformations can be analyzed by translation confluence and we apply the technique to our running example. While the presented concept for model transformations using translation attributes relies on a modification of the additional attributes within the source mode, Sec. 5 presents how the manipulation of the source model is avoided by externalizing the translation attributes using interface graphs. Related work and our conclusion - including a summary of our results and future work - is presented in Sections 6 and 7, respectively.

This technical report is an extended version of [13] and presents the full proofs and the concept of triple graphs with interfaces.

2 Review of Triple Graph Grammars

Triple graph grammars [21] are a well known approach for bidirectional model transformations. Models are defined as pairs of source and target graphs, which are connected via a correspondence graph together with its embeddings into these graphs. In [15], Königs and Schürr formalize the basic concepts of triple graph grammars in a set-theoretical way, which is generalized and extended by Ehrig et al. in [1] to typed, attributed graphs. In this section, we review main constructions and results of model transformations based on triple graph grammars [22, 5].

A triple graph $G = (G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T)$ consists of three graphs G_S, G_C, and G_T, called source, correspondence, and target graphs, together with two graph morphisms $s_G : G_C \to G_S$ and $t_G : G_C \to G_T$. A triple graph morphism $m = (m_S, m_C, m_T) : G \to H$ consists of three graph morphisms $m_S : G_S \to H_S$, $m_C : G_C \to H_C$ and $m_T : G_T \to H_T$ such that $m_S \circ s_G = s_H \circ m_C$ and $m_T \circ t_G = t_H \circ m_C$. A typed triple graph G is typed over a triple graph TG by a triple graph morphism $type_G : G \to TG$.

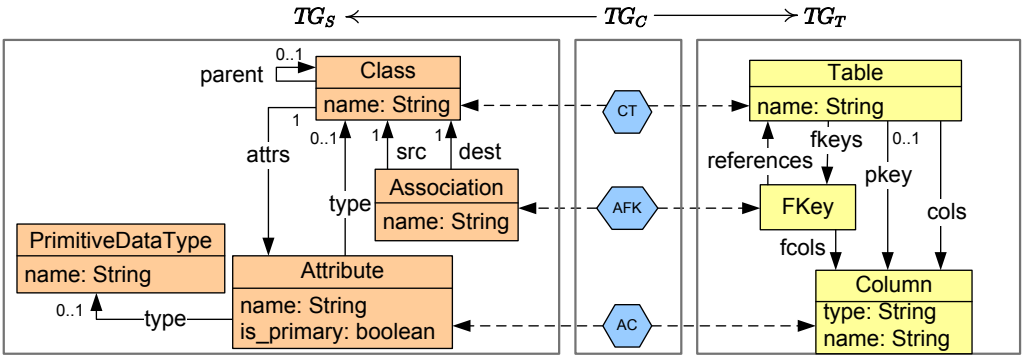


Figure 1: Triple type graph for *CD2RDBM*

Example 1 (Triple Type Graph). *Fig. 1 shows the type graph TG of the triple graph grammar TGG for our example model transformation CD2RDBM from class diagrams to database models. The source component TG_S defines the structure of class diagrams while in its target component the structure of relational database models is specified. Classes correspond to tables, attributes to columns, and associations to foreign keys. Throughout the example, originating from [1], elements are arranged left, center, and right according to the component types source, correspondence and target. Morphisms starting at a correspondence part are specified by dashed arrows. Furthermore, the triple rules of the grammar shown in Fig. 2 ensure several multiplicity constraints, which are denoted within the type graph. In addition, the source language CD only contains class diagrams where classes have unique primary attributes and subclasses have no primary attributes to avoid possible confusion.*

Note that the case study uses attributed triple graphs based on E-graphs as presented in [1] in the framework of $\mathcal{M}$-adhesive categories (called weak adhesive HLR in [3]).

Triple rules synchronously build up source and target graphs as well as their correspondence graphs, i.e. they are non-deleting. A triple rule tr is an injective triple graph morphism $tr = (tr_S, tr_C, tr_T) : L \to R$ and w.l.o.g. we assume tr to be an inclusion. Given a triple graph morphism $m : L \to G$, a triple graph transformation (TGT) step $G \overset{tr,m}{\Longrightarrow} H$ from G to a triple graph H is given by a pushout of triple graphs with comatch $n : R \to H$ and transformation inclusion $t : G \hookrightarrow H$. A grammar $TGG = (TG, S, TR)$ consists of a triple type graph TG, a triple start graph S and a set TR of triple rules.

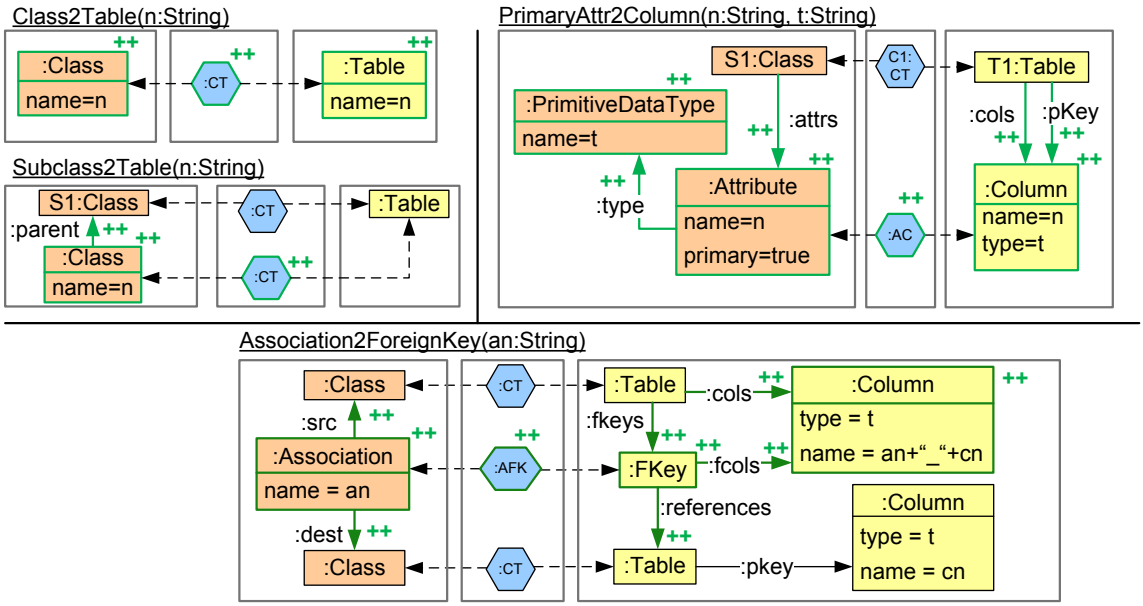


Figure 2: Rules for the model transformation *Class2Table*

Example 2 (Triple Rules)**.** *The triple rules in Fig. 2 are part of the rules of the grammar TGG for the model transformation CD2RDBM. They are presented in short notation, i.e. left and right hand sides of a rule are depicted in one triple graph. Elements, which are created by the rule, are labeled with green "++" and marked by green line colouring. The rule "*Class2Table*" synchronously creates a class in a class diagram with its corresponding table in the relational database. Accordingly, subclasses are connected to the tables of its super classes by rule "*Subclass2Table*". Attributes are created together with their corresponding columns in the database component. The depicted rule "*PrimaryAttr2Column*" concerns primary attributes with primitive data types for which an edge of type "pKey" is inserted that points to the column in the target component. This additional edge is not created for standard attributes, which are created by the rule "*Attr2Column*", which is not depicted. Finally, the rule "Association2ForeignKey" creates associations between two classes together with their corresponding foreign keys and an additional column that specifies the relation between the involved tables.*

$$\begin{array}{ccccc} (L_S & \longleftarrow & \varnothing & \longrightarrow & \varnothing) \\ {\scriptstyle tr_S}\downarrow & & \downarrow & & \downarrow \\ (R_S & \longleftarrow & \varnothing & \longrightarrow & \varnothing) \end{array} \qquad \begin{array}{ccccc} (R_S & \xleftarrow{tr_S \circ s_L} & L_C & \xrightarrow{t_L} & L_T) \\ {\scriptstyle id}\downarrow & & {\scriptstyle tr_C}\downarrow & & \downarrow{\scriptstyle tr_T} \\ (R_S & \xleftarrow{s_R} & R_C & \xrightarrow{t_R} & R_T) \end{array}$$

source rule tr_S forward rule tr_F

The operational rules for model transformations are automatically derived from the set of triple rules TR. From each triple rule tr we derive a forward rule tr_F for forward transformation sequences and a source rule tr_S for the construction resp. parsing of a model of the source language. By TR_S and TR_F we denote the sets of all source and forward rules derived from TR. Analogously, we derive a target rule tr_T and a backward rule tr_B for the construction and transformation of a model of the target language leading to the sets TR_T and TR_B.

A set of triple rules TR and the start graph $\varnothing$ generate a visual language VL of integrated models, i.e. models with elements in the source, target and correspondence component. The source language VL_S and target language VL_T are derived by projection to the triple components, i.e. $VL_S = proj_S(VL)$ and $VL_T = proj_T(VL)$. The set VL_{S0} of models that can be generated resp. parsed by the set of all source rules TR_S is possibly larger than VL_S and we have $VL_S \subseteq VL_{S0} = \{G_S \,|\, \varnothing \Rightarrow^* (G_S \leftarrow \varnothing \rightarrow \varnothing)$ *via* $TR_S\}$. Analogously, we have $VL_T \subseteq VL_{T0} = \{G_T \,|\, \varnothing \Rightarrow^* (\varnothing \leftarrow \varnothing \rightarrow G_T)$ *via* $TR_T\}$.

As introduced in [1, 5] the derived operational rules provide the basis to define model transformations based on source consistent forward transformations $G_0 \Rightarrow^* G_n$ via $(tr_{1,F}, \ldots, tr_{n,F})$, short $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$. Source consistency of $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ means that there is a source sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0$ such that the sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is match consistent, i.e. the S-component of each match $m_{i,F}$ of $tr_{i,F} (i = 1 \ldots n)$ is uniquely determined by the comatch $n_{i,S}$ of $tr_{i,S}$, where $tr_{i,S}$ and $tr_{i,F}$ are source and forward rules of the same triple rules tr_i. Altogether the forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is controlled by the corresponding source sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0$, which is unique in the case of match consistency.

Definition 1 (Model Transformation based on Forward Rules)**.** *A* model transformation sequence $(G_S,\ G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ *consists of a source graph* G_S*, a target graph* G_T*, and a source consistent forward* TGT*-sequence* $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ *with* $G_S = G_{0,S}$ *and* $G_T = G_{n,T}$*.*
A model transformation $MT : VL_{S0} \Rrightarrow VL_{T0}$ *is defined by all model transformation sequences* $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ *with* $G_S \in VL_{S0}$ *and* $G_T \in VL_{T0}$*. All the corresponding pairs* (G_S, G_T) *define the* model transformation relation $MTR_F \subseteq VL_{S0} \times VL_{T0}$*.*

In [1, 5] we have proved that source consistency ensures completeness and correctness of model transformations based on forward rules with respect to the language VL of integrated models. Moreover, source consistency is the basis for the on-the-fly construction defined in [5].

3 Model Transformations based on Forward Translation Rules

Model transformations as defined in the previous section are based on source consistent forward sequences. In order to analyze functional behaviour, we present in this section a characterizion by model transformations based on forward translation rules, which integrate the control condition source consistency using additional attributes (see Thm. 1). For each node, edge and attribute of a graph a new attribute is created and labeled with the prefix "*tr*". If this prefix is used already for an existing attribute, then a unique extended prefix is chosen.

The extension of forward rules to forward translation rules is based on new attributes that control the translation process according to the source consistency condition. For each node, edge and attribute of a graph a new attribute is created and labeled with the prefix "*tr*". Given an attributed graph $AG = (G, D)$ and a family of subsets $M \subseteq G$ for nodes and edges, we call AG' a graph with translation attributes over AG if it extends AG with one boolean-valued attribute tr_x for each element x (node or edge) in M and one boolean-valued attribute tr_x_a for each attribute associated to such an element x in M. The family M together with all these additional translation attributes is denoted by Att_M. Note that we use the attribution concept of E-Graphs as presented in [3], where attributes are possible for nodes and edges.

Definition 2 (Family with Translation Attributes)**.** *Given an attributed graph* $AG = (G, D)$ *we denote by* $|G| = (V_G^G, V_G^D, E_G^G, E_G^{NA}, E_G^{EA})$ *the underlying family of sets containing all nodes and edges. Let* $M \subseteq |G|$*, then a* family with translation attributes *for* (G, M) *extends* M *by additional translation attributes and is given by* $Att_M = (V_M^G, V_M^D, E_M^G, E^{NA}, E^{EA})$ *with:*

- $E^{NA} = E_M^{NA} \cup \{tr\_x \mid x \in V_M^G\} \cup \{tr\_x\_a \mid a \in E_M^{NA}, src_G^{NA}(a) = x \in V_G^G\}$,
- $E^{EA} = E_M^{EA} \cup \{tr\_x \mid x \in E_M^G\} \cup \{tr\_x\_a \mid a \in E_M^{EA}, src_G^{EA}(a) = x \in E_G^G\}$.

