Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars -Extended Version

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

Bericht-Nr. 2010-13
ISSN 1436-9915

Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars - Extended Version

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

[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 are a well-established, formal and intuitive concept for the specification and analysis of bidirectional model transformations. In previous work we have formalized and analyzed already termination, correctness, completeness, local confluence and functional behaviour.

In this paper, we show how to improve the efficiency of the execution and analysis of model transformations in practical applications by using triple rules with negative application conditions (NACs). In addition to *specification NACs*, which improve the specification of model transformations, the generation of *filter NACs* improves the efficiency of the execution and the analysis of functional behaviour supported by critical pair analysis of the tool AGG. We illustrate the results for the well-known model transformation from class diagrams to relation database models.

Keywords: Model Transformation, Triple Graph Grammars, Functional Behaviour

1 Introduction

Model transformations based on triple graph grammars (TGGs) have been introduced by Schürr in [17]. TGGs are grammars that generate languages of graph triples, consisting of a source graph G_S and a target graph G_T, together with a correspondence graph G_C "between" them. From a TGG, operational rules can be derived which define various model transformation and integration tasks, such as consistency checking, consistency recovery and bidirectional model transformation. Since 1994, several extensions of the original TGG definitions have been published [18, 15, 8], and various kinds of applications have been presented [20, 9, 14].

For source-to-target model transformations, so-called *forward* transformations, forward rules are derived 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, completeness, efficient execution and — for several applications — functional behaviour. Termination, completeness and correctness of model transformations have been studied already in [5, 2, 6, 3]. Functional behaviour of model transformations based on triple graph grammars has been analyzed for triple rules without application conditions in [13] using forward translation rules, which are derived from forward rules by additional translation attributes for keeping track of the elements that have been translated so far.

The main aim of this paper is to extend the analysis techniques for functional behaviour in [13] to the case of triple rules with negative application conditions (NACs) and to improve the efficiency of analysis and execution of model transformations studied in [2, 3, 6, 13]. For this purpose, we distinguish between specification NACs and filter NACs. Specification NACs have been introduced already in [6, 3], where triple rules and corresponding derived source and forward rules have been extended by NACs in order to improve the modeling power. Exemplarily, we show that NACs improve the specification of the model transformation *CD2RDBM* from class diagrams to relational data base models presented in [5, 2]. Therefore, we extend the forward translation rules introduced in [13] by corresponding NACs and show that model transformations based on forward translation rules with NACs are equivalent to model transformations studied in [6, 3], such that main results concerning termination, correctness and completeness can be transferred to our new framework (see Thm. 1). In order to analyze functional behaviour we can use general results for local confluence of transformation systems with NACs in [16]. But in order to improve efficiency in the context of model transformations we introduce so-called filter NACs. They filter out several misleading branches considered in the standard analysis of local confluence using critical pairs. In our second main result (see Thm. 2) we show how to analyze functional behaviour of model transformations based on forward translation rules by analyzing critical pairs for forward translation rules with filter NACs. Moreover, we introduce a strong version of functional behaviour, including model transformation sequences. In our third main result (see Thm. 3) we characterize strong functional behaviour by the absence of "significant" critical pairs for the corresponding set of forward translation rules with filter NACs.

In Sec. 2 we introduce model transformations based on TGGs with specification NACs and show the first main result on termination, correctness, and completeness. In Sec. 3 we introduce forward translation rules with filter NACs and present our main results on functional and strong functional behaviour. Based on these main results we discuss in Sec. 4 efficiency aspects of analysis and execution. Related work and a conclusion are presented in Sections 5 and 6.

This technical report is an extended version of [12] and presents the full proofs.

2 Model Transformations based on Triple Graph Grammars with NACs

Triple graph grammars [17] 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 this section, we review main constructions and results of model transformations based on [18, 3, 13] and extend them to the case with NACs.

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 $m = (m_S, m_C, m_T) : G \to H$ between triple graphs G and 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 $type_G : G \to TG$.

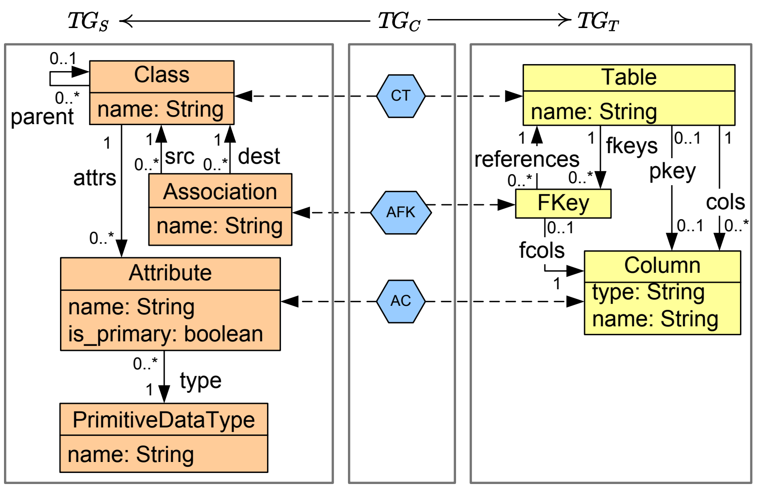

Figure 1: Triple type graph for *CD2RDBM*

Example 1. Triple Type Graph: *Fig. 1 shows the type graph TG of the triple graph grammar TGG for our example model transformation from class diagrams to database models. The source component TG_S defines the structure of class diagrams while in the 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 [5], 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. The denoted multiplicity constraints are ensured by the triple rules in Figs. 3 and 4.*

Note that the case study uses attributed triple graphs based on E-graphs as presented in [5] in the framework of weak adhesive HLR categories and we refer to [1] for more details on attributed graphs.

$$\begin{tikzcd}
L = (L_S \arrow[d, "tr"'] & L_C \arrow[l, "s_L"'] \arrow[r, "t_L"] \arrow[d, "tr_C"] & L_T) \arrow[d, "tr_T"] & L \arrow[r, hook, "tr"] \arrow[d, "m"'] \arrow[dr, phantom, "(PO)"] & R \arrow[d, "n"] \\
R = (R_S & R_C \arrow[l, "s_R"] \arrow[r, "t_R"'] & R_T) & G \arrow[r, hook, "t"'] & H
\end{tikzcd}$$

Figure 2: Triple rule and triple transformation step

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 (left of Fig. 2) 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 $G \xRightarrow{tr,m} H$ (right of Fig. 2) 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.

Example 2. Triple Rules: *The triple rules in Fig. 3 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 side 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 coloring. 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 via the rule* "Attr2Column"*.*

$$\begin{tikzcd}
(L_S \arrow[d, "tr_S"'] & \varnothing \arrow[l] \arrow[r] \arrow[d] & \varnothing) \arrow[d] \\
(R_S & \varnothing \arrow[l] \arrow[r] & \varnothing)
\end{tikzcd}
\qquad
\begin{tikzcd}
(\varnothing \arrow[d] & \varnothing \arrow[l] \arrow[r] \arrow[d] & L_T) \arrow[d, "tr_T"] \\
(\varnothing & \varnothing \arrow[l] \arrow[r] & R_T)
\end{tikzcd}
\qquad
\begin{tikzcd}
(R_S \arrow[d, "id"'] & L_C \arrow[l, "tr_S \circ s_L"'] \arrow[r, "t_L"] \arrow[d, "tr_C"] & L_T) \arrow[d, "tr_T"] \\
(R_S & R_C \arrow[l, "s_R"'] \arrow[r, "t_R"] & R_T)
\end{tikzcd}$$

source rule tr_S target rule tr_T 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 source rule tr_S for the construction resp. parsing of a model of the source language and a forward rule tr_F for forward

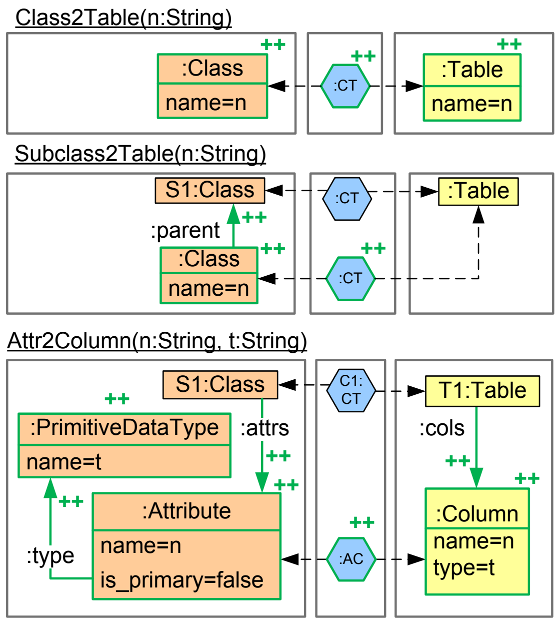

Figure 3: Rules for the model transformation *CD2RDBM*, Part 1

transformation sequences. 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 $VL_T \subseteq VL_{T0} = \{G_T \,|\, \varnothing \Rightarrow^* (\varnothing \leftarrow \varnothing \rightarrow G_T)\ via\ TR_T\}$.

