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

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

[1] Institut für Softwaretechnik und Theoretische Informatik,
Technische Universität Berlin, Germany
{frank,ehrig,ugolas}@cs.tu-berlin.de
[2] Departament de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya, Barcelona, Spain
orejas@lsi.upc.edu

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 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 [18]. 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 [19,13,8] and various kinds of applications have been presented [20,9,12]. For source-to-target model transformations, so-called

H. Ehrig et al. (Eds.): ICGT 2010, LNCS 6372, pp. 155–170, 2010.

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 [1,3,4,6,7].

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 [4] 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 [4], 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. As shown in [11] the translation attributes can be separated 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 [2]), 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 [21] 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. Related work and our conclusion - including a summary of our results and future work - is presented in Sections 5 and 6, respectively.

2 Review of Triple Graph Grammars

Triple graph grammars [18] 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 [13], 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 [6] to typed, attributed graphs. In this section, we review main constructions and results of model transformations based on triple graph grammars [19,4].

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 $type_G : G \to TG$.

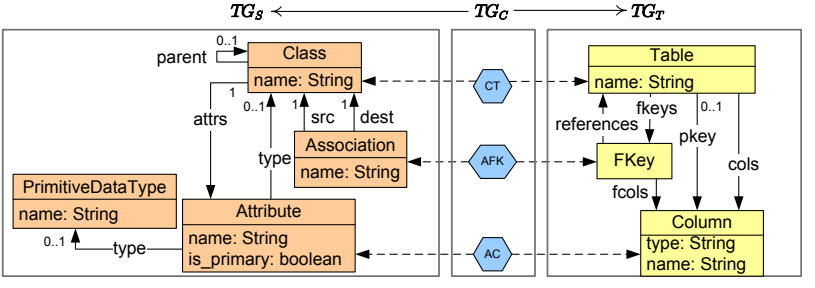


Fig. 1. Triple type graph for *CD2RDBM*

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 [6], 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 $\mathcal{L}_S = CD$ contains only those 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 [6] in the framework of $\mathcal{M}$-adhesive categories (called weak adhesive HLR in [2]).

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.

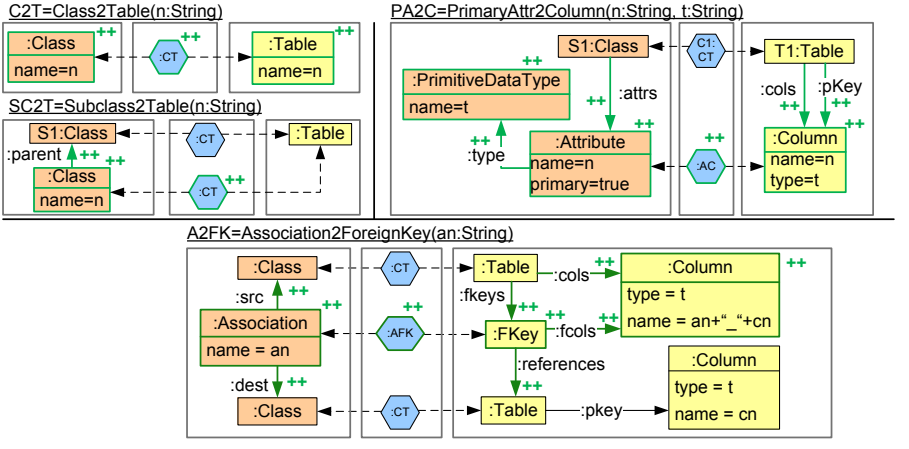
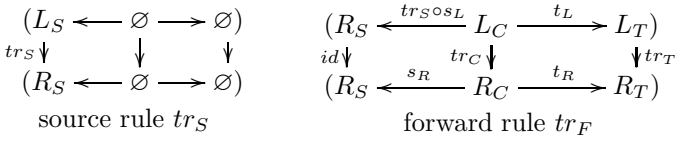


Fig. 2. Rules for the model transformation *Class2Table*

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.



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 [6,4] 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 $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$. A forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is source consistent, if 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. Thus, source consistency is a control condition for the construction of the forward sequence.

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 [6,4] 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 [4].

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.