Definition 3 (Graph with Translation Attributes). *Given an attributed graph $AG = (G, D)$ and a family of subsets $M \subseteq |G|$ with $\{\mathbf{T}, \mathbf{F}\} \subseteq V_M^D$ and let Att_M be a family with translation attributes for (G, M). Then, $AG' = (G', D)$ is a graph with translation attributes over AG, where $|G'|$ is the gluing of $|G|$ and Att_M over M, i.e. the sets of nodes and edges are given by componentwise pushouts and the source and target functions are defined as follows:*

- $src^G_{G'} = src^G_G$, $trg^G_{G'} = trg^G_G$,
- $src^X_{G'}(z) = \begin{cases} src^X_G(z) & z \in E^X_G \\ x & z = tr\_x \text{ or } z = tr\_x\_a \end{cases}$ *for* $X \in \{NA, EA\}$,
- $trg^X_{G'}(z) = \begin{cases} trg^X_G(z) & z \in E^X_G \\ \mathbf{T} \text{ or } \mathbf{F} & z = tr\_x \text{ or } z = tr\_x\_a \end{cases}$ *for* $X \in \{NA, EA\}$.

$$\begin{array}{ccc} M & \hookrightarrow & Att_M \\ \downarrow & (PO) & \downarrow \\ |G| & \longrightarrow & |G'| \end{array}$$

Att^v_M, where $v = \mathbf{T}$ or $v = \mathbf{F}$, denotes a family with translation attributes where all attributes are set to v. Moreover, we denote by $AG \oplus Att_M$ that AG is extended by the translation attributes in Att_M i.e. $AG \oplus Att_M = (G', D) = AG'$. Analogously, we use the notion $AG \oplus Att^v_M$ for translation attributes with value v and we define $Att^v(AG) := AG \oplus Att^v_{|G|}$.

The extension of forward rules to forward translation rules ensures that the effective elements of the rule may only be matched to those elements that have not been translated so far. A first intuitive approach would be to use NACs on the correspondence component of the forward rule in order to check that the effective elements are unrelated. However, this approach is too restrictive, because e.g. edges and attributes in the source graph cannot be checked separately, but only via their attached nodes. Moreover, the analysis of functional behaviour of model transformations with NACs is general more complex compared to using boolean valued translation attributes instead. Thus, the new concept of forward translation rules extends the construction of forward rules by additional translation attributes, which keep track of the elements that have been translated at any point of the transformation process. This way, each element in the source graph cannot be translated twice, which is one of the main aspects of source consistency. For that reason, all translation attributes of the source model of a model transformation are set to false and in the terminal graph we expect that all the translation attributes are set to true. Moreover, also for that reason, the translation rules set to true all the elements of the source rule that would be generated by the corresponding source rule. This requires that the rules are deleting on the translation attributes and we extend a transformation step from a single (total) pushout to the classical double pushout (DPO) approach [3]. Thus, we can ensure source consistency by merely

using attributes in order to completely translate a model. Therefore, we call these rules forward translation rules, while pure forward rules need to be controlled by the source consistency condition. Note that the extension of a forward rule to a forward translation rule is unique.

Definition 4 (Forward Translation Rule)**.** *Given a triple rule* $tr = (L \to R)$*, the* forward translation rule *of* tr *is given by* $tr_{FT} = (L_{FT} \xleftarrow{l_{FT}} K_{FT} \xrightarrow{r_{FT}} R_{FT})$ *defined as follows using the forward rule* $(L_F \xrightarrow{tr_F} R_F)$ *and the source rule* $(L_S \xrightarrow{tr_S} R_S)$ *of* tr*, where we assume w.l.o.g. that* tr *is an inclusion:*

- $K_{FT} = L_F \oplus Att^{\mathbf{T}}_{L_S}$,
- $L_{FT} = L_F \oplus Att^{\mathbf{T}}_{L_S} \oplus Att^{\mathbf{F}}_{R_S \setminus L_S}$,
- $R_{FT} = R_F \oplus Att^{\mathbf{T}}_{L_S} \oplus Att^{\mathbf{T}}_{R_S \setminus L_S} = R_F \oplus Att^{\mathbf{T}}_{R_S}$,
- l_{FT} *and* r_{FT} *are the induced inclusions.*

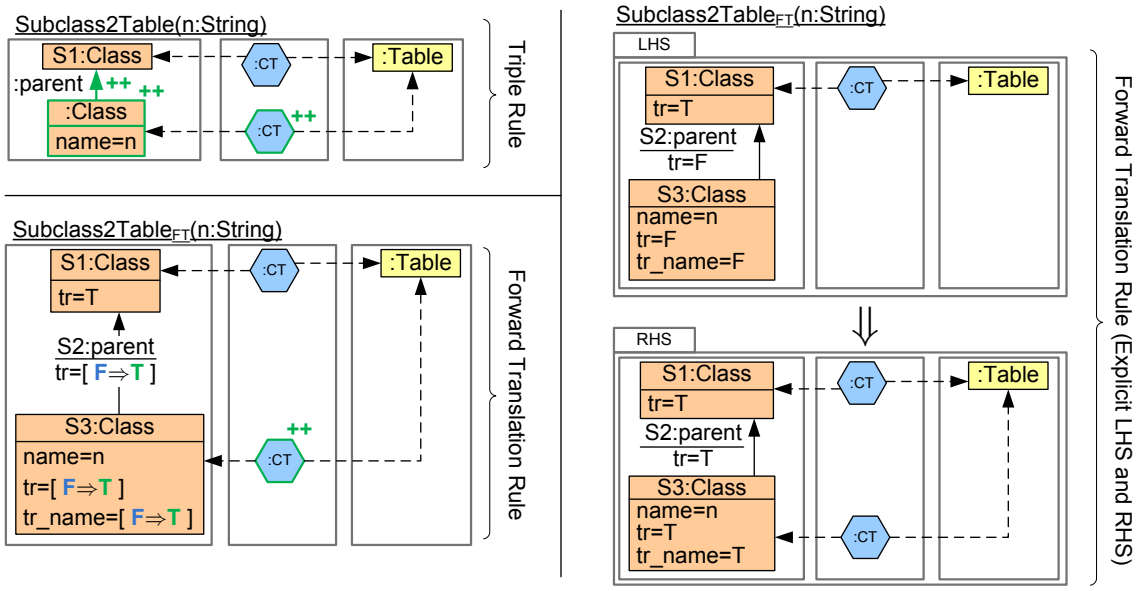


Figure 3: Forward translation rule $Subclass2Table_{FT}(n : String)$

Example 3 (Derived Forward Translation Rules)**.** *Figure 3 shows the derived forward translation rule "*$Subclass2Table_{FT}$*" for the triple rule"Subclass2Table" in Fig. 2. Note that we abbreviate "*tr_x*" for an item (node or edge)* x *by "tr" and "*tr_x_a*" by "*$tr\_type(a)$*" in the figures to increase readability. The compact notation of forward translation rules specifies the modification of translation attributes by "*$[\mathbf{F} \Rightarrow \mathbf{T}]$*", meaning that the attribute is matched with the value "***F***" and set to "***T***" during the transformation step. The detailed complete notation of a forward translation rule is shown on the right of Fig. 3 for "*$Subclass2Table_{FT}$*".*

From the application point of view a model transformation should be injective on the structural part, i.e. the transformation rules are applied along matches that do not identify structural elements. But it would be too restrictive to require injectivity of the matches also on the data and variable nodes in the abstract syntax graphs of models, because the matching should allow to match two different variables in the left hand side of a rule to the same data value in the host graph of a transformation step. Thus, this notion of almost injective matches applies to all model transformations based on abstract syntax graphs with attribution. For this reason we introduce the notion of almost injective matches, which requires that matches are injective except for the data value nodes. This way, attribute values can still be specified as terms within a rule and matched non-injectively to the same value.

Definition 5 (Almost Injective Match and Completeness)**.** *An attributed triple graph morphism $m : L \to G$ is called* almost injective*, if it is non-injective at most for the set of variables and data values in L_{FT}. A forward translation sequence $G_0 \overset{tr^*_{FT}}{\Longrightarrow} G_n$ with almost injective matches is called* complete *if G_n is completely translated, i.e. all translation attributes of G_n are set to true ("***T***").*

In order to prove Fact 1, which is needed for the equivalence in Thm. 1 of model transformations based on forward rules and those based on forward translation rules, we first prove Lemma 1, which states the equivalence for a single step using the on-the-fly construction of [5]. For this purpose, we recall the main definition of partial source consistency, partial match consistency and forward consistent matches.

Definition 6 (Partial Match and Source Consistency)**.** *Let TR be a set of triple rules and let TR_F be the derived set of forward rules. A sequence*

$$\varnothing = G_{00} \overset{tr^*_S}{\Longrightarrow} G_{n0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr^*_F}{\Longrightarrow} G_n$$

defined by pushout diagrams (1) *and* (3) *for $i = 1 \dots n$ with $G^C_0 = \varnothing$, $G^T_0 = \varnothing$ and inclusion $g_n : G_{n0} \hookrightarrow G_0$ is called* partially match consistent*, if diagram* (2) *commutes for all i, which means that the source component of the forward match $m_{i,F}$ is determined by the comatch $n_{i,S}$ of the corresponding step of the source sequence with $g_i = g_n \circ t_{n,S} \dots t_{i-1,S}$.*

$$\begin{array}{ccccccc}
L_{i,S} & \overset{tr_{i,S}}{\hookrightarrow} & R_{i,S} & \hookrightarrow & L_{i,F} & \overset{tr_{i,F}}{\hookrightarrow} & R_{i,F} \\
{\scriptstyle m_{i,S}}\downarrow & (1) & \downarrow{\scriptstyle n_{i,S}} & (2) & {\scriptstyle m_{i,F}}\downarrow & (3) & \downarrow{\scriptstyle n_{i,F}} \\
G_{i-1,0} & \underset{t_{i,S}}{\hookrightarrow} & G_{i,0} & \underset{g_i}{\hookrightarrow} & G_0 & \hookrightarrow & G_{i-1} \quad \underset{t_{i,F}}{\hookrightarrow} \quad G_i
\end{array}$$

A forward sequence $G_0 \overset{tr^*_F}{\Longrightarrow} G_n$ is partially source consistent, if there is a source sequence $\varnothing = G_{00} \overset{tr^*_S}{\Longrightarrow} G_{n0}$ with inclusion $G_{n0} \overset{g_n}{\hookrightarrow} G_0$ such that $G_{00} \overset{tr^*_S}{\Longrightarrow} G_{n0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr^*_F}{\Longrightarrow} G_n$ is partially match consistent.

Definition 7 (Forward Consistent Match)**.** *Given a partially match consistent sequence $\varnothing = G_{00} \overset{tr^*_S}{\Longrightarrow} G_{n-1,0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr^*_F}{\Longrightarrow} G_{n-1}$ then a match $m_{n,F}$:*

$L_{n,F} \to G_{n-1}$ *for* $tr_{n,F} : L_{n,F} \to R_{n,F}$ *is called* forward consistent *if there is a source match* $m_{n,S}$ *such that diagram* (1) *is a pullback.*

$$\begin{array}{ccccc} L_{n,S} & \hookrightarrow & R_{n,S} & \hookrightarrow & L_{n,F} \\ {\scriptstyle m_{n,S}}\downarrow & & (1) & & \downarrow{\scriptstyle m_{n,F}} \\ G_{n-1,0} & \xrightarrow[g_{n-1}]{} & G_0 & \hookrightarrow & G_{n-1} \end{array}$$

We first proof the equivalence of forward translation sequences and source consistent forward transformations for single steps as stated by Lemma 1.

Lemma 1 (Forward translation step)**.** *Let* TR *be a set of triple rules with* $tr_i \in TR$ *and let* TR_F *be the derived set of forward rules. Given a partially match consistent forward sequence* $\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{i-1,0} \xrightarrow{g_{i-1}} G_0 \overset{tr_F^*}{\Longrightarrow} G_{i-1}$ *and a corresponding forward translation sequence* $G'_0 \overset{tr_{FT}^*}{\Longrightarrow} G'_{i-1}$*, both with almost injective matches, such that* $G'_{i-1} = G_{i-1} \oplus Att^{\mathbf{F}}_{G_0 \setminus G_{i-1,0}} \oplus Att^{\mathbf{T}}_{G_{i-1,0}}$*. Then the following are equivalent:*

1. $\exists$ TGT*-step* $G_{i-1} \overset{tr_{i,F}, m_{i,F}}{\Longrightarrow} G_i$ *with forward consistent match* $m_{i,F}$
2. $\exists$ *translation* TGT*-step* $G'_{i-1} \overset{tr_{i,FT}, m_{i,FT}}{\Longrightarrow} G'_i$

and we have $G'_i = G_i \oplus Att^{\mathbf{F}}_{G_0 \setminus G_{i,0}} \oplus Att^{\mathbf{T}}_{G_{i,0}}$.

Proof. For simpler notation we assume w.l.o.g. that rule morphisms are inclusions and matches are inclusions except for the data value component.