According to [6, 3] we present negative application conditions for triple rules. In most case studies of model transformations source-target NACs are sufficient and we regard them as the standard case.

Definition 1. Triple Rules with Negative Application Conditions: *Given a triple rule $tr = (L \rightarrow R)$, a negative application condition (NAC) $(n : L \rightarrow N)$ consists of a triple graph N and a triple graph morphism n. A NAC with $n = (n^S, id_{L_C}, id_{L_T})$ is called* source NAC *and a NAC with $n = (id_{L_S}, id_{L_C}, n^T)$ is called* target NAC. *This means that source-target NACs, i.e. either source or target NACs, prohibit the existence of certain structures either in the source or in the target part only.*

A match $m : L \rightarrow G$ is NAC consistent if there is no injective $q : N \rightarrow G$ such that $q \circ n = m$ for each NAC $L \xrightarrow{n} N$. A triple transformation $G \overset{*}{\Rightarrow} H$ is NAC consistent if all matches are NAC consistent.

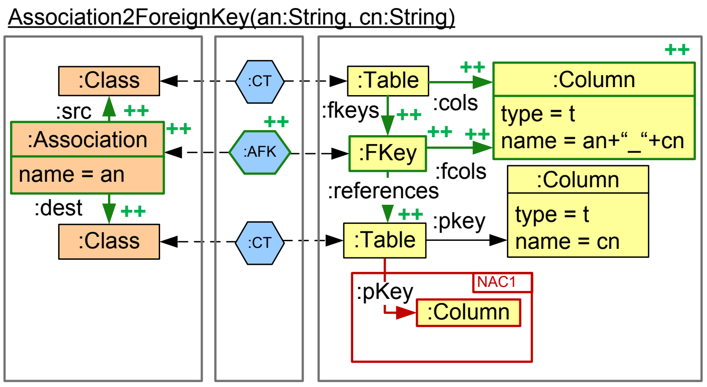

Figure 4: Rules for the model transformation *CD2RDBM*, Part 2

Example 3. Triple Rules with NACs: *Figure 4 and the upper part of Fig. 5 show the remaining triple rules for the model transformation "CD2RDBM". NACs are specified in short notation using the label "NAC" with a frame and red line colour within the frame. A complete NAC is obtained by composing the left hand side of a rule with the red marked elements within the NAC-frame. The rule "Association2ForeignKey" creates an association between two classes and the corresponding foreign key and the NAC ensures that there is only one primary key at the destination table. The parameters "an" and "cn" are used to set the names of the association and column nodes. The rule "PrimaryAttr2Column" extends "Attr2Column" by creating additionally a link of type "pkey" for the column and by setting "primary=true". Furthermore, there is a source and a target NAC, which ensure that there is no primary attribute nor column currently present.*

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 [1], where attributes are possible for nodes and edges. While in this paper the translation attributes are inserted in the source models they can be kept separate as an external pointer structure in order to keep the source model unchanged as shown in Sec. 5 of [11].

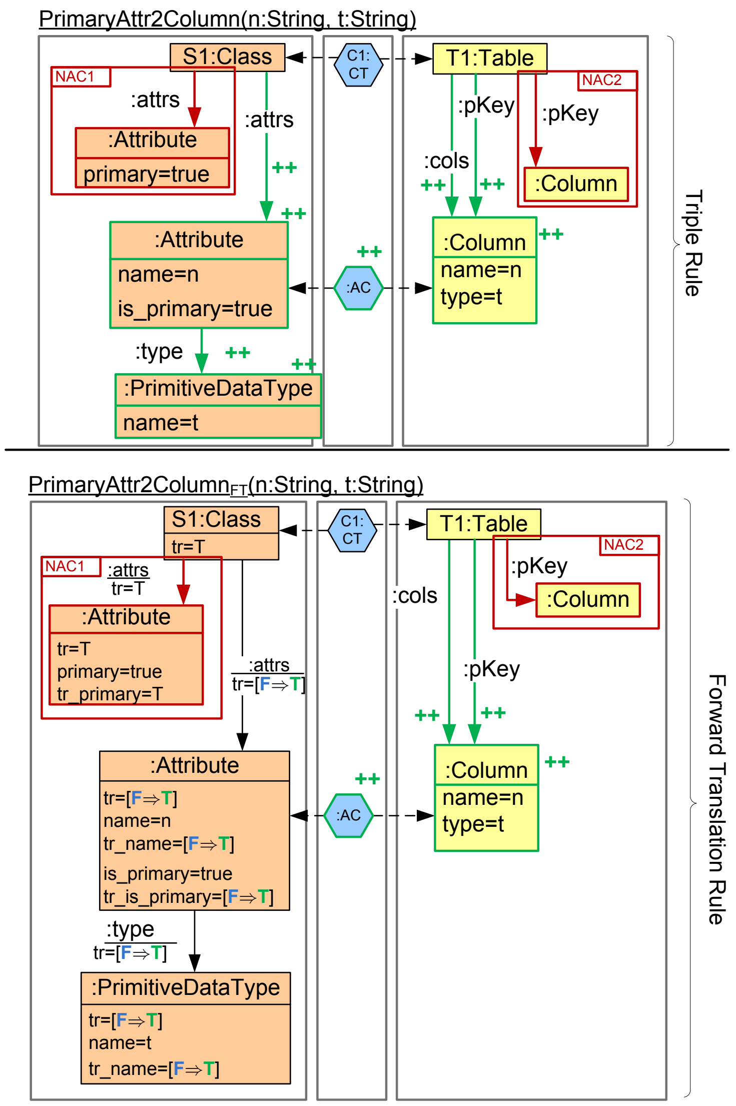

Figure 5: Rules for the model transformation *CD2RDBM*, Part 3

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* $X \in \{NA, EA\}$.

$$\begin{tikzcd} M \arrow[r, hook] \arrow[d, hook] \arrow[dr, phantom, "(PO)"] & Att_M \arrow[d] \\ {|G|} \arrow[r] & {|G'|} \end{tikzcd}$$

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 new concept of forward translation rules as introduced in [13] extends the construction of forward rules by additional translation attributes in the source component. The translation attributes keep track of the elements that have been translated so far, which ensures that each element in the source graph is not translated twice. The rules are deleting on the translation attributes and thus, the triple transformations are extended from a single (total) pushout to the classical double pushout (DPO) approach [1]. We call these rules forward translation rules, because pure forward rules need to be controlled by an additional control conditions, such as the source consistency condition in [5, 3].

Definition 4. Forward Translation Rules with NACs: *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:*

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

Moreover, for each NAC $n : L \to N$ of tr we define a forward translation NAC $n_{FT} : L_{FT} \to N_{FT}$ of tr_{FT} as inclusion with $N_{FT} = (L_{FT} +_L N) \oplus Att^{\mathbf{T}}_{N_S \setminus L_S}$.

Remark 1. *Note that $(L_{FT} +_L N)$ is the union of L_{FT} and N with shared L and for a target NAC n the forward translation NAC n_{FT} does not contain any translation attributes because $N_S = L_S$.*

Example 4. Forward Translation Rule with NACs: *Fig 5 shows in its lower part the forward translation rule with NACs "PrimaryAttr2Column$_{FT}$". According to Def. 4 the source elements of the triple rule "PrimaryAttr2Column" are extended by translation attributes and changed by the rule from "**F**" to "**T**", if the owning elements are created by the triple rule. Furthermore, the additional elements in the NAC are extended by translation attributes set to "**T**". Thus, the source NACs concern only elements that have been translated so far.*

From the application point of view model transformation rules should be applied along matches that are injective on the structural part. But it would be too restrictive to require injectivity of the matches also on the data and variable nodes, because we must allow that two different variables are mapped to the same data value. For this reason we use the notion of "*almost injective matches*" [13], 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. Next, we define model transformations based on forward translation rules based on complete forward translation sequences.