Definition 2 (Graph with Translation Attributes). *Given an attributed graph $AG = (G, D)$ and a subgraph $G_0 \subseteq G$ 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 G_0 and one boolean-valued attribute tr_x_a for each attribute associated to such an element x in G_0. The set of all these additional translation attributes is denoted by Att_{G_0}. $Att^v_{G_0}$, where $v = \mathbf{T}$ or $v = \mathbf{F}$, denotes a translation graph where all the attributes in Att_{G_0} are set to v. By $AG' = AG \oplus Att_{G_0}$ we specify that AG is extended by the attributes in Att_{G_0}. Moreover, we define $Att^v(AG) := AG \oplus Att^v_G$.*

The extenstion 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 [2]. Thus, we can ensure source consistency by merely using attributes in order to completely translate a model. Therefore, we call these rules forward translation

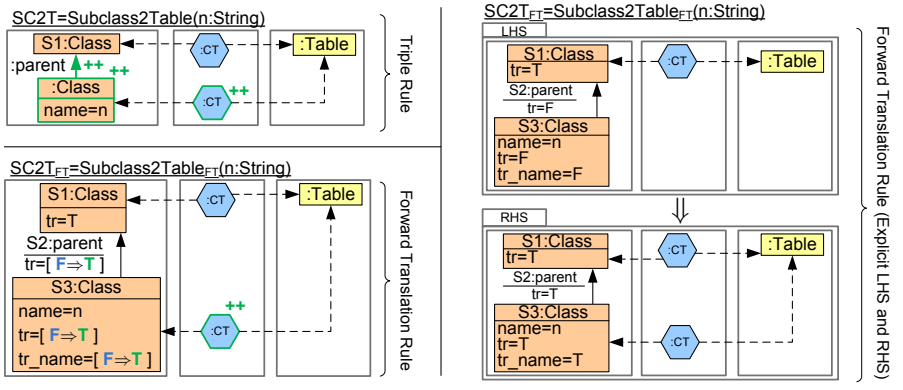


Fig. 3. Forward translation rule $Subclass2Table_{FT}(n : String)$

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 3 (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.*

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.

From the application point of view model transformation rules should be 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 part, 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. This requirement 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 4 (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**").*

Now we are able to show the equivalence of complete forward translation sequences with source consistent forward sequences.

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 (Sketch). Using Thm. 1 in [4] we know that in sequence 1 all matches are forward consistent as defined for the on-the-fly construction in [4]. This allows to show for each step $G_{i-1} \Rightarrow G_i$ starting with $i = 1$ that there is a transformation step $G_{i-1} \overset{tr_{i,F}}{\Longrightarrow} G_i$ iff there is a transformation step $G'_{i-1} \overset{tr_{i,FT}}{\Longrightarrow} G'_i$ leading to the complete equivalence of both sequences as shown in detail in [11]. □

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 [11] in order to keep the source model unchanged.

Definition 5 (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. 5 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 [4]. 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 (Sketch). By Def. 3 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 [4]. The correctness and completeness are based on Thm. 1 above and the proof of Thm. 3 in [3]. □

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 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. 5 in Sec. 4 that the model transformation has this functional behaviour for each source model.

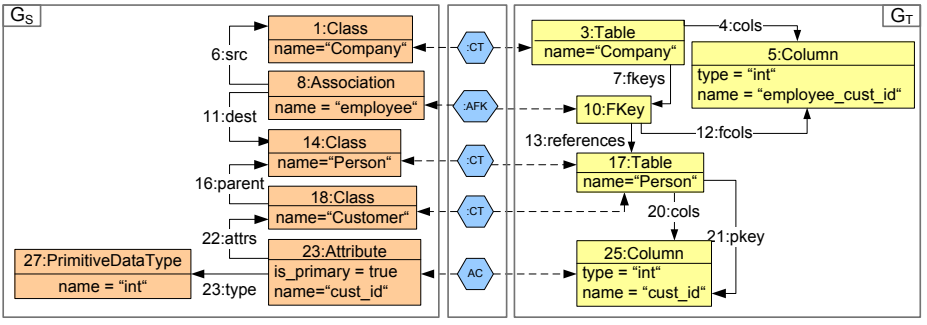


Fig. 4. Result of a model transformation after removing translation attributes

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 [14]. However, an extension of the notion of critical pairs to encompass the additional control condition source consistency directly would be quite complex. Indeed, it would need to cover the interplay between a pair of forward steps and its corresponding pair of source steps at the same time including the relating morphism between them. As a consequence, an extension of the implemented critical pair analysis of AGG would probably be significantly less efficient, because of the generation of more possible overlappings.