Constructions:
1. TGT-step $G_{i-1} \overset{tr_F}{\Longrightarrow} G_i$ with forward consistent match is given by

$$\begin{array}{ccccccccc} L_{i,S} & \xrightarrow{tr_{i,S}} & R_{i,S} & & \longrightarrow & & L_{i,F} & \xrightarrow{tr_{i,F}} & R_{i,F} \\ {\scriptstyle m_{i,S}}\downarrow & (1) & \downarrow{\scriptstyle n_{i,S}} & (2) & & {\scriptstyle m_{i,F}}\downarrow & (3) & \downarrow{\scriptstyle n_{i,F}} & \\ G_{i-1,0} & \xrightarrow[t_{i,S}]{} & G_{i,0} & \xrightarrow[g_i]{} & G_0 & \hookrightarrow & G_{i-1} & \xrightarrow[t_{i,F}]{} & G_i \end{array}$$

where (1) and (3) are pushouts and pullbacks, (2) commutes, and since $m_{i,F}$ is forward consistent we have by Def. 7 that (2) and therefore also $(1+2)$ is a pullback.

$(1+2)$ is a pullback
$\Leftrightarrow m_{i,F}(L_{i,F}) \cap G_{i-1,0} = m_{i,F}(L_{i,S})$
$\Rightarrow m_{i,F}(L_{i,F} \setminus L_{i,S}) \cap G_{i-1,0} = \varnothing.$

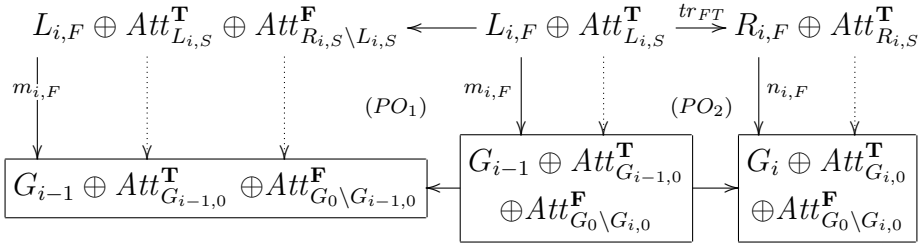
2. Translation TGT-step $G'_{i-1} \overset{tr_{i,FT}, m_{i,FT}}{\Longrightarrow} G'_i$is given by (PO_1), (PO_2)

$$\begin{array}{ccccc} L_{i,FT} & \longleftarrow & K_{i,FT} & \longrightarrow & R_{i,FT} \\ \downarrow & (PO_1) & \downarrow & (PO_2) & \downarrow \\ G'_{i-1} & \longleftarrow & D'_{i-1} & \longrightarrow & G'_i \end{array}$$

$L_{i,FT} = L_{i,F} \oplus Att^{\mathbf{T}}_{L_{i,S}} \oplus Att^{\mathbf{F}}_{R_{i,S} \setminus L_{i,S}}$
$K_{i,FT} = L_{i,F} \oplus Att^{\mathbf{T}}_{L_{i,S}}$

$$L_{i,FT} = R_{i,F} \oplus Att^{\mathbf{T}}_{L_{i,S}} \oplus Att^{\mathbf{T}}_{R_{i,S} \setminus L_{i,S}} = R_{i,F} \oplus Att^{\mathbf{T}}_{R_{i,S}}$$

Direction 1. $\Rightarrow$ 2. : We construct (PO_1), (PO_2) as follows from diagrams $(1) - (3)$:



The match $m_{i,FT}$ is constructed as follows:

$$m_{i,FT}(x) = \begin{cases} m_{i,F}(x), & x \in L_{i,F} \\ tr\_m_{i,F}(y), & x = tr\_y, src_{L_{FT}}(x) = y \\ tr\_m_{i,F}(y)\_a, & x = tr\_y\_a, src_{L_{FT}}(x) = y \end{cases}$$

The match $m_{i,F}$ is injective except for the data value nodes. For this reason, the match $m_{i,FT}$ is an almost injective match, i.e. possibly non-injective on the data values.

Pushouts $(PO_1), (PO_2)$ are equivalent to pushouts $(0), (3)$ below without translation attributes. Thus, the additional translation attributes are not involved in these pushouts.

$$\begin{array}{ccccc} L_{i,F} & \xleftarrow{id} & L_{i,F} & \longrightarrow & R_{i,F} \\ \downarrow & (0) & \downarrow & (3) & \downarrow \\ G_{i-1} & \xleftarrow{id} & G_{i-1} & \longrightarrow & G_i \end{array}$$

We now consider the translation attributes.

Let $E_{i,0} = (G_0 \setminus G_{i-1,0}) \setminus (n_{i,S}(R_{i,S} \setminus L_{i,S}))$ constructed componentwise on the sets of nodes and edges. This implies that $E_{i,0}$ is a family of sets and not necessarily a graph, because some edges could be dangling. However, we only need to show the pushout properties for these sets, because the boundary nodes and context is handled properly in pushouts $(0), (3)$ before and the translation attribute edges for the items in $E_{i,0}$ are derived uniquely according to Def. 3. Thus, we have the following pushouts for the translation attributes:

$$\begin{array}{ccc} L_{i,S} & \longleftarrow & L_{i,S} \\ \downarrow & PO_1^{\mathbf{T}} & \downarrow \\ G_{i-1,0} & \longleftarrow & G_{i-1,0} \end{array} \qquad \begin{array}{ccc} (R_{i,S} \setminus L_{i,S}) & \longleftarrow & \varnothing \\ \downarrow & PO_1^{\mathbf{F}} & \downarrow \\ (G_0 \setminus G_{i-1,0}) & \longleftarrow & E_{i,0} \end{array} \qquad \begin{array}{ccc} L_{i,S} & \longrightarrow & R_{i,S} \\ \downarrow & PO_2^{\mathbf{T}} & \downarrow \\ G_{i-1,0} & \longrightarrow & G_{i,0} \end{array} \qquad \begin{array}{ccc} \emptyset & \longrightarrow & \emptyset \\ \downarrow & PO_2^{\mathbf{F}} & \downarrow \\ E_{i,0} & \longrightarrow & E_{i,0} \end{array}$$

Pushout $(PO_1^{\mathbf{T}})$ is a trivial pushout, $(PO_1^{\mathbf{F}})$ is pushout by the definition of $E_{i,0}$, $(PO_2^{\mathbf{T}})$ is a pushout by (1) and $(PO_2^{\mathbf{F}})$ is a trivial pushout. Using pushout (1) for the source step we have $G_{i,0} = G_{i-1,0} \cup (n_{i,S}(R_{i,S} \setminus L_{i,S}))$ and thus, $E_{i,0} = (G_0 \setminus G_{i-1,0}) \setminus (n_{i,S}(R_{i,S} \setminus L_{i,S})) = G_0 \setminus (G_{i-1,0} \cup n_{i,S}(R_{i,S} \setminus L_{i,S})) = (G_0 \setminus G_{i,0})$. This implies $G'_i = G_i \oplus Att^{\mathbf{T}}_{G_i,0} \oplus Att^{\mathbf{F}}_{G_0 \setminus G_{i,0}}$.

Direction 2. $\Rightarrow$ 1. :

We construct diagrams $(1) - (3)$ from pushouts $(PO_1), (PO_2)$. The pushouts (PO_1)

and (PO_2) without translation attributes are equivalent to the pushouts $(0), (3)$ and $(PO_1^{\mathbf{T}}), (PO_1^{\mathbf{F}}), (PO_2^{\mathbf{T}}), (PO_2^{\mathbf{F}})$ for families of sets. They do not overlap, because the have different types according to the construction of the type graph with attributes by Def. 3. The match is a forward translation match and thus, it is injective on all components except the data value nodes. It remains to construct diagrams (1) and (2) for graphs with (1) as a pushout. Since the C- and T-components of (1) and (2) are trivial it remains to construct the corresponding S-components, denoted here by $L_{i,S}^S$ for $L_{i,S}$ etc. The morphisms $L_{i,S}^S \xhookrightarrow{tr_{i,S}^S} R_{i,S}^S \xrightarrow{id} L_{i,F}^S \xrightarrow{m_{i,F}^S} G_{i-1}^S$ are given already as graph morphisms. By $(PO_2^{\mathbf{T}})$ we have a pushout in family of sets and $G_{i-1,0}^S \subseteq G_0^S = G_{i-1}^S$ by assumption leads to a unique $G_{i-1,0}^S \hookrightarrow G_{i-1}^S = G_0^S$, such that (4) and (5) below commute for families of sets, using that $(PO_2^{\mathbf{T}})_S$ is a pullback and hence, also $(PO_2^{\mathbf{T}})_S + (4)$ is a pullback for families of sets.

$$\begin{array}{ccccc} L_{i,S}^S & \hookrightarrow & R_{i,S}^S & \xrightarrow{id} & L_{i,F}^S \\ \downarrow & (PO_2^{\mathbf{T}})_S & \downarrow & (4) & \downarrow \\ G_{i-1,0}^S & \longrightarrow & G_{i,0}^S & \dashrightarrow & G_{i-1}^S = G_0^S \end{array} \qquad (5)$$

Since $L_{i,S}^S \xhookrightarrow{tr_{i,S}^S} R_{i,S}^S \xrightarrow{id} L_{i,F}^S \xrightarrow{m_{i,F}^S} G_{i-1}^S = G_0^S$ and $G_{i-1,0}^S \hookrightarrow G_0^S$ are graph morphisms by assumption and $G_{i-1,0}^S \hookrightarrow G_0^S$ is injective, we also have that $L_{i,S}^S \to G_{i-1}^S$ is a graph morphism such that $(PO_2^{\mathbf{T}})_S$ becomes a pushout in Graphs with unique source and target maps for $G_{i,0}^S$. Finally, this implies that $G_{i,0}^S \to G_{i-1}^S = G_0^S$ is an injective graph morphism and w.l.o.g. an inclusion. Hence, we obtain the diagrams (1) and (2) for triple graphs from $(PO_2^{\mathbf{T}})_S$ and (4) for graphs, where (4) is a pushout and a pullback and $(1) + (2)$ is a pullback by pullback (1) and injective $G_{i,0} \hookrightarrow G_0 \hookrightarrow G_{i-1}$.

Using pushout (1) for families of sets given by $(PO_2^{\mathbf{T}})_S$ we have $G_{i,0} = G_{i-1,0} \cup n_{i,S}(R_{i,S} \setminus L_{i,S})$ and thus, $E_{i,0} = (G_0 \setminus G_{i,0})$ implying $G_i' = G_i \oplus Att_{G_{i,0}}^{\mathbf{T}} \oplus Att_{G_0 \setminus G_{i,0}}^{\mathbf{F}}$.

□

Now we are able to show the equivalence of complete forward translation sequences with source consistent forward sequences as stated by Fact 1 below.

Fact 1 (Complete Forward Translation Sequences)**.** *Given a triple graph grammar* $TGG = (TG, \varnothing, TR)$ *and a triple graph* $G_0 = (G_S \leftarrow \varnothing \rightarrow \varnothing)$ *typed over* TG*. Let* $G_0' = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$*. Then, the following are equivalent for almost injective matches:*

1. $\exists$ *a source consistent* TGT*-sequence* $G_0 \overset{tr_F^*}{\Longrightarrow} G$ *via forward rules and* $G = (G_S \leftarrow G_C \rightarrow G_T)$*.*

2. $\exists$ *a complete* TGT*-sequence* $G_0' \overset{tr_{FT}^*}{\Longrightarrow} G'$ *via forward translation rules and* $G' = (Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T)$*.*

Proof.
1. $\Leftrightarrow G_0 \overset{tr_{1,F}, m_{1,F}}{\Longrightarrow} G_1 \overset{tr_{2,F}, m_{2,F}}{\Longrightarrow} G_2 \ldots \overset{tr_{n,F}, m_{n,F}}{\Longrightarrow} G_n = G$, where each match is forward consistent according to Thm. 1 in [5].

2. $\Leftrightarrow G'_0 \overset{tr_{1,FT},m_{1,FT}}{\Longrightarrow} G'_1 \overset{tr_{2,FT},m_{2,FT}}{\Longrightarrow} G'_2 \ldots \overset{tr_{n,FT},m_{n,FT}}{\Longrightarrow} G'_n = G'$.
It remains to show that $G'_{0,S} = Att^{\mathbf{F}}(G_S)$ and $G'_S = Att^{\mathbf{T}}(G_S)$.
We apply Lemma 1 for $i = 0$ with $G_{0,0} = \varnothing$ up to $i = n$ with $G_{n,0} = G_0$ and using $G_{0,S} = G_S$ we derive:
$G'_{0,S} = G_{0,S} \oplus Att^{\mathbf{T}}_{G_{0,0}} \oplus Att^{\mathbf{F}}_{G_{0,S} \setminus G_{0,0,S}} = G_{0,S} \oplus Att^{\mathbf{F}}_{G_{0,S}} = G_S \oplus Att^{\mathbf{F}}_{G_S} = Att^{\mathbf{F}}(G_S)$.
$G'_S = G'_{n,S} = G_{n,S} \oplus Att^{\mathbf{T}}_{G_{n,0,S}} \oplus Att^{\mathbf{F}}_{G_{0,S} \setminus G_{n,0,S}} = G_{n,S} \oplus Att^{\mathbf{T}}_{G_{n,0,S}} = G_S \oplus Att^{\mathbf{T}}_{G_S} = Att^{\mathbf{T}}(G_S)$.
□

Now, we define model transformations based on forward translation rules in the same way as for forward rules in Def. 1, where source consistency of the forward sequence is replaced by completeness of the forward translation sequence. Note that we can separate the translation attributes from the source model as shown in Sec. 5 in order to keep the source model unchanged.

