*Math. Struct. in Comp. Science* (2014), vol. 24, iss. 4, e240408, 57 pages. 
doi:10.1017/S0960129512000370

# Formal analysis of model transformations based on triple graph grammars

FRANK HERMANN[†‖], HARTMUT EHRIG[‡‖], ULRIKE GOLAS[§] and FERNANDO OREJAS[¶]

[†]*Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, Berlin, Germany*
*and*
*Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg*
*Email:* frank.hermann@uni.lu
[‡]*Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, Berlin, Germany*
*Email:* ehrig@cs.tu-berlin.de
[§]*Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany*
*Email:* golas@zib.de,
[¶]*Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain*
*Email:* orejas@lsi.upc.edu

*Received 15 February 2011; revised 7 January 2012*

Triple graph grammars (TGGs) are a well-established concept for the specification and execution of bidirectional model transformations within model driven software engineering. Their main advantage is an automatic generation of operational rules for forward and backward model transformations, which simplifies specification and enhances usability as well as consistency. In this paper we present several important results for analysing model transformations based on the formal categorical foundation of TGGs within the framework of attributed graph transformation systems.

Our first main result shows that the crucial properties of correctness and completeness are ensured for model transformations. In order to analyse functional behaviour, we generate a new kind of operational rule, called a *forward translation rule*. We apply existing results for the analysis of local confluence for attributed graph transformation systems. As additional main results, we provide sufficient criteria for the verification of functional behaviour as well as a necessary and sufficient condition for strong functional behaviour. In fact, these conditions imply polynomial complexity for the execution of the model transformation. We also analyse information and complete information preservation of model transformations, that is, whether a source model can be reconstructed (uniquely) from the target model

[¶] The participation of Fernando Orejas was supported by the MEC project FORMALISM (ref. TIN2007-66923).
[‖] The participation of Frank Hermann and Hartmut Ehrig was supported by the DFG project BEHAVIOUR-GT.

computed by the model transformation. We illustrate the results for the well-known model transformation example from class diagrams to relational database models.

## 1. Introduction

### 1.1. *The main challenges for model transformations*

Model transformations are a key concept for modular and distributed model driven development. They are widely used for model optimisation and other forms of model evolution. Moreover, model transformations are used to map models between different domains in order to perform code generation or to apply analysis techniques. In this multi-domain context, triple graph grammars (TGGs) have been applied in several case studies and have exhibited a convenient combination of formal and intuitive specification abilities.

Triple graph grammars were introduced in Schürr (1994) and have been used, among other things, for the specification and execution of bidirectional model transformations between domain specific languages (DSLs). The power of bidirectional model transformations is based on the simultaneous support of transformations in both the forward and backward directions. In addition to the general advantages of bidirectional model transformations, TGGs simplify the design of model transformations. A single set of triple rules is sufficient to generate the operational rules for the forward and backward model transformations.

In the current paper, we consider a number of important properties for model transformations. More precisely, assuming that we have specified a class of transformations using a triple graph grammar, we study the following properties of forward transformations from the class of source models to the class of target models:

(1) *Syntactical correctness and completeness*:
The syntactical correctness of a transformation method means that if we can transform any source model $G^S$ into a model $G^T$ using the method, then the model $G^T$ is a valid target model and, moreover, the pair $(G^S, G^T)$ is consistent with respect to the specification of the model transformation provided by the triple graph grammar. Completeness, on the other hand, means that given any consistent pair $(G^S, G^T)$ according to the specification, our transformation method will be able to build $G^T$ from $G^S$.

(2) *Functional and strong functional behaviour*:
Functional behaviour means that for each source model $G^S$, each forward transformation starting with $G^S$ leads to a unique valid target model $G^T$. Strong functional behaviour means that the forward transformation from $G^S$ to $G^T$ is also essentially unique, that is, unique up to switchings of independent transformation steps.

(3) *Information and complete information preservation*:
In the case of bidirectional model transformations, information preservation means that for each forward transformation from $G_S$ to $G_T$, there is also a backward

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

transformation from $G_T$ to $G_S$. Complete information preservation means that in addition, each backward transformation starting with $G_T$ leads to the same $G_S$.

The main aim of the current paper is to determine what conditions are required to ensure that the properties defined above can be guaranteed, and how these conditions can be checked with suitable tool support. We do not consider other important properties, such as semantical correctness, which for a forward transformation from $G^S$ to $G^T$ means that $G^S$ and $G^T$ are semantically equivalent in a suitable sense – see Bisztray *et al.* (2009) and Hermann *et al.* (2010d) for further details.

### 1.2. *Model transformations based on TGGs and the main results of the paper*

The key idea for the execution of model transformations using TGGs is to preserve the given source model and add the missing target and correspondence elements in separate but connected components. For this reason, the transformation rules add new structures and do not necessarily need to delete existing elements. The resulting target model is obtained by type restriction. Indeed, non-deleting triple rules are sufficient for many case studies. However, in general, it may be very difficult, if not impossible, to specify a model transformation whose validity depends on some global properties of the given models. An example of this may arise with automata minimisation, where we transform a finite automaton into an automaton with the same behaviour, but with the smallest possible set of states. In this case, the transformation should translate any two states with the same behaviour into a single state. However, knowing if two states have the same behaviour is a global property of the given automaton. Nevertheless, a possible solution to simplify the model transformation is to perform some additional pre-processing of the source model or post-processing of the target model. For this reason, and since it is common practice for TGGs, we consider transformation rules that are non-deleting.