We now show, how the generation and use of forward translation rules enables us to ensure termination and to integrate the control structure source consistency in the analysis of functional behaviour, such that we can apply the existing results [2] 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 [16,17] confluence of critical pairs is not sufficient for this purpose, but a slightly stronger version, called strict confluence, which additionally requires that the preserved elements of the given steps are preserved in the merging steps. This result is also valid for typed attributed graph transformation systems [2] and we apply them to show

functional behaviour of model transformations in the following sense, where the source language $\mathcal{L}_S$ may be a subset of VL_S derived from the triple rules.

Definition 6 (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.*

Model transformations based on forward translation rules are terminating, if each rule rewrites at least one translation attribute from "**F**" to "**T**". In contrast to that, termination of model transformations based on forward rules is ensuresed by an additional control structure – either source consistency in [4] or a controlling transformation algorithm as e.g. in [19]. 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 [10] 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 too 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 and 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 7 (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 can be transformed into completely translated graphs 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. 5 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 [4,6]). 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 $G \overset{*}{\Longrightarrow} G_1$ and $G \overset{*}{\Longrightarrow} G_2$ 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 $G_{1,T} \cong G_{2,T}$. □

The flattening construction presented in [3] for triple graph grammars enables us to use the tool AGG [21] for typed attributed graph grammars for generating and analyzing the critical pairs of the system of forward translation rules. The construction requires 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.

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 [3], 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. The additional translation attributes increase the size of the triple rules by one attribute for each element in the source model. However, in order to improve scalability, they can be reduced by grouping those which always occur in the same combination within the rules.

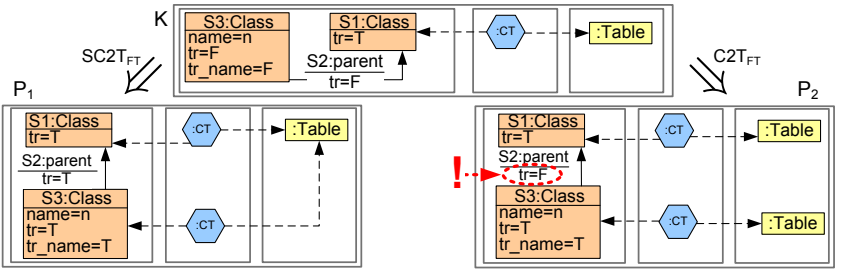


Fig. 5. Critical pair for the rules *Subclass2Table*$_{FT}$ and *Class2Table*$_{FT}$

Example 5 (Functional Behaviour). We show functional behaviour of the model transformation *CD2RDBM*, which is terminating, because all rules are source creating, but the system is not confluent w.r.t. terminal graphs. The tool AGG generates four critical pairs respecting the maximum multiplicity constraints according to Fig. 1. We explain why they can be neglected and refer to [11] for further details. The first two pairs can be neglected, because they refer to class diagrams containing a class with two primary attributes, which is not allowed for the source language $\mathcal{L}_S = CD$ (see Ex. 1). The two remaining critical pairs are identical – only the order is swapped. One pair is shown in Fig. 5. The edge "S2" in the graph P_2 is labeled with "**F**" but its source node is labeled with "**T**". Only the rule "$SC2T_{FT}$" can change the translation attribute of a "parent"-edge, 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. Assuming that our system is not translation confluent by two diverging complete forward translation sequences $s_1 = (G \overset{*}{\Longrightarrow} G_1)$ and $s_2 = (G \overset{*}{\Longrightarrow} G_2)$ leads to a contradiction. 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, we can merge them using the Local Church Rosser (LCR) Thm. leading to possibly two new diverging pairs of steps. If they are dependent we can embed the critical pair and conclude by LCR that the problematic step of the critical pair was also applied in the first diverging situation leading to incompleteness of s_1 or s_2. This procedure is repeated till the end of the sequences and we can conclude that there is no parallel dependent situation and by LCR we have that $G_1 \cong G_2$, because we have termination. The system is translation confluent and we can apply Thm. 3 to show the functional behaviour of the model transformation *CD2RDBM*.

5 Related Work

As pointed out in the introduction our work is based on triple graph grammars presented by Schürr et.el. in [19,18,13] with various applications in [8,9,12,13,20]. The formal approach to TGGs has been developed in [1,3,4,5,6,7]. In [6] 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 [1] 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 [4], 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 [15] 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 [5] 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 [3], 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 [2] 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 [3,4] 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 [19], which provides a control mechanism for model transformations based on TGGs by keeping track of the elements that are translated so far. In [4] 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.