Definition 8 (Model Transformation Based on Forward Translation Rules)**.** *A* model transformation sequence $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G_T)$ *based on forward translation rules consists of a source graph G_S, a target graph G_T, and a complete* TGT*-sequence $G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n$ with almost injective matches, $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ and $G'_n = (Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T)$. A* model transformation $MT : VL_{S0} \Rrightarrow VL_{T0}$ *based on forward translation rules is defined by all model transformation sequences $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G_T)$ based on forward translation rules with $G_S \in VL_{S0}$ and $G_T \in VL_{T0}$. All these pairs (G_S, G_T) define the* model transformation relation $MTR_{FT} \subseteq VL_{S0} \times VL_{T0}$. *The model transformation is* terminating *if there are no infinite* TGT*-sequences via forward translation rules and almost injective matches starting with $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ for some source graph G_S.*

The main result of this section in Thm. 1 below states that model transformations based on forward translation rules are equivalent to those based on forward rules.

Theorem 1 (Equivalence of Model Transformation Concepts)**.** *Given a triple graph grammar, then the model transformation $MT_F : VL_{S0} \Rrightarrow VL_{T0}$ based on forward rules and the model transformation $MT_{FT} : VL_{S0} \Rrightarrow VL_{T0}$ based on forward translation rules, both with almost injective matches, define the same model transformation relation $MTR_F = MTR_{FT} \subseteq VL_{S0} \times VL_{T0}$.*

Proof. The theorem follows directly from Def. 1, Def. 8 and Fact 1. □

Remark 1. *It can be shown that the model transformation relation MTR defined by the triple rules TR coincides with the relations MTR_F and MTR_{FT} of the model transformations based on forward and forward translation rules TR_F and TR_{FT}, respectively.*

The equivalence of model transformations in Thm. 1 above directly implies Thm. 2 beneath, because we already have shown the results for model transformations based on forward rules in [5]. Note that the provided condition for termination is sufficient and in many cases also necessary. The condition is not necessary only for the case that there are some source identic triple rules, but none of them is applicable to any integrated model in the triple language *VL*.

Theorem 2 (Termination, Correctness and Completeness)**.** *Each model transformation* $MT : VL_{S0} \Rrightarrow VL_{T0}$ *based on forward translation rules is*

- terminating*, if each forward translation rule changes at least one translation attribute,*
- correct*, i.e. for each model transformation sequence* $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G_T)$ *there is* $G \in VL$ *with* $G = (G_S \leftarrow G_C \rightarrow G_T)$*, and it is*
- complete*, i.e. for each* $G_S \in VL_S$ *there is* $G = (G_S \leftarrow G_C \rightarrow G_T) \in VL$ *with a model transformation sequence* $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G_T)$*.*

Proof. By Def. 4 we have that a rule changes the translation attributes iff the source rule of the original triple rule is creating, which is a sufficient criteria for termination by Thm. 3 in [5]. The correctness and completeness are based on Thm. 1 above and the proof of Thm. 3 in [4]. Note that Thm. 3 in [4] states a weaker result of correctness and completeness for source consistent forward transformations.

However, the proof is based on the composition and decomposition of triple graph transformation sequences shown by Thm. 1 in [1]. In more detail, each triple transformation sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_n$ can be decomposed into a match consistent triple transformation sequence $\varnothing \overset{tr^*_S}{\Longrightarrow} G_{n,0} \overset{tr^*_F}{\Longrightarrow} G_n$, which means that the forward sequence $G_{n,0} \overset{tr^*_F}{\Longrightarrow} G_n$ is source consistent. Vice versa, given a source consistent forward sequence $G_{n,0} \overset{tr^*_F}{\Longrightarrow} G_n$, there there is a source sequence $\varnothing \overset{tr^*_S}{\Longrightarrow} G_{n,0}$ such that the triple transformation sequence $\varnothing \overset{tr^*_S}{\Longrightarrow} G_{n,0} \overset{tr^*_F}{\Longrightarrow} G_n$ is match consistent and can be composed to the triple sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_n$.

Now, given a model transformation sequence based on forward translation rules $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G_T)$ we have by Thm. 1 thate there is model transformation sequence based on forward rules $(G_S, G_0 \overset{tr^*_{\mathcal{F}}}{\Longrightarrow} G_n, G_T)$, which means by definition that $G_0 \overset{tr^*_{\mathcal{F}}}{\Longrightarrow} G_n$ is source consistent. Source consistency implies by definition that there is a source sequence $\varnothing \overset{tr^*_S}{\Longrightarrow} G_{n,0} = G_0$ such that $\varnothing \overset{tr^*_S}{\Longrightarrow} G_{n,0} \overset{tr^*_F}{\Longrightarrow} G_n$ is match consistent and can be composed to the triple sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_n$ by the composition result Thm. 1 in [1]. This means that the model transformation based on forward translation rules is correct.

Vice versa, given $G_n \in VL$ there is a triple transformation sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_n$ and using the decomposition result in Thm. 1 in [1] we derive the model transformation sequence based on forward rules $(G_S, G_0 \overset{tr^*_{\mathcal{F}}}{\Longrightarrow} G_n, G_T)$ and finally, the equivalence result in Thm. 1 leads to the model transformation sequence based on forward translation rules $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G_T)$, i.e. the model transformation based on forward translation rules is complete.

□

Example 4 (Model Transformation)**.** *Figure 4 shows a triple graph* $G \in VL$*. By Thm. 1 and Thm. 2 we can conclude that the class diagram* G_S *of the source language can be*

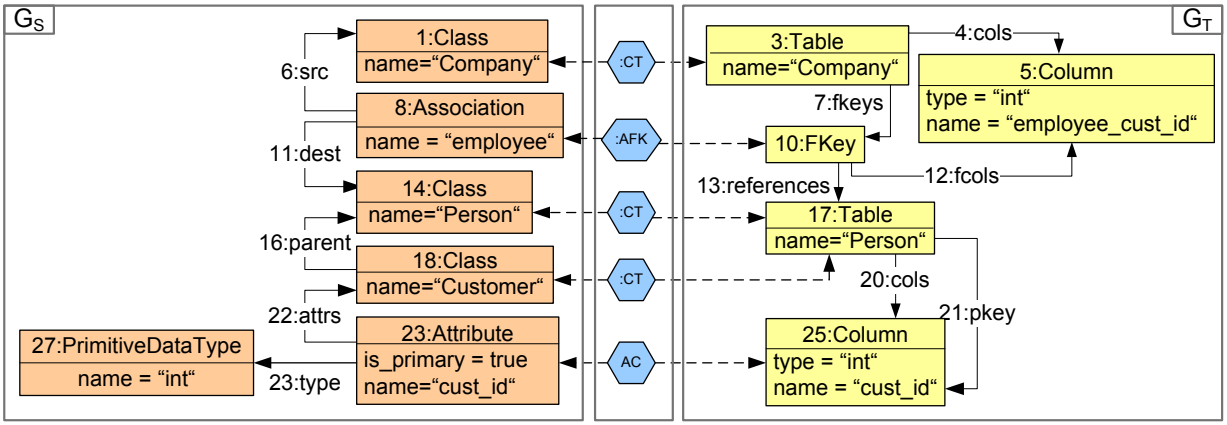


Figure 4: Result of a model transformation after removing translation attributes

translated into the relation database model G_T *by the application of the forward translation rules, i.e. there is a forward translation sequence* $G_0 \overset{tr^*_{FT}}{\Longrightarrow} G_n$ *starting at the source model with translation attributes* $G_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ *and ending at a completely translated model* $G_n = (Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T)$*. Furthermore, any other complete translation sequence leads to the same target model* G_T *(up to isomorphism). We show in Ex. 7 in Sec. 4 that the model transformation has this functional behaviour for each source model.*

4 Analysis of Functional Behaviour

When a rewriting or transformation system describes some kind of computational process, it is often required that it shows a functional behaviour, i.e. every object can be transformed into a unique (terminal) object that cannot be transformed anymore. One way of ensuring this property is proving termination and confluence of the given transformation system. Moreover, if the system is ensured to be terminating, then it suffices to show local confluence according to Newman's Lemma [17].

We now show, how the generation and use of forward translation rules enables us to ensure termination and then to adapt and apply the existing results [3] for showing local confluence of the transformation system leading to functional behaviour of the model transformation.

The standard approach to check local confluence is to check the confluence of all *critical pairs* $P_1 \Leftarrow K \Rightarrow P_2$, which represent the minimal objects where a confluence conflict may occur. The technique is based on two results. On one hand, the *completeness* of critical pairs implies that every confluence conflict $G_1 \Leftarrow G \Rightarrow G_2$ embeds a critical pair $P_1 \Leftarrow K \Rightarrow P_2$. On the other hand, it is also based on the fact that the transformations $P_1 \overset{*}{\Rightarrow} K' \overset{*}{\Leftarrow} P_2$ obtained by confluence of the critical pair can be embedded into transformations $G_1 \overset{*}{\Rightarrow} G' \overset{*}{\Leftarrow} G_2$ that solve the original confluence conflict. However, as shown by Plump [19, 20] confluence of the critical pairs is not sufficient for this purpose, but a slightly stronger version, called strict confluence. This result is also valid for typed attributed

graph transformation systems [3] and we apply them to show functional behaviour of model transformations in the following sense.

Definition 9 (Functional Behaviour of Model Transformations)**.** *A model transformation has* functional behaviour *if each model G_S of the source language $\mathcal{L}_S \subseteq VL_S$ is transformed into a unique terminal model G_T and, furthermore, G_T belongs to the target language VL_T.*

In our approach, we know that the forward translation rules that we generate for performing model transformations are terminating if each of them changes at least one translation attribute. In contrast to that, termination of model transformation sequences based on forward rules requires an additional control structure - being source consistency in [5] or a controlling transformation algorithm as e.g. in [22]. A common alternative way of ensuring termination is the extension of rules by NACs that prevent an application at the same match. However, termination is only one aspect and does not ensure correctness and completeness of the model transformation. In particular, this means that matches must not overlap on effective elements, i.e. elements that are created by the source rule, because this would mean to translate these elements twice. But matches are allowed to overlap on other elements. Since the forward rules are identic on the source part there is no general way to prevent a partial overlapping of the matches by additional NACs and even nested application conditions [12] do not suffice. Nevertheless, in our case study *CD2RDBM* partial overlapping of matches can be prevented by NACs using the created correspondence nodes, but this is not possible for the general case with more complex rules.

Therefore, an analysis of functional behaviour based on the results for local confluence strictly depends on the generation of the system of forward translation rules. This means that, in principle, to prove functional behaviour of a model transformation, it is enough to prove local confluence of the forward translation rules. However, local confluence or confluence may be to strong to show functional behavior in our case. In particular, a model transformation system has a functional behavior if each source model, G_S, can be transformed into a unique target model, G_T. Or, more precisely, that $(Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ can be transformed into a unique completely translated graph $(Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T)$. However, this does not preclude that it may be possible to transform $(Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ into some triple graph $(G'_S \leftarrow G'_C \rightarrow G'_T)$ where not all translation attributes in G'_S are set to true but no other forward translation rule is applicable. This means that, to show the functional behaviour of a set of forward translation rules, it is sufficient to use a weaker notion of confluence, called translation confluence.

Definition 10 (Translation Confluence)**.** *Let TR_{FT} be a set of forward translation rules for the source language $\mathcal{L}_S \subseteq VL_S$. Then, TR_{FT} is* translation confluent *if for every triple graph $G = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ with $G_S \in \mathcal{L}_S \subseteq VL_{S0}$, we have that if $G \overset{*}{\Rightarrow} G_1$ and $G \overset{*}{\Rightarrow} G_2$ and moreover G_1 and G_2 are completely translated graphs, then the target components of G_1 and G_2 are isomorphic, i.e. $G_{1,T} \cong G_{2,T}$.*

The difference between confluence with terminal graphs and translation confluence is that, given $G_1 \overset{*}{\Leftarrow} G \overset{*}{\Rightarrow} G_2$, we only have to care about the confluence of these two transformations if both graphs, G_1 and G_2 are translatable into a completely translated graph

and furthermore, that they do not necessarily coincide on the correspondence part. This concept allows us to show the second main result of this paper in Thm. 3 that characterizes the analysis of functional behaviour of model transformations based on forward translation rules by the analysis of translation confluence, which is based on the analysis of critical pairs.

In Ex. 7 we will show that the set of forward translation rules of our model transformation "*CD2RDBM*" is translation confluent and hence, we have functional behaviour according to the following Thm. 3. In future work we will give sufficient conditions in order to ensure translation confluence, which will lead to a more efficient analysis technique for functional behaviour.