Definition 5. Completely Translated Graphs and Complete Sequences: *A forward translation sequence $G_0 \xRightarrow{tr^*_{FT}} G_n$ 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***").*

Definition 6. Model Transformation Based on Forward Translation Rules: *A* model transformation sequence $(G_S,\, G'_0 \xRightarrow{tr^*_{FT}} G'_n, G_T)$ *based on forward translation rules with NACs consists of a source graph G_S, a target graph G_T, and a complete* TGT*-sequence $G'_0 \xRightarrow{tr^*_{FT}} G'_n$ with almost injective matches, $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \to \varnothing)$ and $G'_n = (Att^{\mathbf{T}}(G_S) \leftarrow G_C \to G_T)$.*
A model transformation *$MT : VL_{S0} \Rrightarrow VL_{T0}$ based on forward translation rules with NACs is defined by all model transformation sequences as above with $G_S \in VL_{S0}$ and $G_T \in VL_{T0}$. All these pairs (G_S, G_T) define the* model transformation relation *$MTR \subseteq VL_{S0} \times VL_{T0}$. The model transformation is* terminating *if there are no infinite* TGT*-sequences via forward translation rules and almost injective matches starting with $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \to \varnothing)$ for some source graph G_S.*

Using Fact 1 below we are able to state our first main result in Thm. 1 concerning termination, correctness and completeness of model transformations.

Fact 1. Complete Forward Translation Sequences with NACs: *Given a triple graph grammar $TGG = (TG, \varnothing, TR)$ with NACs 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 and NAC-consistent* TGT*-sequence $G_0 \xRightarrow{tr_F^*} G$ via forward rules and $G = (G_S \leftarrow G_C \rightarrow G_T)$.*

2. *$\exists$ a complete NAC-consistent* TGT*-sequence $G'_0 \xRightarrow{tr_{FT}^*} G'$ via forward translation rules and $G' = (Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T)$.*

Proof. By Lemma 1 and Fact 1 in [13] we have that each single step as well as the complete sequences are equivalent disregarding the NACs. Thus, we show that the single steps are equivalent and NAC consistent, the equivalence of the complete sequences follows directly.

For each step, we have transformations $G_{i-1,0} \xRightarrow{tr_{i,S}, m_{i,S}} G_{i,0}$, $G_{i-1} \xRightarrow{tr_{i,F}, m_{i,F}} G_i$, $G'_{i-1} \xRightarrow{tr_{i,FT}, m_{i,FT}} G'_i$ with $G'_{i-1} = G_{i-1} \oplus Att^{\mathbf{F}}_{G_0 \backslash G_{i-1,0}} \oplus Att^{\mathbf{T}}_{G_{i-1,0}}$, $G'_i = G_i \oplus Att^{\mathbf{F}}_{G_0 \backslash G_{i,0}} \oplus Att^{\mathbf{T}}_{G_{i,0}}$, and $m_{i,FT}|_{L_{i,F}} = m_{i,F}$.

For a target NAC $n : L_i \rightarrow N$, we have to show that $m_{i,F} \models n$ iff $m_{i,FT} \models n_{FT}$, the corresponding forward translation NAC. If $m_{i,FT} \not\models n_{F_T}$, we find a monomorphism q' with $q' \circ n_{FT} = m_{i,FT}$. Since $n = n_{FT}|_N$, define $q = q'|_N$ and it follows that $q \circ n = m_{i,F}$, i.e. $m_{i,F} \not\models n$. Vice versa, if $m_{i,F} \not\models n$, we find a monomorphism q with $q \circ n = m_{i,F}$. Since $N_S = L_{i,S}$, we do not have any additional translation attributes in N_{FT}. Thus $m_{i,FT}$ can be extended by q to $q' : N_{FT} \rightarrow G'_{i-1}$ such that $m_{i,FT} \not\models n_{FT}$.

Similarly, we have to show that for a source NAC $n : L \rightarrow N$, $m_{i,S} \models n$ iff $m_{i,FT} \models n_{FT}$. As for target NACs, if $m_{i,FT} \not\models n_{F_T}$, we find a monomorphism q' with $q' \circ n_{FT} = m_{i,FT}$ and for the restriction to $L_{i,S}$ and N it follows that $q \circ n = m_{i,S}$, i.e. $m_{i,S} \not\models n$. Vice versa, if $m_{i,S} \not\models n$, we find a monomorphism q with $q \circ n = m_{i,S}$. Now define q' with $q'(x) = m_{i,FT}(x)$ for $x \in L_{FT}$, $q'(x) = q(x)$ for $x \in N \backslash L_i$, and for each $x \in N_S \backslash L_{i,S}$ we have that $q(x) \in G_{i-1,0}$. From the above characterization of G'_{i-1} it follows that the corresponding translation attributes tr_x and tr_x_a are set to $\mathbf{T}$ in G'_{i-1}. Thus, q' is well-defined and $q' \circ n_{FT} = m_{i,FT}$, i.e. $m_{i,FT} \not\models n_{FT}$. □

Theorem 1. 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 from "$\mathbf{F}$" to "$\mathbf{T}$",*

- correct*, i.e. for each model transformation sequence $(G_S, G_0 \xRightarrow{tr_{FT}^*} G_n, G_T)$ 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 $(G_S, G_0 \xRightarrow{tr_{FT}^*} G_n, G_T)$.*

Proof. Termination: Let TR_{FT} be a finite set of forward translation rules, such that each rule changes at least one translation attribute. By Def. 4 we have that a forward translation rule does not change the structure of the source component, but only the translation attributes. These attributes are modified exclusively from "**F**" to "**T**" and the number of translation attributes remains the same. For a model G_S of the source language VL_S that is finite on the graph part, i.e. attributed with an algebra with possibly infinite data sets, we have that also the model $G_0 = (Att^{\mathbf{T}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$ is finite on the graph part. Thus there are finitely many translation attributes set to "**F**" and each application of a forward translation reduces the amount of translation attributes set to "**F**". This implies that any sequence of transformation steps via TR_{FT} is finite. Furthermore, since the rule components are finite on the graph part also the models remain finite on the graph part ensuring that at each step there are finitely many matches. Thus, the model transformation terminates.

The correctness and completeness properties follow directly using Fact 1 and Thms. 2 and 3 in [6] as well as Thm. 2 in [13]. □

Applying a rule according to the DPO approach involves the check of the gluing condition in general. However, in the case of forward translation rules and almost injective matches we have that the gluing condition is always satisfied. This means that the condition does not have to be checked, which simplifies the analysis of functional behaviour in Sec. 3.

Fact 2. Gluing Condition for Forward Translation Rules: *Let tr_{FT} be a forward translation rule and $m_{FT} : L_{FT} \rightarrow G$ be an almost injective match, then the gluing condition is satisfied, i.e. there is the transformation step $G \xRightarrow{tr_{FT}, m_{FT}} H$.*

Proof. According to Def. 9.8 in [1] we need to check that $DP \cup IP \subseteq GP$. First of all, the set IP may only contain data elements by the restriction of the match, which are in GP. Furthermore, the set DP does only contain nodes. The rule is only deleting on attribution edges and thus, $DP \cup IP \subseteq GP$. □

3 Analysis of Functional Behaviour

Functional behaviour of a model transformation means that each model of the source language $\mathcal{L}_S \subseteq VL_S$ is transformed into a unique model of the target language. This section presents new techniques especially developed to show functional behaviour of correct and complete model transformations based on TGGs.