6 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 [11]. Alternatively, the model transformation can be executed using the on-the-fly construction in [4], 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. Ehrig, H., Ehrig, K., Hermann, F.: From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. In: Ermel, C., de Lara, J., Heckel, R. (eds.) Proc. GT-VMT 2008, EC-EASST, EASST, vol. 10 (2008)
2. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs. Springer, Heidelberg (2006)
3. Ehrig, H., Ermel, C., Hermann, F.: On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars. In: Karsai, G., Taentzer, G. (eds.) Proc. GraMoT 2008. ACM, New York (2008)
4. Ehrig, H., Ermel, C., Hermann, F., Prange, U.: On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 241–255. Springer, Heidelberg (2009)
5. Ehrig, H., Prange, U.: Formal Analysis of Model Transformations Based on Triple Graph Rules with Kernels. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS, vol. 5214, pp. 178–193. Springer, Heidelberg (2008)
6. Ehrig, H., Ehrig, K., Ermel, C., Hermann, F., Taentzer, G.: Information preserving bidirectional model transformations. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 72–86. Springer, Heidelberg (2007)

7. Ehrig, H., Hermann, F., Sartorius, C.: Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions. In: Heckel, R., Boronat, A. (eds.) Proc. GT-VMT 2009, EC-EASST, EASST, vol. 18 (2009)
8. Guerra, E., de Lara, J.: Attributed typed triple graph transformation with inheritance in the double pushout approach. Tech. Rep. UC3M-TR-CS-2006-00, Universidad Carlos III, Madrid, Spain (2006)
9. Guerra, E., de Lara, J.: Model view management with triple graph grammars. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) ICGT 2006. LNCS, vol. 4178, pp. 351–366. Springer, Heidelberg (2006)
10. Habel, A., Pennemann, K.H.: Correctness of high-level transformation systems relative to nested conditions. Mathematical Structures in Computer Science 19, 1–52 (2009)
11. Hermann, F., Ehrig, H., Golas, U., Orejas, F.: Formal Analysis of Functional Behaviour for Model Transformations Based on Triple Graph Grammars - Extended Version. Tech. Rep. TR 2010-8, TU Berlin (2010), http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/2010
12. Kindler, E., Wagner, R.: Triple graph grammars: Concepts, extensions, implementations, and application scenarios. Tech. Rep. TR-ri-07-284, Department of Computer Science, University of Paderborn, Germany (2007)
13. Königs, A., Schürr, A.: Tool Integration with Triple Graph Grammars - A Survey. In: Proc. SegraVis School on Foundations of Visual Modelling Techniques. ENTCS, vol. 148, pp. 113–150. Elsevier Science, Amsterdam (2006)
14. Newman, M.H.A.: On theories with a combinatorial definition of "equivalence". Annals of Mathematics 43(2), 223–243 (1942)
15. Orejas, F., Guerra, E., de Lara, J., Ehrig, H.: Correctness, completeness and termination of pattern-based model-to-model transformation. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO 2009. LNCS, vol. 5728, pp. 383–397. Springer, Heidelberg (2009)
16. Plump, D.: Hypergraph rewriting: Critical pairs and undecidability of confluence. In: Term Graph Rewriting: Theory and Practice, pp. 201–213. John Wiley, Chichester (1993)
17. Plump, D.: Confluence of graph transformation revisited. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F., de Vrijer, R. (eds.) Processes, Terms and Cycles: Steps on the Road to Infinity. LNCS, vol. 3838, pp. 280–308. Springer, Heidelberg (2005)
18. Schürr, A.: Specification of Graph Translators with Triple Graph Grammars. In: Mayr, E.W., Schmidt, G., Tinhofer, G. (eds.) WG 1994. LNCS, vol. 903, pp. 151–163. Springer, Heidelberg (1995)
19. Schürr, A., Klar, F.: 15 years of triple graph grammars. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS, vol. 5214, pp. 411–425. Springer, Heidelberg (2008)
20. Taentzer, G., Ehrig, K., Guerra, E., de Lara, J., Lengyel, L., Levendovsky, T., Prange, U., Varro, D., Varro-Gyapay, S.: Model Transformation by Graph Transformation: A Comparative Study. In: Proc. MoDELS 2005 Workshop MTiP 2005 (2005)
21. TFS-Group, TU Berlin: AGG (2009), http://tfs.cs.tu-berlin.de/agg