Theorem 3 (Functional Behaviour)**.** *A model transformation based on forward translation rules has functional behaviour, iff the corresponding system of forward translation rules is translation confluent.*

Proof. **"if":** For $G_S \in \mathcal{L}_S \subseteq VL_S$, there is a transformation $\varnothing \overset{tr_S^*}{\Longrightarrow} (G_S \leftarrow \varnothing \rightarrow \varnothing) = G_0$ via source rules leading to a source consistent transformation $G_0 \overset{tr_F^*}{\Longrightarrow} G_n = (G_S \leftarrow G_C \rightarrow G_T)$ (see [5, 1]). Using Fact 1, there is also a complete transformation $G_0' = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing) \overset{tr_{FT}^*}{\Longrightarrow} (Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T) = G_n'$ leading to $(G_S, G_T) \in MTR_{FT}$. For any other complete transformation via forward translation rules $\overline{tr}_{FT}^*$ we have $G_0' \overset{\overline{tr}_{FT}^*}{\Longrightarrow} (Att^{\mathbf{T}}(G_S) \leftarrow G_C' \rightarrow G_T')$. Translation confluence implies that $G_T \cong G_T'$, i.e. G_T is unique up to isomorphism.

"only if": For $G_S \in \mathcal{L}_S \subseteq VL_S$, suppose $G \overset{*}{\Rightarrow} G_1$ and $G \overset{*}{\Rightarrow} G_2$ with $G = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ and G_1, G_2 are completely translated. This means that $(G_S, G_{1,T}), (G_S, G_{2,T}) \in MTR_{FT}$, and the functional behaviour of the model transformation implies that $G_{1,T} \cong G_{2,T}$. □

In order to provide tool support for the analysis of functional behaviour of model transformations we apply the flattening construction as presented in [4] for triple graphs and derive a "plain" graph grammar GG. The analysis of GG can be performed using the implemented critical pair analysis of the tool AGG [24] for typed attributed graph grammars which allows to generate and analyze all critical pairs of a grammar. In order to apply the flattening construction we additionally require that the correspondence component TG_C of the type graph TG is discrete, i.e. has no edges. This condition is fulfilled for our case study and many others as well. An extension of the tool AGG to general triple graphs will be part of future work.

The flattening of a triple graph $G = (G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T)$ is a (single plain) graph $\mathcal{F}(G)$ obtained by disjoint union of the components G_S, G_C and G_T extended by additional edges $Link_S$ and $Link_T$, which encode the internal morphisms s_G and t_G.

Definition 11. Flattening Construction: *Given a triple graph* $G = (G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T)$ *the* flattening $\mathcal{F}(G)$ *of* G *is a plain graph defined by the disjoint union* $\mathcal{F}(G) = G_S + G_C + G_T + Link_S(G) + Link_T(G)$ *with links (additional edges) defined by*

$Link_S(G) = \{(x,y) \mid x \in V_{G_C}, y \in V_{G_S}, s_G(x) = y\}$,
$Link_T(G) = \{(x,y) \mid x \in V_{G_C}, y \in V_{G_T}, t_G(x) = y\}$
with $src_{\mathcal{F}(G)}((x,y)) = x$ *and* $tgt_{\mathcal{F}(G)}((x,y)) = y$ *for* $(x,y) \in Link_S \cup Link_T$. *Given a triple graph morphism* $f = (f_S, f_C, f_T) : G \to G'$ *the flattening* $\mathcal{F}(f) : \mathcal{F}(G) \to \mathcal{F}(G')$ *is defined by* $\mathcal{F}(f) = f_S + f_C + f_T + f_{LS} + f_{LT}$ *with* $f_{LS} : Link_S(G) \to Link_S(G')$, $f_{LT} : Link_T(G) \to Link_T(G')$ *defined by* $f_{LS}((x,y)) = (f_C(x), f_S(y))$ *and* $f_{LT}((x,y)) = (f_C(x), f_T(y))$.

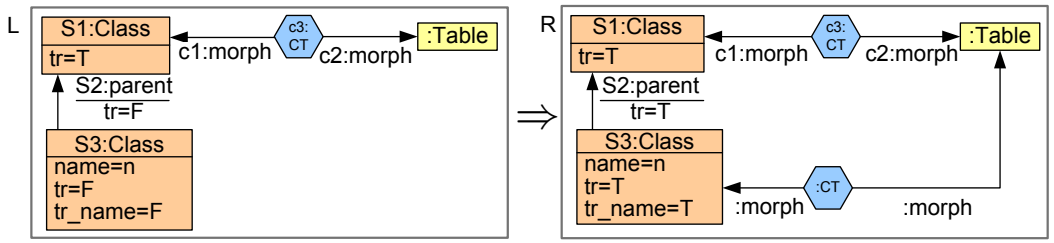


Figure 5: Flattening of the forward translation rule $Subclass2Table_{FT}$

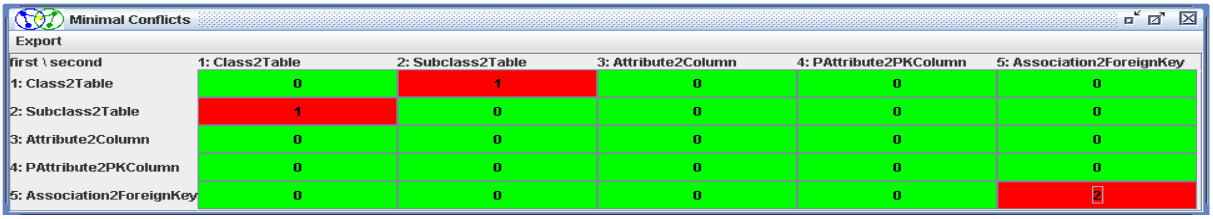
Example 5 (Flattened Forward Translation Rule)**.** *Figure 5 shows the result of the flattening construction applied to the forward translation rule* $Subclass2Table_{FT}$*, which is depicted in the right part of Fig. 3. The triple graphs are flattened to plain graphs, where each mapping of the internal graph morphisms of the triple graphs is encoded as an explicit edge of type morph denoted by a solid line. The figure shows the one-to-one relationship between the forward translation rule and the flattened rule.*

Using Thm. 1 we know that the system of forward translation rules has the same behaviour as the system of forward rules controlled by the source consistency condition. Therefore, it suffices to analyze the pure transformation system of forward translation rules without any additional control condition. This allows us furthermore, to transfer the analysis from a triple graph transformation system to a plain graph transformation system using Thm. 2 in [4], which states that there is a one-to-one correspondence between a triple graph transformation sequence and its flattened plain transformation sequence. Hence, we can analyze confluence, in particular critical pairs, of a set of triple rules by analyzing the corresponding set of flattened rules. This allows us to use the tool AGG for the generation of critical pairs for the flattened forward translation rules.

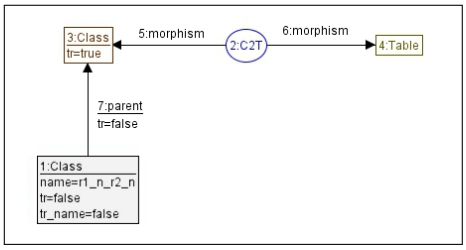
Example 6 (Generation of Critical Pairs)**.** *The tool AGG generates four critical pairs for the flattened forward translation rules of CD2RDBM using the maximum multiplicity constraints according to Fig. 1.*

The overlapping graph for the combination $(SC2T, C2T)$ *is the same as for the combination* $(C2T, SC2T)$ *and it is shown in Fig. 6b. Given a subclass node then both rules,* $SC2T_{FT}$ *and* $C2T_{FT}$ *are applicable, but lead to different results. We will show in Ex. 7 that these critical pairs do not affect the functional behaviour.*

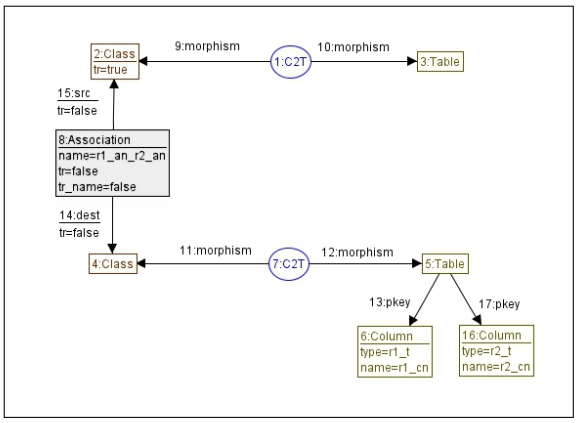
The overlapping graphs for the combination $(A2FK, A2FK)$ *contain in both cases a table with two primary keys. Assume that we can embed this overlapping graph into a*



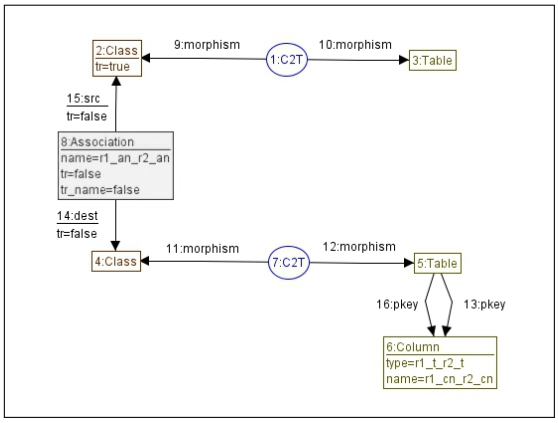
(a) Table of generated critical pairs



(b) Overlapping graph for $(SC2T, C2T)$



(c) $(A2FK, A2FK)$: overlapping 1

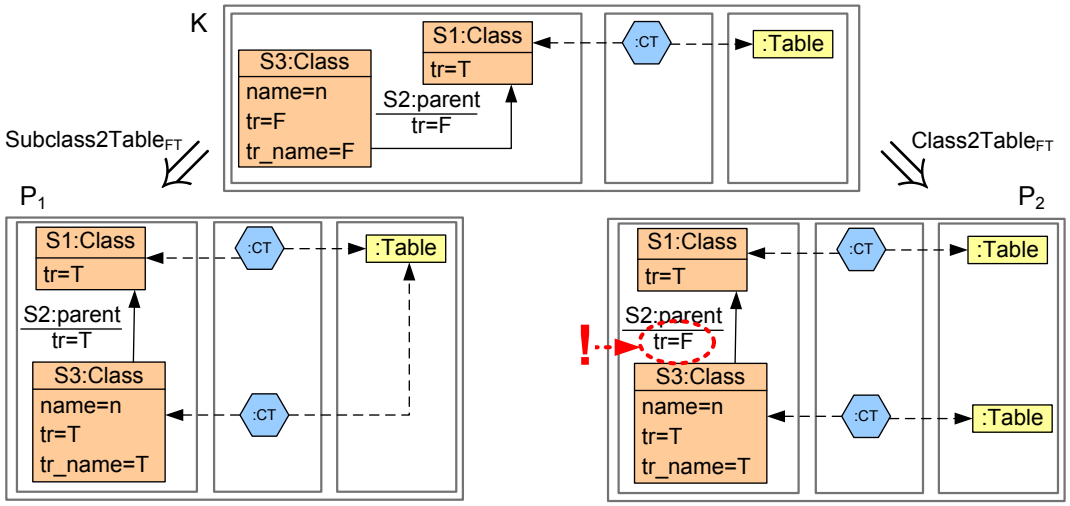


(d) $(A2FK, A2FK)$: overlapping 2

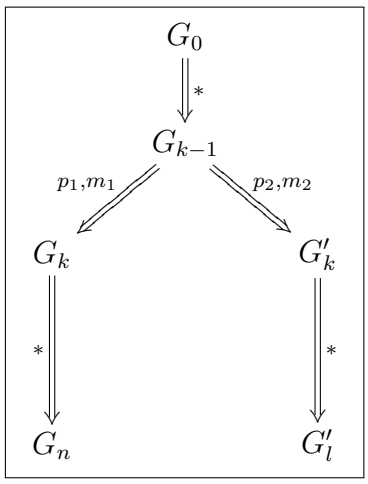
Figure 6: Critical pair generation in AGG with overlapping graphs

flattened intermediate graph of a transformation sequence via the forward translation rules starting with a valid source model. This implies that the table is related to a set of classes with a shared root class w.r.t. the parent edges. However, the source language forbids primary attributes for subclasses and allows at most one primary attribute for the top most class within a hierarchy according to Ex. 1. This means that the forward translation rule "PrimaryAttr2Column$_{FT}$" will never create a second primary attribute for a table and both overlapping graphs cannot be embedded into any intermediate graph of a model transformation sequence.