Definition 7. Functional Behaviour of Model Transformations: *A model transformation MT based on forward translation rules has* functional behaviour *if each execution of MT starting at a source model G_S of the source language $\mathcal{L}_S \subseteq VL_S$ leads to a unique target model $G_T \in VL_T$. The execution of MT requires backtracking, if there are terminating TGT-sequences $(Att^F(G_S) \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{tr^*_{FT}} G'_n$ with $G'^S_n \neq Att^T(G_S)$.*

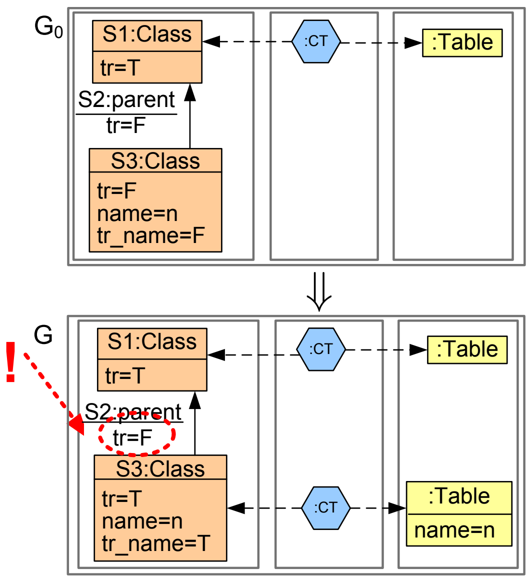

Figure 6: Step $G_0 \xRightarrow{Class2Table_{FT}} G$ with misleading graph G

The standard way to analyze functional behaviour is to check whether the underlying transformation system is confluent, i.e. all diverging derivation paths starting at the same model finally meet again. In the context of model transformations, confluence only needs to be ensured for transformation paths which lead to completely translated models. For this reason, we introduce so-called *filter NACs* that extend the model transformation rules in order to avoid misleading paths that cause backtracking. The overall behaviour w.r.t. the model transformation relation is preserved. Filter NACs are based on the following notion of misleading graphs, which can be seen as model fragments that are responsible for the backtracking of a model transformation.

Definition 8. Translatable and Misleading Graphs: *A triple graph with translation attributes G is* translatable *if there is a transformation $G \overset{*}{\Rightarrow} H$ such that H is completely translated. A triple graph with translation attributes G is* misleading, *if every triple graph G' with translation attributes and $G' \supseteq G$ is not translatable.*

Example 5. Misleading Graph: *Consider the transformation step shown in Fig. 6. The resulting graph G is misleading according to Def. 8, because the edge $S2$ is labeled with a translation attribute set to "**F**", but there is no rule which may change this attribute in any bigger context at any later stage of the transformation. The only rule which changes the translation attribute of a "parent"-edge is "Subclass2Table$_{FT}$", but it requires that the source node $S3$ is labeled with a translation attribute set to "**F**". However, forward translation rules do not modify translation attributes if they are set to "**T**" already and additionally do not change the structure of the source component.*

Definition 9. Filter NAC: *A filter NAC n for a forward translation rule $tr_{FT} : L_{FT} \rightarrow R_{FT}$ is given by a morphism $n : L_{FT} \rightarrow N$, such that there is a* TGT *step $N \xRightarrow{tr_{FT},n} M$ with M being misleading. The extension of tr_{FT} by some set of filter NACs is called forward translation rule tr_{FN} with filter NACs.*

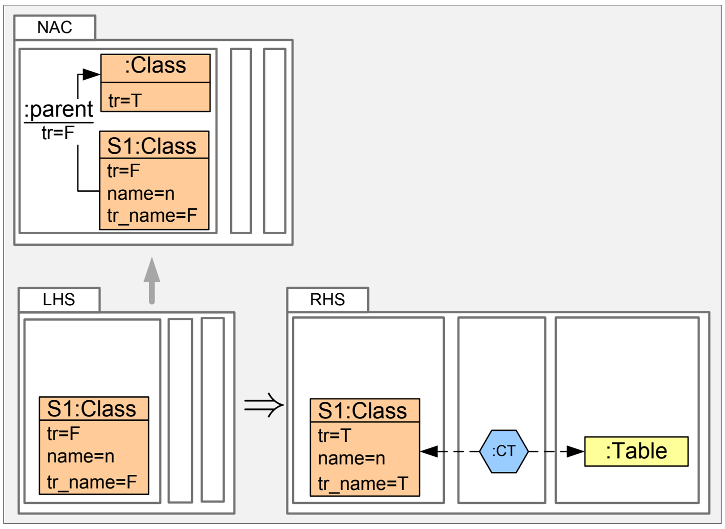

Figure 7: A forward translation rule with filter NAC: *Class2Table*$_{FN}$

Example 6. Forward Translation Rule with Filter NACs: *The rule Class2Table$_{FT}$ is extended by a filter NAC in Fig. 7, which is obtained from the graph G_0 of the transformation step $G_0 \xRightarrow{Class2Table_{FT}} G$ in Fig. 6, where G is misleading according to Ex. 5.*

A direct construction of filter NACs according to Def. 9 would be inefficient, because the size of the considered graphs to be checked is unbounded. For this reason we now present efficient techniques which support the generation of filter NACs and we can bound the size without losing generality. At first we present a static technique for a subset of filter NACs and thereafter, a dynamic generation technique leading to a much larger set of filter NACs. The first procedure in Fact 3 below is based on a sufficient criteria for checking the misleading property. Concerning our example this static generation leads to the filter NAC shown in Fig. 7 for the rule *Class2Table*$_{FT}$ for an incoming edge of type "*parent*".

Fact 3. Static Generation of Filter NACs: *Given a triple graph grammar, then the following procedure applied to each triple rule $tr \in TR$ generates filter NACs for the derived forward translation rules TR_{FT} leading to forward translation rules TR_{FN} with filter NACs:*

- *Outgoing Edges: Check whether the following properties hold*
  - *tr creates a node $(x : T_x)$ in the source component and the type graph allows outgoing edges of type "T_e" for nodes of type "T_x", but tr does not create an edge $(e : T_e)$ with source node x.*

  - *Each rule in TR which creates an edge $(e : T_e)$ also creates its source node.*
  - *Extend L_{FT} to N by adding an outgoing edge $(e : T_e)$ at x together with a target node. Add a translation attribute for e with value* **F**. *The inclusion $n : L_{FT} \to N$ is a NAC-consistent match for tr.*

  For each node x of tr fulfilling the above conditions, the filter NAC $(n : L_{FT} \to N)$ is generated for tr_{FT} leading to tr_{FN}.

- *Incoming Edges: Dual case, this time for an incoming edge $(e : T_e)$.*
- *TR_{FN} is the extension of TR_{FT} by all filter NACs constructed above.*

Proof. Consider a generated NAC $(n : L_{FT} \to N)$ for a node x in tr with an outgoing edge e in $N \setminus L$. A transformation step $N \xRightarrow{tr_{FT},n} M$ exists according to Fact 2 and leads to a graph M, where the edge e is still labeled with a translation attribute set to "**F**", but x is labeled with "**T**", because it is matched by the rule. Now, consider a graph $H' \supseteq M$, such that H' is a graph with translation attributes over a graph without translation attributes H, i.e. $H' = H \oplus Att_{H_0}$ for $H_0 \subseteq H'$ meaning that H' has at most one translation attributes for each element in H without translation attributes.

In order to have that M is misleading (Def. 8), it remains to show that H' is not translatable. Forward translation rules only modify translation attributes from "**F**" to "**T**", they do not increase the amount of translation attributes of a graph and no structural element is deleted. Thus, each graph H_i in a TGT sequence $H' \xRightarrow{tr^*_{FT}} \overline{H}_n$ will contain the edge e labeled with "**F**", because the rules, which modify the translation attribute of e are not applicable due to x being labeled with "**T**" in each graph $\overline{H}_i$ in the sequence and there is only one translation attribute for x in H'. Thus, each $\overline{H}_n$ is not completely translated and therefore, M is misleading. This means that $(n : L_{FT} \to N)$ is a filter NAC of tr_{FT}. Dualizing the proof leads to the result for a generated NAC w.r.t. an incoming edge. □

The following dynamic technique for deriving relevant filter NACs is based on the generation of critical pairs, which define conflicts of rule applications in a minimal context. By the completeness of critical pairs (Lemma 6.22 in [1]) we know that for each pair of two parallel dependent transformation steps there is a critical pair which can be embedded. For this reason, the generation of critical pairs can be used to derive filter NACs. A critical pair either directly specifies a filter NAC or a conflict that may lead to non-functional behaviour of the model transformation.