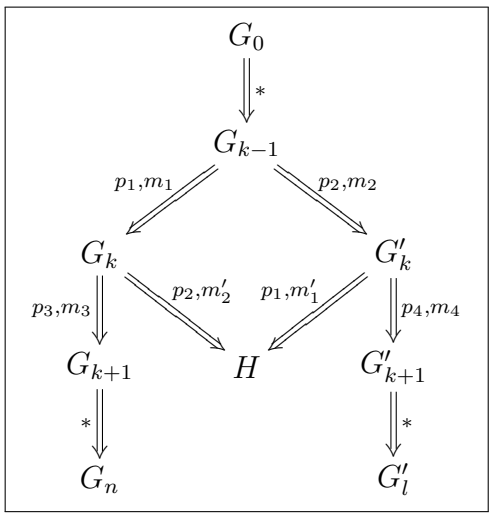
Example 7 (Functional Behaviour). *We show functional behaviour of the model transformation CD2RDBM using Thm. 3. But note that that we focus on the source language of class diagrams $CD = \mathcal{L}_S \subset VL_S$ as specified in Ex. 1, i.e. class diagrams, where subclasses*



(a) Critical pair for the rules $Subclass2Table_{FT}$ and $Class2Table_{FT}$



(b) Diverging Situation



(c) Case for parallel independence

Figure 7: Diverging sequences s_1 and s_2

*do not have primary attributes and the top most super classes may have at most one primary attribute to avoid confusion. The system is terminating, because all rules are source creating, but the system is not confluent w.r.t. terminal graphs. The critical pairs for the combination $(A2FK, A2FK)$ can be neglected, because the overlapping graph cannot be embedded into any intermediate graph of a transformation via the forward translation rules as explained in Ex. 6. The remaining two critical pairs are symmetric, thus it is sufficient to consider the pair $(P_1 \xleftarrow{SC2T_{FT}} K \xrightarrow{C2T_{FT}} P_2)$ shown in Fig. 7a. The edge "S2" is labeled with "***F***" while its source node is labeled with "***T***". The only forward translation rule which can change the translation attribute of a "parent"-edge is "$SC2T_{FT}$", but it requires that the source node is labeled with "**F**". Thus, no forward translation sequence where rule "$C2T_{FT}$"*

is applied to a source node of a parent edge, will lead to a completely translated graph.

Now, assume that our system is not translation confluent by two diverging complete forward translation sequences $s_1 = (G \overset{*}{\Rightarrow} G_n)$ *and* $s_2 = (G \overset{*}{\Rightarrow} G'_l)$ *as shown in Fig. 7b. If the first diverging pair of steps in* s_1 *and* s_2 *is parallel dependent we can embed the critical pair and have that one sequence is incomplete, because the particular edge "S2" remains untranslated. Otherwise, the steps are parallel independent and we can merge them using the Local Church Rosser (LCR) Thm. leading to possibly two new diverging pairs of steps* $(G_{k+1} \overset{p_3,m_3}{\Longleftarrow} G_k \overset{p_2,m'_2}{\Longrightarrow} H)$ *as shown in Fig. 7c. If they are dependent we can embed the critical pair. If the rule* $p = 3 = C2T_{FT}$ *we can conclude that* G_n *is not completely translated. Thus, we have that* $p_2 = C2T_{FT}$ *and by LCR we can reflect this step back to* $G_{k-1} \overset{p_2,m_2}{\Longrightarrow} G'_k$ *and have that* G'_k *cannot be completely translated. This means that the diverging steps* $(G_{k+1} \overset{p_3,m_3}{\Longleftarrow} G_k \overset{p_2,m'_2}{\Longrightarrow} H)$ *are again parallel independent. By induction this leads to the final situation* $(G_n \overset{p^*,m^*}{\Longleftarrow} G_{n-1} \overset{p_2,m''_2}{\Longrightarrow} H)$ *and we have can conclude that the steps are parallel independent. Since* G_n *is completely translated we have that* $H \cong G_n$ *and all together* $G_n \cong G'_l$ *because we have termination.*

Thus, the system is translation confluent and we can apply Thm. 3 showing the functional behaviour of the model transformation CD2RDBM for the considered source language $\mathcal{L}_S = CD$.

5 Model Transformation via Interfaces

During the execution of a model transformation the given source model may be simultaneously used by other applications within an MDA environment and therefore, the model transformation should not modify the source model. Considering our case study, the model transformation transforms class diagrams to data base tables. However, the class diagram may be additionally used for documenting the system structure and thus, should be available unchanged for the software development groups. Furthermore, other interrelated models may rely on a synchronized connection to the class diagram, e.g. a synchronization with corresponding block diagrams is common in the automotive domain as presented in [9].

For this reason, we now present how the concept of model transformations based on forward translation rules with translation attributes can be equivalently implemented using a marking structure that points to the handled elements of the source model leaving the source model itself unchanged. This way the additional structure necessary for ensuring the correctness and completeness of the model transformation is externalized from the source model and kept separately. More precisely, a triple graph consisting of the source, correspondence and target model is extended by an additional triple graph, called interface graph, which specifies the elements of the source model that have been translated so far. This means that the boolean valued translation attributes are represented by the presence and absence of elements in the interface graph. This way, the concept of translation attributes can be used for the analysis of functional behaviour of a model transformation,

while the equivalent concept using interfaces is used for implementations that need to ensure the preservation of the source model.

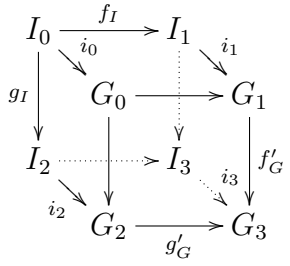
Definition 12 (Category of Triple Graphs with Interfaces)**.** *A* triple graph with interface $IG = (I_{IG}, G_{IG}, i_{IG})$ *is given by a triple graph morphism* $i_{IG} : I_{IG} \to G_{IG}$ *in the category* **TrGraphs** *of triple graphs, where* I_{IG} *is the* interface *for the triple graph* G_{IG}*. A morphism* $m : IG_1 \to IG_2$ *between triple graphs with interfaces with* $(IG_k = (I_k, G_k, i_k))_{(k=1,2)}$ *is given by a pair* $m = (m_I, m_G)$ *of triple graph morphisms* $m_I : I_1 \to I_2$ *and* $m_G : G_2 \to G_2$ *compatible with the interface morphisms, i.e.* $i_2 \circ m_I = m_G \circ i_1$*. The category* **TrGraphsI** *consists of triple graphs with interfaces as objects and morphisms between triple graphs with interfaces as morphisms.*

$$\begin{array}{ccc} I_1 & \xrightarrow{i_1} & G_1 \\ {\scriptstyle m_I}\downarrow & (=) & \downarrow{\scriptstyle m_G} \\ I_2 & \xrightarrow{i_2} & G_2 \end{array}$$

Transformation steps within the category of triple graphs with interfaces are constructed componentwise, i.e. by two pushouts, one for the interface triple graph and one for the main triple graph. The new interface morphism of the resulting triple graph with interface is induced by the pushout property, such that a rule is applicable at any match.

Definition 13 (Transformations in **TrGraphsI**)**.** *A rule* tr *in* **TrGraphsI** *is an injective morphism* $tr : L \to R$*. Given a morphism* $m : L \to IG$*, called match, the transformation step* $IG \overset{tr}{\Longrightarrow} IH$ *is given by a pushout in* **TrGraphsI***, which is constructed componentwise for the* I*- and* G*-components and the new interface morphism* i_{IH} *is induced by the pushout in the* I*-component. The transformation step is* interface-consistent*, if* $(I_0 \xrightarrow{f_I} I_1 \xrightarrow{f'_G \circ i_1} G_3 \xleftarrow{g'_G \circ i_2} I_2 \xleftarrow{f_I} I_0)$ *in Cube* (2) *beneath is a pullback in* **TrGraphs***.*

$$\begin{array}{ccc} IG_0 & \xrightarrow{f} & IG_1 \\ {\scriptstyle g}\downarrow & (PO) & \downarrow{\scriptstyle g'} \\ IG_2 & \xrightarrow{f'} & IG_3 \end{array} \quad (1)$$



(2)

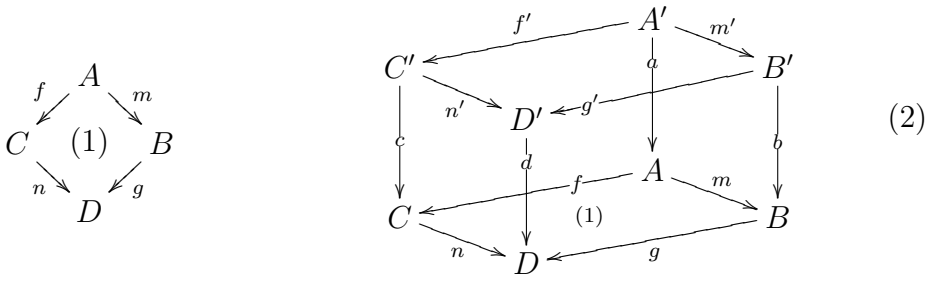
Moreover, triple graphs with interfaces form an $\mathcal{M}$-adhesive category as presented in [6], which are a generalization of weak adhesive HLR in [3]. This way, the important HLR results valid for all $\mathcal{M}$-adhesive categories are available.

Definition 14 ($\mathcal{M}$-adhesive category)**.** *A pair* $(\mathbf{C}, \mathcal{M})$ *containing a category* $\mathbf{C}$ *and a morphism class* $\mathcal{M}$ *is called a* $\mathcal{M}$*-adhesive category if:*

1. $\mathcal{M}$ *is a class of monomorphisms closed under isomorphisms, composition* $(f : A \to B \in \mathcal{M}, g : B \to C \in \mathcal{M} \Rightarrow g \circ f \in \mathcal{M})$*, and decomposition* $(g \circ f \in \mathcal{M}, g \in \mathcal{M} \Rightarrow f \in \mathcal{M})$*.*

2. **C** *has pushouts and pullbacks along* $\mathcal{M}$*-morphisms, and* $\mathcal{M}$*-morphisms are closed under pushouts and pullbacks.*

3. *Pushouts in* **C** *along* $\mathcal{M}$*-morphisms are weak VK squares, i.e. the VK square property holds for all commutative cubes with* $m, a, b, c, d \in \mathcal{M}$ *(see 8).*



(2)

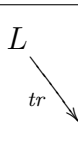
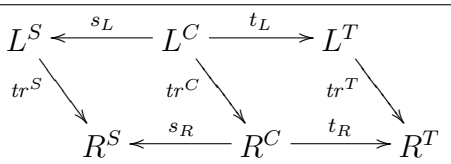
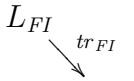
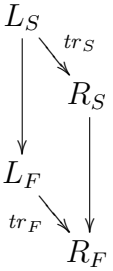
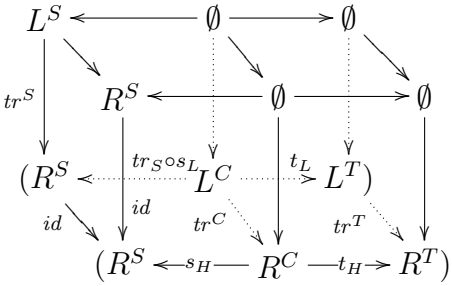
Figure 8: Pushout (1) and commutative cube (2) for VK property.

Fact 2. *The category* $(\mathbf{TrGraphsI}, \mathcal{M})$ *with the class* $\mathcal{M}$ *of morphisms that consist of two triple graph morphisms in* $\mathcal{M}$ *for* $(\mathbf{TrGraphs}, \mathcal{M})$ *is an* $\mathcal{M}$*-adhesive category.*

Proof. The category **TrGraphsI** can be constructed as comma category $\mathbf{CommCat}(F, G, I)$ with $I = \{1\}$, $F = G = ID_{TrGraphs}$. We further have that $(\mathbf{TrGraphs}, \mathcal{M})$ is an $\mathcal{M}$-adhesive category with $\mathcal{M}$ the class of morphisms that consist of attributed $\mathcal{M}$-morphisms for each triple component within $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$. Since F and G preserve pushouts and pullbacks we have by item 4 in Thm. 4.15 in [3] that $(\mathbf{TrGraphsI}, \mathcal{M})$ is a weak adhesive HLR category and hence, also an $\mathcal{M}$-adhesive category. □

In order to perform model transformations based on triple rules with interfaces the operational rules, called forward translation rules with interfaces, are derived analogously to the forward translation rules with translation attributes in Def. 4. The boolean values of the translation attributes correspond to the absence (**F**) and presence (**T**) of the elements in the source component. This means that the effective elements of a forward rule are created within the interface part of the forward translation rule with interfaces and all other source elements are preserved within the interface part. Moreover, the correspondence and target components of the interface graphs are always empty.

Definition 15 (Forward Translation Rule with Interfaces). *Given a triple rule* $tr = (tr^S, tr^C, tr^T) : L \to R$ *with* $(tr^X : L^X \to R^X)_{(X=S,C,T)}$ *its forward translation rule with interfaces* $tr_{FI} : IL \to IR$ *is a morphism in* **TRGraphsI** *with*
$I_{IL} = (L^S \leftarrow \emptyset \to \emptyset)$, $G_{IL} = (R^S \xleftarrow{tr^S \circ s_L} L^C \xrightarrow{t_L} L^T)$,
$I_{IR} = (R^S \leftarrow \emptyset \to \emptyset)$, $G_{IR} = (R^S \xleftarrow{s_R} R^C \xrightarrow{t_R} R^T)$, *and*
$tr_{FI,I} = (tr_S, \emptyset, \emptyset)$, $tr_{FI,G} = (id_{R^S}, tr^C, tr^C)$.