For the dynamic generation of filter NACs we use the tool AGG [21] for the generation of critical pairs for a plain graph transformation system. For this purpose, we first perform the flattening construction for triple graph grammars presented in [2, 13] extended to NACs using the flattening construction for morphisms. A critical pair $P_1 \xLeftarrow{tr_{1,FT}} K \xRightarrow{tr_{2,FT}} P_2$ consists of a pair of parallel dependent transformation steps. If a critical pair contains a misleading graph P_1 we can use the overlapping graph K as a filter NAC of the rule $tr_{1,FT}$. However, checking the misleading property needs human assistance, such that the generated critical pairs can be seen as filter NAC candidates. But we are currently working on a technique that uses a sufficient criteria to check the misleading property automatically and we are confident that this approach will provide a powerful generation technique.

Fact 4. Dynamic Generation of Filter NACs: *Given a set of forward translation rules, then generate the set of critical pairs* $P_1 \xLeftarrow{tr_{1,FT},m_1} K \xRightarrow{tr_{2,FT},m_2} P_2$. *If* P_1 *(or similarly* P_2*) is misleading, we generate a new filter NAC* $m_1 : L_{1,FT} \to K$ *for* $tr_{1,FT}$ *leading to* $tr_{1,FN}$*, such that* $K \xRightarrow{tr_{1,FN}} P_1$ *violates the filter NAC. Hence, the critical pair for* $tr_{1,FT}$ *and* $tr_{2,FT}$ *is no longer a critical pair for for* $tr_{1,FN}$ *and* $tr_{2,FT}$*. But this construction may lead to new critical pairs for the forward translation rules with filter NACs. The procedure is repeated until no further filter NAC can be found or validated. This construction starting with* TR_{FT} *always terminates, if the structural part of each graph of a rule is finite.*

Proof. The constructed NACs are filter NACs, because the transformation step $K \xRightarrow{tr_{1,FT},m_1} P_1$ contains the misleading graph P_1. The procedure terminates, because the critical pairs are bounded by the amount of possible pairwise overlappings of the left hand sides of the rules. The amount of overlappings can be bounded by considering only constants and variables as possible attribute values. □

For our case study the dynamic generation terminates already after the second round, which is typical for practical applications, because the amount of already translated elements in the new critical pairs usually decreases. Furthermore, the amount of NACs can be reduced by combining similar NACs differing only on some translation attributes. The remaining critical pairs that do not specify filter NACs show effective conflicts between transformation rules and they can be provided to the developer of the model transformation to support the design phase.

The filter NACs introduced in this paper on the one hand support the analysis of functional behaviour and on the other hand, they also improve the efficiency of the execution. By definition, the occurrence of a filter NAC at an intermediate model means that the application of the owning rule would lead to a model that cannot be translated completely, i.e. the execution of the model transformation would perform backtracking at a later step. This way, a filter NAC *cuts off possible backtracking paths* of the model transformation. As presented in Fact 3 some filter NACs can be generated automatically and using Fact 4 a larger set of them can be obtained based on the generation of critical pairs. Finally, by Thms. 2 and 3 we can completely avoid backtracking if TR_{FN} has no significant critical pair or alternatively, if all critical pairs are strictly confluent.

As shown by Fact 5 below, filter NACs do not change the behaviour of model transformations. The only effect is that they filter out derivation paths, which would lead to misleading graphs, i.e. to backtracking for the computation of the model transformation sequence. This means that the filter NACs filter out backtracking paths. This equivalence is used on the one hand for the analysis of functional behaviour in Thms. 2 and 3 and furthermore, for improving the efficiency of the execution of model transformations as explained in Sec. 4.

Fact 5. Equivalence of Transformations with Filter NACs: *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 complete* TGT*-sequence $G'_0 \xRightarrow{tr^*_{FT}, m^*_{FT}} G'$ via forward translation rules.*
2. *$\exists$ a complete* TGT*-sequence $G'_0 \xRightarrow{tr^*_{FN}, m^*_{FT}} G'$ via forward translation rules with filter NACs.*

Proof. Sequence 1 consists of the same derivation diagrams as Sequence 2. We need to show that NAC-consistency is implied for both directions.

Direction $\Leftarrow$: Sequence 1 is NAC-consistent, because sequence 2 is NAC-consistent and each step in Sequence 2 involves a superset of the NACs for the corresponding step in Sequence 1.

Direction $\Rightarrow$: Consider a step $G_{i-1} \xRightarrow{tr_{(i,FT)}, \mathrm{m}_{(i,FT)}} G_i$, which leads to the step $G_{i-1} \xRightarrow{tr_{(i,FN)}, \mathrm{m}_{(i,FT)}} G_i$ if NACs are not considered. Assume that m_{FT} does not fulfill some NAC of tr_{FN}. This implies that a filter NAC $(n : L_{i,FT} \rightarrow N)$ is not fulfilled, because all other NACs are fulfilled by NAC-consistency of Sequence 1. Thus, there is a triple morphism $q : N \rightarrow G_{i-1}$ with $q \circ n = m_{i,FT}$. By Thm. 6.18 (Restriction Thm.) in [1] we have that the transformation step $G_{i-1} \xRightarrow{tr_{(i,FN)}, \mathrm{m}_{(i,FT)}} G_i$ can be restricted to $N \xRightarrow{tr_{(i,FT)}, n} H$ with embedding $H \rightarrow G_i$. By Def. 9 of filter NACs we know that $N \xRightarrow{tr_{(i,FT)}, n} H$ and H is misleading, which implies by Def. 8 that G_i is not translatable. But Sequence 1 leads to a completely translated graph G_n and we have $G_1 \Rightarrow^* G_n$. This is a contradiction and we have that the filter NAC is fulfilled. Therefore, each transformation step is NAC-consistent. □

Theorem 2. Functional Behaviour: *Let MTbe a model transformation based on forward translation rules TR_{FT} and let TR_{FN} extend TR_{FT} with filter NACs such that TR_{FN} is terminating and all critical pairs are strictly confluent. Then, MT has functional behaviour. Moreover, the model transformation MT' based on TR_{FN} does not require backtracking and defines the same model transformation relation, i.e. $MTR' = MTR$.*

Remark 2. *TR_{FN} is terminating, if TR_{FT} is terminating and a sufficient condition is given in Thm. 1. Termination of TR_{FN} with strict confluence of critical pairs implies unique normal forms by the Local Confluence Theorem in [16].*

Proof. For functional behaviour of the model transformation we have to show that each source model $G_S \in VL_S$ is transformed into a unique (up to isomorphism) completely translated target model G_T, which means that there is a completely translated triple model G' with $G'_T = G_T$, and furthermore $G_T \in VL_T$.

For $G_S \in VL_S$ we have by definition of VL that there is a $G_T \in VL_T$ and a TGT-sequence $\varnothing \xRightarrow{tr^*} (G_S \leftarrow G_C \rightarrow G_T)$ via TR and using the decomposition theorem with

NACs in [6] we obtain a match consistent TGT-sequence $\varnothing \xRightarrow{tr_S^*} (G_S \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{tr_F^*}$ $(G_S \leftarrow G_C \rightarrow G_T)$ and by Fact 1 in [13] together with Fact 1 above a complete TGT-sequence $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{tr_{FT}^*} (Att^{\mathbf{T}}(G_S) \leftarrow G_C \rightarrow G_T) = G'$.

This means that $(G_S, G'_0 \xRightarrow{tr_{FT}^*} G', G_T)$ is a model transformation sequence based on TR_{FT}. Assume that we also have a completely translated TGT-sequence $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{\overline{tr_{FT}^*}} (Att^{\mathbf{T}}(G_S) \leftarrow \overline{G_C} \rightarrow \overline{G_T}) = \overline{G}'$. By Fact. 5 we also have the complete TGT-sequences $G'_0 \xRightarrow{tr_{FN}^*} G'$ and $G'_0 \xRightarrow{tr_{FN}^*} \overline{G}'$. Using the precondition that TR_{FN} is terminating and all critical pairs are strictly confluent we can conclude by the Local Confluence Theorem in [16] that $G' \cong \overline{G}'$ (see Remark 2.2) and hence, $G_T \cong \overline{G}_T$.

Backtracking is not required, because termination of TR_{FN} with strict confluence of critical pairs implies unique normal forms by the Local Confluence Theorem in [16]. By local confluence we have that there any terminating TGT-sequences $(Att^F(G_S) \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{tr_{FN}^*} G'_n$ leads to a unique G'_n up to isomorphism and by correctness and completeness (Thm. 1) we have that $G'^S_n = Att^T(G_S)$.

The model transformation relation is the same, because we have by Fact 5 the equivalence of the model transformation sequences of MT and MT'.

$\square$

If the set of generated critical pairs of a system of forward translation rules with filter NACs TR_{FN} is empty, we can directly conclude from Thm. 2 that the corresponding system with forward translation rules TR_{FT} has functional behaviour. From an efficiency point of view, model transformations should be based on a compact set of rules, because large rule sets usually involve more attempts of matching until finding a valid one at a concrete step. In the optimal case, the rule set is minimal in the sense that each transformation sequence of the model transformation is itself unique up to switch equivalence. For this reason, we introduce the notion of strong functional behaviour.

Definition 10. Strong Functional Behaviour of Model Transformations: *A model transformation based on forward translation rules TR_{FN} with filter NACs has* strong functional behaviour *if for each $G_S \in \mathcal{L}_S \subseteq VL_S$ there is a $G_T \in VL_T$ and a model transformation sequence $(G_S, G_0 \xRightarrow{tr_{FN}^*} G_n, G_T)$ and each two terminating TGT-sequences $G'_0 \xRightarrow{tr_{FN}^*} G'_n$ and $G'_0 \xRightarrow{\overline{tr_{FN}^*}} \overline{G}'_m$ are switch-equivalent up to isomorphism.*

Remark 3. *1. The sequences are terminating means that no rule in TR_{FN} is applicable any more, but it is not required that the sequences are complete, i.e. that G'_n and $\overline{G}'_m$ are completely translated.*

2. Strong functional behaviour implies functional behaviour, because G'_n and $\overline{G}'_m$ completely translated implies that $G'_0 \xRightarrow{tr_{FN}^*} G'_n$ and $G'_0 \xRightarrow{\overline{tr_{FN}^*}} \overline{G}'_m$ are terminating TGT-sequences.

3. Two sequences $t1 : G_0 \Rightarrow^* G_1$ and $t2 : G_0 \Rightarrow^* G_2$ are called switch-equivalent, written $t1 \approx t2$, if $G_1 = G_2$ and $t2$ can be obtained from $t1$ by switching sequential *independent steps according to the Local Church Rosser Theorem with NACs [16]. The sequences $t1$ and $t2$ are called switch-equivalent up to isomorphism if $t1 : G_0 \Rightarrow^* G_1$ has an isomorphic sequence $t1' : G_0 \to G_2$ (using the same sequence of rules) with $i : G_1 \xrightarrow{\sim} G_2$, written $t1' = i \circ t1$, such that $t1' \approx t2$. This means especially that the rule sequence in $t2$ is a permutation of that in $t1$.*

The third main result of this paper shows that strong functional behaviour of model transformations based on forward translation rules with filter NACs can be completely characterized by the absence of "significant" critical pairs.

Definition 11. Significant Critical Pair: *A critical pair $P_1 \overset{tr_{1,FN}}{\Longleftarrow} K \overset{tr_{2,FN}}{\Longrightarrow} P_2$ for TR_{FN} is called significant, if it can be embedded into a parallel dependent pair $G'_1 \overset{tr_{1,FN}}{\Longleftarrow} G' \overset{tr_{2,FN}}{\Longrightarrow} G'_2$ such that there is $G_S \in VL_S$ and $G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'$ with $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \to \varnothing)$.*

$$\begin{tikzcd} & & G1' \\ G'_0 \arrow[r, Rightarrow, "*"] & G' \arrow[ur, Rightarrow, "tr_{1,FN}"] \arrow[dr, Rightarrow, "tr_{2,FN}"'] & \\ & & G'_2 \end{tikzcd}$$

Theorem 3. Strong Functional Behaviour: *A model transformation based on terminating forward translation rules TR_{FN} with filter NACs has strong functional behaviour and does not require backtracking iff TR_{FN} has no significant critical pair.*

Proof.

Direction "$\Leftarrow$": Assume that TR_{FN} has no significant critical pair. Similar to the proof of Thm. 2 we obtain for each $G_S \in VL_S$ a $G_T \in VL_T$ and a complete TGT-sequence $G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'$ and a model transformation $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G', G_T)$ based on TR_{FT} underlying TR_{FN}. By Lem. 5 above we also have a complete TGT-sequence $G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'$ and hence, also a model transformation $(G_S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G', G_T)$ based on TR_{FT} underlying TR_{FN}. In order to show strong functional behaviour let $G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'_n$ and $G'_0 \overset{\overline{tr}^*_{FN}}{\Longrightarrow} \overline{G}'_m$ be two terminating TGT-sequences with $m, n \geq 1$. We have to show that they are switch-equivalent up to isomorphism. We show by induction on the combined length $n + m$ that both sequences can be extended to switch-equivalent sequences.

For $n+m = 2$ we have $n = m = 1$ with $t1 : G'_0 \overset{tr_{FN},m}{\Longrightarrow} G'_1$ and $\overline{t1} : G'_0 \overset{\overline{tr}_{FN},\overline{m}}{\Longrightarrow} \overline{G}'_1$. If $tr_{FN} = \overline{tr}_{FN}$ and $m = \overline{m}$, then both are isomorphic with isomorphism $i : \overline{G}'_1 \xrightarrow{\sim} G'_1$, such that $t1 \approx i \circ \overline{t1}$. If not, then $t1$ and $\overline{t1}$ are parallel independent, because otherwise we would have a significant critical pair by completeness of critical pairs in [16]. By the Local Church Rosser Theorem [16] we have $t2 : G'_1 \overset{\overline{tr}_{FN}}{\Longrightarrow} G'_2$ and $\overline{t2} : \overline{G}'_1 \overset{tr_{FN}}{\Longrightarrow} G'_2$, such that $t2 \circ t1 \approx \overline{t2} \circ \overline{t1} : G'_0 \Rightarrow^* G'_2$.

Now assume that for $t1 : G'_0 \Rightarrow^* G'_{n-1}$ and $\overline{t1} : G'_0 \Rightarrow^* \overline{G}'_m$ we have extensions $t2 : G'_{n-1} \Rightarrow^* H$, $\overline{t2} : \overline{G}'_m \Rightarrow^* H$, such that $t2 \circ t1 \approx \overline{t2} \circ \overline{t1}$.

$$\begin{tikzcd}
G'_0 \arrow[r, Rightarrow, "t1"] \arrow[d, Rightarrow, "\overline{t1}"', "*"] & {}^{*}\, G'_{n-1} \arrow[r, Rightarrow, "t"] \arrow[d, Rightarrow, "t2", "*"'] & G'_n \arrow[d, Rightarrow, "t3", "*"'] \\
\overline{G}'_m \arrow[r, Rightarrow, "\overline{t2}"', "*"] & H \arrow[r, Rightarrow, "\overline{t3}"', "*"] & K
\end{tikzcd}$$

Now consider a step $t : G'_{n-1} \Rightarrow G'_n$, then we have to show that $t \circ t1$ and $\overline{t1}$ can be extended to switch-equivalent sequences. By induction hypothesis and definition of significant critical pairs also t and $t2$ can be extended by $t3 : G'_n \Rightarrow^* K$, $\overline{t3} : H \Rightarrow^* K$, such that $t3 \circ t \approx \overline{t3} \circ t2$. Now, composition closure of switch equivalence implies $t3 \circ t \circ t1 \approx \overline{t3} \circ \overline{t2} \circ \overline{t1} : G'_0 \Rightarrow^* K$. This completes the induction proof.

Now, we use that G'_n and $\overline{G}'_m$ are both terminal which implies that $t3$ and $\overline{t3} \circ \overline{t2}$ must be isomorphisms. This shows that $G'_0 \xRightarrow{tr^*_{FN}} G'_n$ and $G'_0 \xRightarrow{\overline{tr}^*_{FN}} \overline{G}'_m$ are switch-equivalent up to isomorphism.

Direction "⇒": Assume now that TR_{FN} has strong functional behaviour and that TR_{FN} has a significant critical pair. We have to show a contradiction in this case.