	TrGraphsI	**TrGraphs**	Graphs
triple rule tr			
forward translation rule with interfaces tr_{FI}			

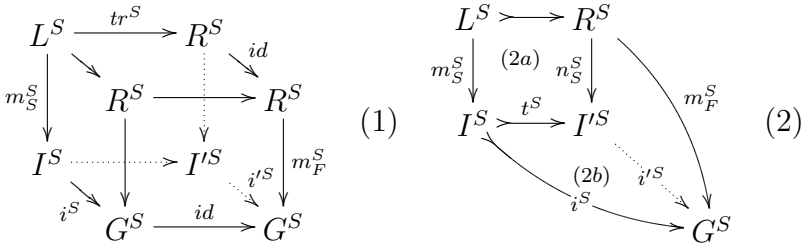
Def. 15 shows that the forward translation rule with interfaces tr_{FI} of a triple rule tr is composed of the source rule tr_S and the forward rule tr_F, where the source rule concerns the interfaces. In order to perform model transformations along almost injective matches as in Sec. 3 we lift the notion of almost injective matches to the case with interfaces by requiring that both the interface and the main components are almost injective in the category of triple graphs.

Definition 16 (Almost injective Match in **TrGraphsI**). *An almost injective match* $m_{FI} = (m_S, m_F)$ *in* **TrGraphsI** *is given by two almost injective matches* m_S, m_F *in* **TrGraphs** *according to Def. 5.*

In Thm. 4 we show that interface consitency is a sufficient and necessary condtion for the correctness and completeness of model transformations in **TrGraphsI**. Hence, we first characterize interface consistency by showing that the pullback condition is equivalent to the condition that the induced interface morphism is an $\mathcal{M}$-morphism.

Fact 3 (Characterization of Interface Consistency). *Let* $IG \overset{tr_{FI}, m_{FI}}{\Longrightarrow} IG'$ *be a transformation in* **TrGraphsI** *via a forward translation rule with interface and an almost injective match, where the interface morphism* $i_{IG} : I \to G$ *is in* $\mathcal{M}$. *Then, the transformation is interface consistent iff the induced interface morphism* $i' : I' \to G'$ *is an* $\mathcal{M}$*-monomorphism.*

Proof. **Direction "⇒":** According to Def. 13 for interface consistency we have the following pullback (2) for the source component with respect to the source component of the transformation step shown in diagram (1):



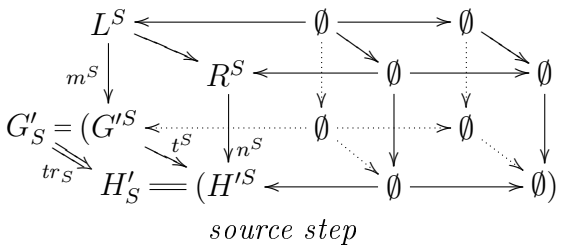
The morphism $i^S \in \mathcal{M}$ by assumption and $t^S \in \mathcal{M}$ because $\mathcal{M}$-morphisms are preserved by pushouts. This implies for the algebra part that t^S_D and i^S_D are isomorphisms and by commutativity of $(2b)$ we derive that $i'^S = t^S_D \circ (i^S_D)^{-1}$ is an isomorphism. The match is a forward translation match and thus, by Def. 5 we have that it is injective on all parts except on the data values. The pushout (1) is constructed componentwise for each E-graph component and thus, we can analyze i' for each component separately. Using Thm. 4.7 in [16] for effective unions in adhesive categories and thus in particular for **Sets** in each of the remaining E-graph components we derive that i'^S is injective .

Concerning the complete triple morphism i' we have that the correspondence and target component of the interface part of the rule tr_{FI} consists of empty graphs and empty morphism and therefore i' coincides with i on these components. This leads to $i' \in \mathcal{M}$.

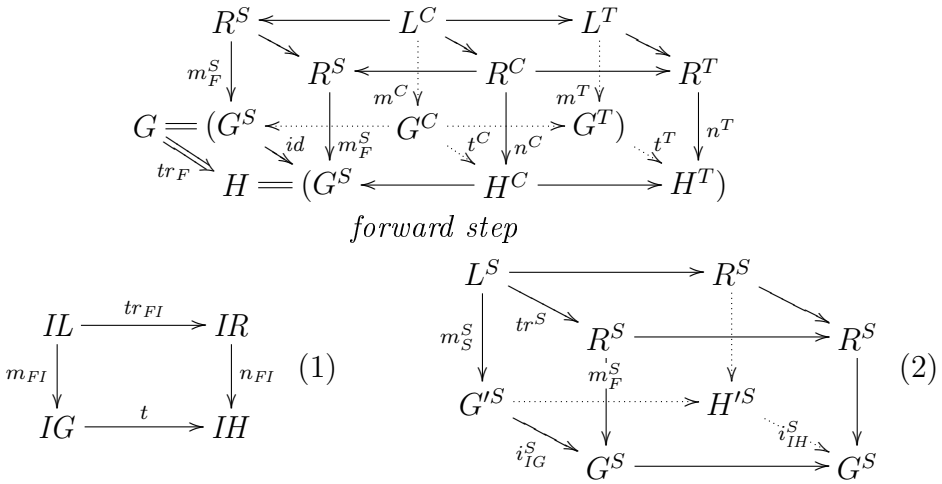
Direction "⇐": The square $(2a)$ is a pushout along an $\mathcal{M}$-morphism and thus a pullback. Pullbacks can be extended by $\mathcal{M}$-monomorphisms, because pullbacks can be extended by monomorphisms in general. Therefore, Diagram (2) is a pullback. □

Fact 4 below shows that the application of a forward translation rule with interfaces is composed of transformation steps in **TrGraphs** using the source and forward triple rules in a compatible way. This builds the basis for showing the correctness and completeness of model transformations based on forward translation rules with interfaces in Thm. 4.

Fact 4 (Transformation via a Forward Translation Rule with Interfaces)**.** *Let* $IG = (G'_S, G, i_G)$ *be a triple graph with interface, where* $G = (G^S \leftarrow G^C \rightarrow G^T)$, $G'_S = (G'^S \leftarrow \emptyset \rightarrow \emptyset)$. *Let further* $tr_{FI} : IL \rightarrow IR$ *be a forward translation rule with interfaces of a triple rule* $tr = (tr^S, tr^C, tr^T) : L \rightarrow R$. *A transformation step* $IG \overset{tr_{FI}, m_{FI}}{\Longrightarrow} IH$ *in* **TrGraphsI** *is given by the source and forward transformation steps* $I_{IG} \overset{tr_S, m_S}{\Longrightarrow} I_{IH}$ *and* $G_{IG} \overset{tr_F, m_F}{\Longrightarrow} G_{IH}$ *with matches* $(m_S, m_F) = m_{FI}$ *depicted below. The interface morphism* i_{IH} *is induced by by the pushout in the source step as shown in Cube* (2) *for the source component using* $m^S_F \circ tr^S = i^S_{IG} \circ m^S_S$ *by* m_{FI} *being a morphism in* **TrGraphsI**, *while for the correspondence and target component we have* $i^C_{IH} = \emptyset$ *and* $i^T_{IH} = \emptyset$.



source step



TrGraphsI *source component in* **Graphs**

Proof. It remains to show that (1) is a pushout. First of all, t and n_{FI} are morphisms in **TrGraphsI** by the commutativity with the induced morphism i_{IH}, which is direct for the correspondence and target component with empty graphs and presented for the source component in (2). Diagram (1) commutes, because it commutes componentwise.

Now, let $(IX = (X', X, i_{IX}), x_1 = (x_{1,I}, x_{1,G}) : IG \to IX, x_2 = (x_{2,I}, x_{2,G}) : IR \to IX)$ be a comparison object. This implies, that $(X', x_{1,I} : G' \to X', x_{2,I} : R' \to X')$ is a comparison object for the pushout in **TrGraphs** given by the source step and $(X, x_{1,G} : G \to X, x_{2,G} : R \to X)$ is a comparison object for the pushout in **TrGraphs** given by the forward step. We derive the induced morphism $h = (h_I, h_G) : IH \to IX$. It remains to show that h is compatible with the interfaces, i.e. $h_G \circ i_{IH} = i_{IX} \circ h_I$. This is direct for the correspondence and target components, because $H'^C = H'^T = \emptyset$. For the source component we have that $(X, x_{1,G} \circ i_{IG}^S, x_{2,G} \circ i_{IR}^S)$ is also a comparison object for the pushout (2) and we derive a unique $f : H'^S \to X$ with $f \circ n^S = x_{1,G}^S \circ id$ and $f \circ t^S = x_{2,G}^S \circ i_{IG}^S$. Both conditions are also valid for $f = h_G \circ i_{IH}$ and for $f = i_{IX} \circ h_I$, such that $h_G \circ i_{IH} = i_{IX} \circ h_I$ by uniqueness of f. Thus, (1) is a pushout in **TrGraphsI** □

Now, we define model transformations based on forward translation rules with interfaces in the same way as for forward translation rules without interfaces in Derf. 8, where completeness of the forward translation sequence is replaced by interface consistency of the forward translation sequence with interfaces.

Definition 17 (Model Transformation Based on Forward Translation Rules with Interfaces)**.** *A* model transformation sequence $(G_S,\ G'_0 \overset{tr^*_{FI}}{\Longrightarrow} G'_n, G_T)$ *based on forward translation rules with interfaces consists of a source graph* G_S*, a target graph* G_T*, and an interface consistent transformation sequence with interfaces* $G'_0 \overset{tr^*_{FI}}{\Longrightarrow} G'_n$ *with almost injective matches,* $G'_0 = ((\varnothing \leftarrow \varnothing \to \varnothing) \to (G_S \leftarrow \varnothing \to \varnothing))$ *and* $G'_n = ((G_S \leftarrow \varnothing \to \varnothing) \to (G_S \leftarrow G_C \to G_T))$.

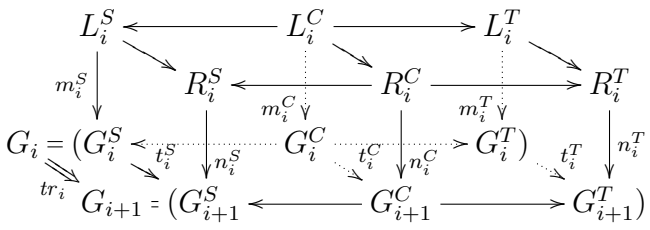
A model transformation $MT : VL_{S0} \Rrightarrow VL_{T0}$ *based on forward translation rules with interfaces is defined by all model transformation sequences* $(G_S, G'_0 \overset{tr^*_{FI}}{\Longrightarrow} G'_n, G_T)$ *based on forward translation rules with* $G_S \in VL_{S0}$ *and* $G_T \in VL_{T0}$. *All these pairs* (G_S, G_T) *define the* model transformation relation $MTR_{FI} \subseteq VL_{S0} \times VL_{T0}$. *The model transformation is* terminating *if there are no infinite* TGT*-sequences via forward translation rules and almost injective matches starting with* $G'_0 = ((\varnothing \leftarrow \varnothing \rightarrow \varnothing) \rightarrow (G_S \leftarrow \varnothing \rightarrow \varnothing))$ *for some source graph* G_S.

The following lemma shows that model transformations in **TrGraphsI** are one to one to triple transformation sequences via the triple rules of the given triple graph grammar. This is the basis for showing the main result of this section in Thm. 4.

Lemma 2 (Forward Translation Sequences with Interfaces)**.** *There is a triple transformation sequence* $\emptyset \overset{tr^*}{\Longrightarrow} G$ *in* **TrGraphs** ***iff*** *there is an interface consistent transformation* $(\emptyset, G_S, \emptyset) \overset{tr^*_{FI}}{\Longrightarrow} (G_S, G, i)$ *in* **TrGraphsI**.

Proof. 1. **Direction "⇒":**

Let $(s1)\emptyset \overset{tr_1,m_1}{\Longrightarrow} \ldots \overset{tr_k,m_k}{\Longrightarrow} G_k = G$ be a derivation in **TrGraphs**.