Let $P_1 \xLeftarrow{tr_{1,FN}} K \xRightarrow{tr_{2,FN}} P_2$ be the significant critical pair which can be embedded into a parallel dependent pair $G_1 \xLeftarrow{tr_{1,FN}} G' \xRightarrow{tr_{2,FN}} G_2$, such that there is $G_S \in VL_S$ with $G'_0 \xRightarrow{tr^*_{FN}} G'$ and $G'_0 = (Att^{\mathbf{F}}(G_S) \leftarrow \varnothing \rightarrow \varnothing)$. Since TR_{FN} is terminating we have terminating sequences $G_1 \Rightarrow^* G_{1n}$ via TR_{FN} and $G_2 \Rightarrow^* G_{2m}$ via TR_{FN}. By composition we have the following terminating TGT-sequences

1. $G'_0 \xRightarrow{tr_{FN}} G' \xRightarrow{tr_{1,FN}} G_1 \Rightarrow^* G_{1n}$
2. $G'_0 \xRightarrow{tr_{FN}} G' \xRightarrow{tr_{2,FN}} G_2 \Rightarrow^* G_{2m}$

Since TR_{FN} has strong functional behaviour both are switch-equivalent up to isomorphism. For simplicity assume $G_{1n} = G_{2m}$ instead of $G_{1n} \cong G_{2m}$. This implies $n = m$ and $G' \xRightarrow{tr_{1,FN}} G_1 \Rightarrow^* G_{1n}$ switch-equivalent to $G' \xRightarrow{tr_{2,FN}} G_2 \Rightarrow^* G_{1n}$. This means $tr_{2,FN}$ occurs in $G_1 \Rightarrow^* G_{1n}$ and can be shifted in $G' \xRightarrow{tr_{1,FN}} G_1 \Rightarrow^* G_{1n}$, such that we obtain $G' \xRightarrow{tr_{2,FN}} G_2 \Rightarrow^* G_{1n}$.

But this implies that in an intermediate step we can apply the parallel rule $tr_{1,FN} + tr_{2,FN}$ leading to parallel independence of $G' \xRightarrow{tr_{1,FN}} G_1$ and $G' \xRightarrow{tr_{2,FN}} G_2$, which is a contradiction. Hence, TR_{FN} has no significant critical pair.

It remains to show that strong functional behaviour implies that backtracking is not required. We have that TR_{FN} is terminating. Assume there is a terminating TGT-sequences $(Att^F(G_S) \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{tr^*_{FT}} G'_n$ with $G'^S_n \neq Att^T(G_S)$. By correctness and completeness (Thm. 1) we have that there is a further terminating sequence $(Att^F(G_S) \leftarrow \varnothing \rightarrow \varnothing) \xRightarrow{tr^*_{FT}} G''_n$ with $G''^S_n = Att^T(G_S)$. By strong functional behaviour we have that both sequences are switch-equivalent up to isomorphism and hence, $G'^S_n = G''^S_n$, which is a contradiction and we have that backtracking is not required. □

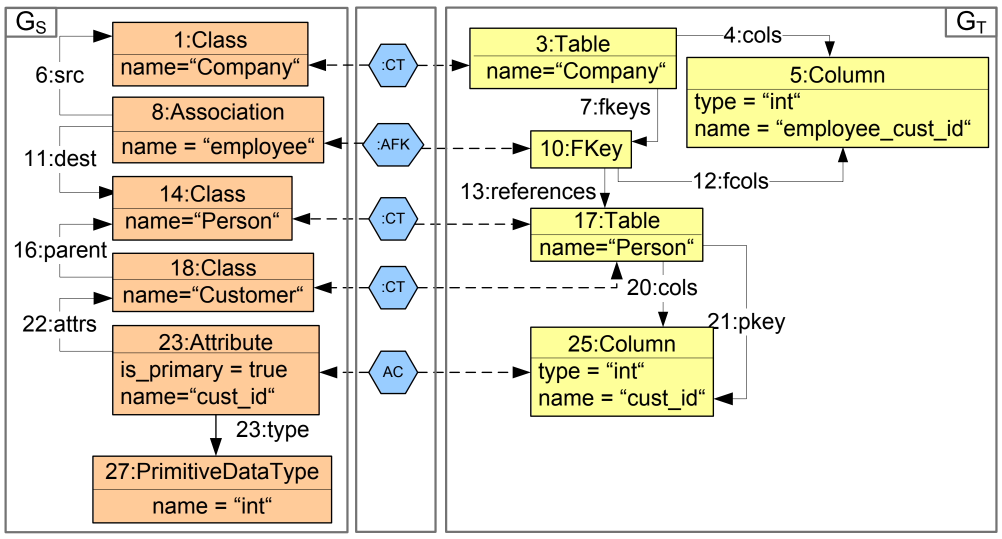

Figure 8: Triple graph instance

Example 7. Functional Behaviour: *We analyze functional behaviour of the model transformation CD2RDBM with triple rulesTR given in Figs. 3 and 4. First of all, CD2RDBM is terminating according to Thm. 1. For analyzing the local confluence we can use the tool AGG [21] for the generation of critical pairs. We use the extended rule Class2Table*$_{FN}$ *as shown in Fig. 7 and extend it by a further filter NAC obtained by the static generation acc. to Fact 3. AGG detects two critical pairs showing a conflict of the rule "PrimaryAttr2Column" with itself for an overlapping graph with two primary attributes. Both critical pairs lead to additional filter NACs by the dynamic generation of filter NACs in Fact 4 leading to a system of forward translation rules with filter NACs without any critical pair. Thus, we can apply Thm. 3 and show that the model transformation based on the forward translation rules with filter NACs* TR_{FN} *has* strong functional behaviour *and does not require backtracking. Furthermore, by Thm. 2 we can conclude that the model transformation based on the forward translation rules* TR_{FT} *without filter NACs has* functional behaviour *and does not require backtracking. As an example, Fig. 8 shows the resulting triple graph (translation attributes are omitted) of a model transformation starting with the class diagram* G_S.

4 Efficient Analysis and Execution

Our approach to model transformations based on triple graph grammars (TGGs) with NACs will be discussed now with respect to the efficiency for both, analysis of properties and execution.

Correctness and Completeness: As shown by Thm. 1 based on [6, 3] model transformations based on TGGs with NACs are correct and complete with respect to the language of integrated models *VL* generated by the triple rules. Thus, correctness and completeness are ensured by construction.

Termination: As presented in [3] termination is essentially ensured, if all triple rules are creating on the source component. This property can be checked statically, automatically and efficiently by checking $(R_S \setminus L_S) \neq \emptyset$. In Thm. 1 we have given an explicit condition for the forward translation rules to be terminating.

Functional Behaviour: The new concept of filter NACs introduced in this paper provides a powerful basis for reducing the analysis efforts w.r.t. functional behaviour. Once termination is shown as explained above, functional behaviour of model transformations based on forward translation rules TR_{FT} can be checked by generating the critical pairs of the transformation system with AGG [21] and showing strict confluence. The static and dynamic generation of filter NACs (Facts 3 and 4) allows to eliminate critical pairs. In the best case, all critical pairs disappear showing the functional behaviour of the model transformation immediately. The new notion of strong functional behaviour of a system based on transformation rules TR_{FN} with filter NACs is completely characterized by the absence of "significant" critical pairs, such that we can ensure for each source model that the transformation sequence is unique up to switch equivalence. Furthermore, the critical pairs generated by AGG can be used to find the conflicts between the rules which may cause non-functional behaviour of the model transformation and the modeler can decide whether to change the rules or to keep the non-functional behaviour.

Model Size	Model Transformation Sequences of CD2RDBM				
	without Filter NACs		with Filter NACs		
	Time[1)]	Success Rate	Time[1)]	Overhead	Success Rate
[Elements[2)]]	[ms]	[%]	[ms]	[%]	[%]
11	143,75	42,86	158,33	10,14	100,00
25	302,75	16,84	335,45	10,80	100,00
53	672,68	3,94	742,62	10,40	100,00
109	1.481,43	0,17	1.584,86	6,98	100,00

1) Average time of 100 successful model transformation sequences

2) Nodes and Edges

Table 1: Benchmark, Tool: AGG [21]

Efficient Execution: Filter NACs do not only improve the analysis of functional behaviour of a TGG, but also the execution of the model transformation process by forbidding the application of misleading transformation steps that would lead to a dead-end eliminating the need of backtracking for these cases. Table 1 shows execution times using the transformation engine AGG [21]. The additional overhead caused by filter NACs is fairly small and lies in the area of 10% for the examples in the benchmark, which is based on the average execution times for 100 executions concerning models with 11, 25, 53 and 109 elements (nodes and edges), respectively. The first model with 11 elements is the presented class diagram in the source component of Fig. 8. The listed times concern successful execution paths only, i.e. those executions that lead to a completely translated model. The success rate for transformations without filter NACs is lower for larger models and in particular, for the model with 109 elements the success rate is below 0.2%. This means that the error rate is above 99.8% while the success rate for the system with filter NACs is always 100%. Times for the unsuccessful executions, which appear in the system without filter NACs, are not considered. However, in order to ensure completeness there is the need for backtracking for the system without filter NACs. This backtracking overhead is in general exponential. In our case study, e.g., the rule "Class2Table" can be applied at misleading parts already at the beginning of a transformation. Backtracking for general model transformation systems is reduced by filter NACs and avoided completely in the case that no "significant critical pair" remains present (see Thm. 3), which we have shown to be fulfilled for our example. The additional overhead of about 10% for filter NACs is in most cases much smaller than the efforts for backtracking.

Moreover, in order to perform model transformations using highly optimized transformation machines for plain graph transformation, such as Fujaba and GrGen.Net [19], we have presented how the transformation rules and models can be equivalently represented by plain graphs and rules. First of all, triple graphs and morphisms are flattened according to the construction presented in [2, 13], which can be extended to NACs using the flattening of morphisms. Furthermore, we presented in this paper how forward rules with NACs are extended to forward translation rules with NACs, such that the control condition "source consistency" [5] and also the gluing condition (Fact 2) are ensured automatically for complete sequences, i.e. they do not need to be checked during the transformation.

Summing up, the presented results allow us to combine the easy, intuitive and formally well founded specification of model transformations based on triple graph grammars with NACs with the best available tools for executing graph transformations while still ensuring correctness and completeness.

5 Related Work

Since 1994, several extensions of the original TGG definitions [17] have been published [18, 15, 8] and various kinds of applications have been presented [20, 9, 14]. The formal construction and analysis of model transformations based on TGGs has been started in [5] by analyzing information preservation of bidirectional model transformations and continued in [2, 4, 3, 6, 13], where model transformations based on TGGs are compared with those on plain graph grammars in [2], TGGs with specification NACs are analyzed in [6] and an efficient on-the-fly construction is introduced in [3]. A first approach analyzing functional behaviour was presented for restricted TGGs with distinguished kernels in [4] and a more general approach, however without NACs, based on forward translation rules in [13]. The results in this paper for model transformations based on forward translation rules with specification and filter NACs are based on the results of all these papers except of [4].

In [5] a similar case study based on forward rules is presented, but without using NACs. This causes that more TGT-sequences are possible, in particular, an association can be transformed into a foreign key with one primary key, even if there is a second primary attribute that will be transformed into a second primary key at a later stage. This behaviour is not desired from the application point of view. Thus, the extended grammar with NACs in this paper handles primary keys and foreign keys in a more appropriate way. Furthermore, the current extended version has strong functional behaviour as shown in Sec. 3.

In the following we discuss how the presented results can be used to meet the “Grand Research Challenge of the TGG Community” formulated by Schürr et.al. in [18]. The main aims are “Consistency”, “Completeness”, “Expressiveness” and “Efficiency” of model transformations. The first two effectively require correctness, completeness w.r.t. the triple language *VL* and additionally termination and functional behaviour. They are ensured as shown in Sec. 3. “Expressiveness” requires suitable control mechanisms like NACs, which are used extensively in this paper and we further extend the technique by additional control mechanisms. In [7] more general application conditions [10] are considered, but functional behaviour is not yet analyzed. Finally, we discussed in Sec. 4 that our approach can be executed efficiently based on efficient graph transformation engines. Especially model transformations fulfilling the conditions in Thm. 3 do not need to backtrack, which bounds the number of transformation steps to the elements in the source model as required in [18].

6 Conclusion

In this paper we have studied model transformations based on triple graph grammars (TGGs) with negative application conditions (NACs) in order to improve efficiency of analysis and execution compared with previous approaches in the literature. The first key idea is that model transformations can be constructed by applying forward translation rules with NACs, which can be derived automatically from the given TGG-rules with NACs. The first main result shows termination under weak assumptions, correctness and completeness of model transformations in this framework, which is equivalent to the approach in [6]. The second key idea is to introduce filter NACs in addition to the NACs in the given TGG-rules, which in contrast are called specification NACs in this paper. Filter NACs are useful to improve the analysis of functional behaviour for model transformations based on critical pair analysis (using the tool AGG [21]) by filtering out backtracking paths and this way, some critical pairs. The second main result provides a sufficient condition for functional behaviour based on the analysis of critical pairs for forward translation rules with filter NACs. If we are able to construct filter NACs such that the corresponding rules have no more “significant” critical pairs, then the third main result shows that we have strong functional behaviour, i.e. not only the results are unique up to isomorphism but also the corresponding model transformation sequences are switch-equivalent up to isomorphism. Surprisingly, we can show that the condition “no significant critical pairs” is not only sufficient, but also necessary for strong functional behaviour. Finally, we discuss efficiency aspects of analysis and execution of model transformations and show that our sample model transformation *CD2RDBM* based on TGG-rules with NACs has strong functional behaviour.

The main challenge in applying our main results on functional and strong functional behaviour is to find suitable filter NACs, such that we have a minimal number of critical pairs for the forward translation rules with filter NACs. For this purpose, we are able to provide static and dynamic techniques for the generation of filter NACs (see Facts 3 and 4). The dynamic technique includes a check that certain models are misleading. In any case, the designer of the model transformation can specify some filter NACs directly by himself, if he can ensure the filter NAC property. In future work, we will study further static conditions to check whether a model is "misleading", because this allows to filter out misleading execution paths. In addition to that, we currently develop a extensions to layered model transformations and amalgamated rules, which allow to further reduce backtracking in general cases and to simplify the underlying rule sets.

References

[1] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs, Springer (2006)

[2] 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'08. ACM (2008)

[3] 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.) Proc. ACM/IEEE MODELS'09. LNCS, vol. 5795, pp. 241–255. Springer (2009)

[4] 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.) Proc. ICGT'08. LNCS, vol. 5214, pp. 178–193. Springer (2008)

[5] Ehrig, H., Ehrig, K., Ermel, C., Hermann, F., Taentzer, G.: Information preserving bidirectional model transformations. In: Dwyer, M.B., Lopes, A. (eds.) Proc. FASE'07. LNCS, vol. 4422, pp. 72–86. Springer (2007)

[6] 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'09. EC-EASST, vol. 18. EASST (2009)

[7] Golas, U., Ehrig, H., Hermann, F.: Enhancing the Expressiveness of Formal Specifications for Model Transformations by Triple Graph Grammars with Application Conditions (2010), (Submitted to GCM'10, online available at `http://tfs.cs.tu-berlin.de/publikationen/Papers10/GEH10.pdf`)

[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.) Proc. ICGT'06. LNCS, vol. 4178, pp. 351–366. Springer (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, Fak. IV (2010), `http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/`

[12] Hermann, F., Ehrig, H., Golas, U., Orejas, F.: Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars. In: Proc. Int. Workshop on Model Based Interoperability (MDI'2010). ACM (2010), to appear.

[13] Hermann, F., Ehrig, H., Orejas, F., Golas, U.: 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] 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)

[15] 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 (2006)

[16] Lambers, L.: Certifying Rule-Based Models using Graph Transformation. Ph.D. thesis, Technische Universität Berlin (November 2009)

[17] Schürr, A.: Specification of Graph Translators with Triple Graph Grammars. In: Tinhofer, G. (ed.) Proc. WG'94. LNCS, vol. 903, pp. 151–163. Springer (1994)

[18] Schürr, A., Klar, F.: 15 years of triple graph grammars. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) Proc. ICGT'08. pp. 411–425. LNCS, Springer (2008)

[19] Taentzer, G., Biermann, E., Bisztray, D., Bohnet, B., Boneva, I., Boronat, A., Geiger, L., Geiß, R., Horvath, A., Kniemeyer, O., Mens, T., Ness, B., Plump, D., Vajk, T.:

Generation of Sierpinski Triangles: A Case Study for Graph Transformation Tools. In: Schürr, A., Nagl, M., Zündorf, A. (eds.) Proc. AGTIVE'07. LNCS, vol. 5088, pp. 514 – 539. Springer (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'05 (2005)

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