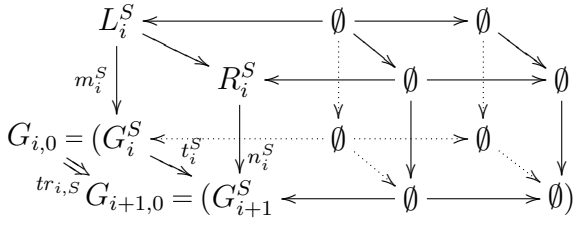


triple step

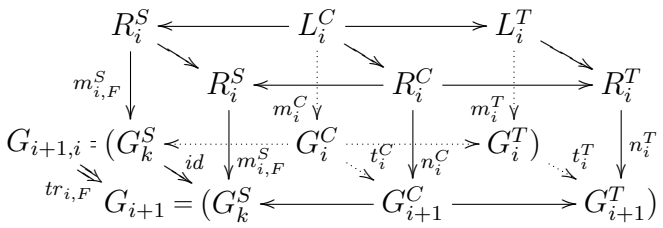
Using the decomposition result (Thm. 1 in [7]) $\Rightarrow \forall\, i \in \{1, \ldots, k\}$: $(G_i^S \leftarrow \emptyset \rightarrow \emptyset) \overset{tr_{i,S}, m_{i,S}}{\Longrightarrow} (G_{i+1}^S \leftarrow \emptyset \rightarrow \emptyset)$ and

$(G_k^S \leftarrow G_i^C \rightarrow G_i^T) \overset{tr_{i,F}, m_{i,F}}{\Longrightarrow} (G_k^S \leftarrow G_{i+1}^C \rightarrow G_{i+1}^T)$ in **TrGraphs**

with $m_{i,F}^S = g_{i+1}^S \circ n_{i,S}^S$. $(*)$



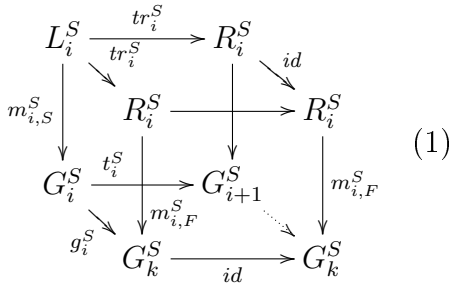
source step



forward step

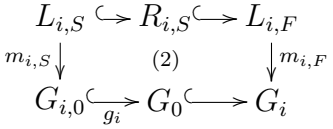
$\Rightarrow \ \forall\ i \in \{1,\ldots,k\} : (G_i^S \leftarrow \emptyset \rightarrow \emptyset) \rightarrow (G_k^S \leftarrow G_i^C \rightarrow G_i^T) \overset{tr_{i,FI}, m_{i,FI}}{\Longrightarrow} (G_{i+1}^S \leftarrow \emptyset \rightarrow \emptyset) \xrightarrow{i_{G_{i+1}}} (G_k^S \leftarrow G_{i+1}^C \rightarrow G_{i+1}^T)$ in **TrGraphsI** with $m_{i,FI} = (m_{i,S}, m_{i,F})$. The transformation step is given by the source and the forward step and the morphism $i_{G_{i+1}}$ is given by the inclusion $g_{i+1} : G_{i+1,0} \rightarrow G_0 = (G_k^S \leftarrow \emptyset \rightarrow \emptyset)$ as in Def. 3 of [5].

In order to show that each step i in **TrGraphsI** is interface-consistent we can first state that the correspondence and target components are pullbacks, because the components of the interfaces I_0, I_1 and I_2 in Def. 13 are the empty graphs. It remains to show the pullback property for the source component.



(1)

Using $(*)$ we can apply Thm. 1 and Def. 4 of [5] and derive the following pullback in **TrGraphs**:



with $G_{i,0} = (G_i^S \leftarrow \emptyset \rightarrow \emptyset)$ and $G_i = (G_k^S \leftarrow G_i^C \rightarrow G_i^T)$. Thus, we have that $(L_i^S \xrightarrow{tr_i^S} R_i^S \xrightarrow{m_{i,F}^S} G_k^S \xleftarrow{g_i^S} G_i^S \xleftarrow{m_{i,S}^S} L_i^S)$ in Cube (1) is a pullback of $(R_i^S \xrightarrow{m_{i,F}^S} G_k^S \xleftarrow{g_i^S} G_i^S)$ in Graphs. Together with the pullbacks in the correspondence and target component and commutativity with the empty morphisms we have the desired pullback in **TrGraphs**.

2. **Direction "⇐":**

   Let $(s2) : (\emptyset \rightarrow G^S) \overset{tr_{FI}^*}{\Longrightarrow} (G_S \rightarrow G)$ be an interface consistent transformation in **TrGraphsI**.

   ⇒ Each step i in $(s2)$ defines a source and a forward transformation step as in the first part. We need to show that the forward match is forward consistent according

to Def. 4 in [5]. By the interface-consistency of $(s2)$ we have the pullback property for Cube (1) before. Thus we derive the pullback property for the source component of Diagram (2) before. Since $G_{i,0}$ has empty graphs on the correspondence and target component we derive the pullback property for Diagram (2) in **TrGraphs**. Using the resulting triple graph with interface in $(s1)$ we know that $g_k = id$ and by point 3 of Thm. 1 in [5] we derive a source consistent forward sequence that leads to the triple sequence $(s1)$ as needed.

□

The main result of this section in Thm. 1 below states that model transformations based on forward translation rules with interfaces are equivalent to those based on forward translation rules without interfaces.

Theorem 4 (Forward Translation Sequences with Interfaces)**.** *Given a triple graph grammar, then the model transformation $MT_{FT} : VL_{S0} \Rrightarrow VL_{T0}$ based on forward translation rules without interfaces and the model transformation $MT_{FI} : VL_{S0} \Rrightarrow VL_{T0}$ based on forward translation rules with interfaces, both with almost injective matches, define the same model transformation relation $MTR_{FT} = MTR_{FI} \subseteq VL_{S0} \times VL_{T0}$.*

Proof. By Lemma 2 we have the equivalence of triple sequence via the triple rules of the given triple graph grammar and the model transformation sequences based on forward translation rules with interfaces. Using the correctness and completeness result for model transformations based on forward translation rules in Thm. 2 we also have the equivalence of model transformation sequences based on forward rules and the triple sequences. Thus, model transformation sequences based on forward translation rules with interfaces are equivalent to model transformation sequences based on forward translation rules without interfaces. □

6 Related Work

As pointed out in the introduction our work is based on triple graph grammars presented by Schürr et.el. in [22, 21, 15] with various applications in [10, 11, 14, 15, 23]. The formal approach to TGGs has been developed in [2, 4, 5, 8, 1, 7]. In [1] it is shown how to analyze bi-directional model transformations based on TGGs with respect to information preservation, which is based on a decomposition and composition result for triple graph transformation sequences.

As shown in [2] and [7], the notion of *source consistency* ensures correctness and completeness of model transformations based on TGGs. A construction technique for correct and complete model transformation sequences *on-the-fly* is presented in [5], i.e. correctness and completeness properties of a model transformation do not need to be analyzed after completion, but are ensured by construction. In this construction, source consistency is checked on-the-fly, which means during and not after the construction of the forward sequence. Moreover, a strong sufficient condition for termination is given. The main construction and results are used for the proof of Fact 1 and hence, also for our first main result

in Thm. 1. Similarly to the generated forward translation rules in this paper, the generated operational rules in [18] also do not need an additional control condition. However, the notion of correctness and completeness is much more relaxed, because it is not based on a given triple graph grammar, but according to a pattern specification, from which usually many triple rules are generated.

A first approach to analyze functional behaviour for model transformations based on TGGs was already given in [8] for triple rules with distinguished kernel typing. This strong restriction requires e.g. that there is no pair of triple rules handling the same source node type - which is, however, not the case for the first two rules in our case study $CD2RDBM$. The close relationship between model transformations based on TGGs and those on "plain graph transformations" is discussed in [4], but without considering the special control condition source consistency. The treatment of source consistency based on translation attributes is one contribution of this paper in order to analyze functional behaviour. As explained in Sec. 3 additional NACs are not sufficient to obtain this result. Functional behaviour for a case study on model transformations based on "plain graphs" is already studied in [3] using also critical pair analysis in order to show local confluence. But the additional main advantage of our TGG-approach in this paper is that we can transfer the strong results concerning termination, correctness and completeness from previous TGG-papers [4, 5] based on source consistency to our approach in Thm. 2 by integrating the control structure source consistency in the analysis of functional behaviour. Finally there is a strong relationship with the model transformation algorithm in [22], which provides a control mechanism for model transformations based on TGGs by keeping track of the elements that are translated so far. In [5] we formalized the notion of elements that are translated at a current step by so-called effective elements. In this paper we have shown that the new translation attributes can be used to automatically keep track of the elements that have been translated so far.

7 Conclusion

In this paper we have analyzed under which conditions a model transformation based on triple graph grammars (TGGs) has functional behaviour. For this purpose, we have shown how to generate automatically forward translation rules from a given set of triple rules, such that model transformations can be defined equivalently by complete forward translation sequences. The main result shows that a terminating model transformation has functional behaviour if the set of forward translation rules is translation confluent. This allows to apply the well-known critical pair analysis techniques for typed attributed graph transformations with support from the tool AGG to the system of forward translation rules, which was not possible before, because the control condition source consistency could not be integrated in the analysis. These techniques have been applied to show functional behaviour of our running example, the model transformation from class diagrams to data base models. In order to keep the source model unchanged during the transformation the translation attributes can be separated from the source model as presented in Sec. 5.

Alternatively, the model transformation can be executed using the on-the-fly construction in [5], which is shown to be equivalent by Thm. 1. In future work we give sufficient conditions in order to check translation confluence, which will further improve the analysis techniques. Moreover, we will extend the results to systems with control structures like negative application conditions (NACs), rule layering and amalgamation. In order to extend the main result concerning functional behaviour to the case with NACs, we have to extend the generation of forward translation rules by extending the NACs with translation attributes and we have to prove the equivalence of the resulting model transformation with the on-the-fly construction in [7].

References

[1] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, and G. Taentzer. Information preserving bidirectional model transformations. In M. B. Dwyer and A. Lopes, editors, *Proc. FASE'07*, volume 4422 of *LNCS*, pages 72–86. Springer, 2007.

[2] H. Ehrig, K. Ehrig, and F. Hermann. From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. In C. Ermel, J. de Lara, and R. Heckel, editors, *Proc. GT-VMT'08*, volume 10 of *EC-EASST*. EASST, 2008.

[3] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs. Springer, 2006.

[4] H. Ehrig, C. Ermel, and F. Hermann. On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars. In G. Karsai and G. Taentzer, editors, *Proc. GraMoT'08*. ACM, 2008.

[5] H. Ehrig, C. Ermel, F. Hermann, and U. Prange. On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars. In A. Schürr and B. Selic, editors, *Proc. ACM/IEEE MODELS'09*, volume 5795 of *LNCS*, pages 241–255. Springer, 2009.

[6] H. Ehrig, U. Golas, and F. Hermann. Categorical Frameworks for Graph Transformation and HLR Systems based on the DPO Approach. *Bulletin of the EATCS*, 2010. To appear.

[7] H. Ehrig, F. Hermann, and C. Sartorius. Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions. In R. Heckel and A. Boronat, editors, *Proc. GT-VMT'09*, volume 18 of *EC-EASST*. EASST, 2009.

[8] H. Ehrig and U. Prange. Formal Analysis of Model Transformations Based on Triple Graph Rules with Kernels. In H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, editors, *Proc. ICGT'08*, volume 5214 of *LNCS*, pages 178–193. Springer, 2008.

[9] H. Giese and R. Wagner. From model transformation to incremental bidirectional model synchronization. *Software and System Modeling*, 8(1):21–43, 2009.

[10] E. Guerra and J. de Lara. Attributed typed triple graph transformation with inheritance in the double pushout approach. Technical Report UC3M-TR-CS-2006-00, Universidad Carlos III, Madrid, Spain, 2006.

[11] E. Guerra and J. de Lara. Model view management with triple graph grammars. In A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, editors, *Proc. ICGT'06*, volume 4178 of *LNCS*, pages 351–366, Heidelberg, 2006. Springer.

[12] A. Habel and K.-H. Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science*, 19:1–52, 2009.

[13] F. Hermann, H. Ehrig, F. Orejas, and U. Golas. Formal Analysis of Functional Behaviour of Model Transformations Based on Triple Graph Grammars. In *Proc. Int. Conf. on Graph Transformation*. Springer, 2010. to appear.

[14] E. Kindler and R. Wagner. Triple graph grammars: Concepts, extensions, implementations, and application scenarios. Technical Report TR-ri-07-284, Department of Computer Science, University of Paderborn, Germany, 2007.

[15] A. Königs and A. Schürr. Tool Integration with Triple Graph Grammars - A Survey. In *Proc. SegraVis School on Foundations of Visual Modelling Techniques*, volume 148 of *ENTCS*, pages 113–150. Elsevier Science, 2006.

[16] S. Lack and P. Sobociński. Adhesive Categories. In *Proc. FOSSACS 2004*, volume 2987 of *LNCS*, pages 273–288. Springer, 2004.

[17] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.

[18] F. Orejas, E. Guerra, J. de Lara, and H. Ehrig. Correctness, completeness and termination of pattern-based model-to-model transformation. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *Proc. CALCO'09*, volume 5728 of *LNCS*, pages 383–397. Springer, 2009.

[19] D. Plump. Hypergraph rewriting: Critical pairs and undecidability of confluence. In *Term Graph Rewriting: Theory and Practice*, pages 201–213. John Wiley, 1993.

[20] D. Plump. Confluence of graph transformation revisited. In *Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, volume 3838 of *LNCS*, pages 280–308. Springer, 2005.

[21] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In G. Tinhofer, editor, *Proc. WG'94*, volume 903 of *LNCS*, pages 151–163. Springer, 1994.

[22] A. Schürr and F. Klar. 15 years of triple graph grammars. In H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, editors, *Proc. ICGT'08*, LNCS, pages 411–425. Springer, 2008.

[23] G. Taentzer, K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovsky, U. Prange, D. Varro, and S. Varro-Gyapay. Model Transformation by Graph Transformation: A Comparative Study. In *Proc. MoDELS 2005 Workshop MTiP'05*, 2005.

[24] TFS-Group, TU Berlin. *AGG*, 2009. `http://tfs.cs.tu-berlin.de/agg`.