Using Ehrig *et al.* (2009a) as a basis, we will show in our first main result (Theorem 2.10) that syntactical correctness and completeness are ensured for model transformations based on the formal control condition of source consistency. Moreover, we can ensure termination for the execution of a model transformation by the static condition that all TGG-rules are creating on the source component, that is, they add elements of the source domain language. This condition can be checked automatically.

In the general context of transformation systems, it is well known that termination and local confluence implies confluence and hence functional behaviour, where local confluence can be checked by analysing all critical pairs between pairs of transformation rules. However, in the context of model transformations, a weaker notion of confluence is sufficient because the uniqueness of results is only required for successful executions of the model transformation. Moreover, we also have to include the control condition of source consistency in the analysis. To this end, we generate a new kind of operational rule, called forward translation rules, which extend forward rules by additional attributes keeping track of the elements that have already been translated.

In our main results for the analysis of functional behaviour, we extend the results of Hermann *et al.* (2010a) and Hermann *et al.* (2010c), and also provide a less restrictive

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

condition for functional behaviour. We will show in Theorem 3.13 that the functional behaviour of model transformations is ensured by strict confluence of all significant critical pairs of the forward translation rules. This means that several critical pairs can be ignored if they are not significant. We will also analyse the strong functional behaviour of model transformations, where uniqueness is also required for the transformation sequences up to switch equivalence. In Theorem 3.16, we characterise strong functional behaviour by the absence of significant critical pairs. The results for functional behaviour can also be used to improve the execution efficiency so that we can ensure polynomial time complexity if the appropriate sufficient conditions are satisfied (see Section 5.1).

Finally, we will analyse the information preservation of model transformations, that is, the problem of whether a source model can be reconstructed from the target model computed by the model transformation. To do this, we extend the results we presented in Ehrig *et al.* (2007) to TGGs with application conditions and to the notion of complete information preservation. We will show in Theorem 3.19 that model transformations based on forward rules always ensure information preservation, which requires that there is a backward transformation sequence starting at the derived target model that results in the given source model. Then, in Theorem 3.22, we will provide a sufficient condition for complete information preservation, that is, that any reconstructed source model coincides with the original.

### 1.3. *Mathematical framework*

The mathematical background for the current paper is the algebraic theory of graph transformations (Rozenberg 1997), and, in particular, the double pushout approach for graphs introduced in Ehrig *et al.* (1973), Rozenberg (1997) and Ehrig *et al.* (2006). This approach has been generalised from graphs to adhesive, adhesive HLR and M-adhesive categories (Lack and Sobociński 2005; Ehrig *et al.* 2006; Ehrig *et al.* 2010). These are categorical frameworks where specific constructions like pushouts and pullbacks exist and are compatible with each other. This allows the categorical theory to be instantiated not only for graphs, but also for several other high-level structures, including typed and attributed graphs, hypergraphs and various kinds of Petri net.

In our approach to model transformations, the abstract syntax of models is given by typed attributed graphs in the sense of Ehrig *et al.* (2006). In fact, the main parts of the theory can be presented in adhesive or $\mathcal{M}$-adhesive categories (Lack and Sobociński 2005; Ehrig *et al.* 2010) as shown in Hermann *et al.* (2010c). However, for simplicity, the construction of forward translation rules in the current paper is based on attributed graphs, which we will just call graphs for short. However, the corresponding category of typed attributed graphs (see Appendix A) is an important example of an $\mathcal{M}$-adhesive category.

In the current paper, we will assume basic knowledge of the algebraic theory of graph transformations as presented, for example, in Part I of Ehrig *et al.* (2006), but no knowledge of general category theory is assumed. See Section 5 for an informal summary of our main results and some potential applications that go beyond our running example.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

### 1.4. *Organisation of the paper*

In Section 2, we present model transformations based on forward rules and forward translation rules as well as our first main result concerning the correctness and completeness of model transformations. In Section 3, we develop our main results for analysing functional behaviour and information preservation, that is, whether and how source models can be (completely) reconstructed from target models. Then, in Sections 4 and 5, we discuss related work and present our conclusions. Finally, in Appendix A, we recall the technical details of the $\mathcal{M}$-adhesive category of typed attributed graphs and in Appendix B, we prove some auxiliary facts – the proofs of the main results (Theorems 2.10, 3.13, 3.16, 3.19 and 3.22) are given in the main part of the paper.

## 2. Model transformation based on triple graph grammars

Triple graph grammars were developed by Schürr, and are a technique for specifying (bidirectional) model transformations (Schürr 1994). In particular, a triple graph grammar describes a class of triple graphs consisting of pairs of models together with the relation between their elements. More precisely, a triple graph

$$G = \left(G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T\right)$$

consists of a source graph $G_S$ and a target graph $G_T$, which are related through a correspondence graph $G_C$ and two graph morphisms

$$s_G : G_C \to G_S$$
$$t_G : G_C \to G_T$$

specifying how source elements correspond to target elements. In this context, the target graph of $G$ may be considered to be the *forward* transformation of its source graph and the source graph may be considered to be the *backward* transformation of its target graph. Moreover, a given set of triple graphs can be seen as a class of model transformations, and the triple graph grammar that generates this set may be considered to be its specification.

Triple graphs are related by means of triple graph morphisms, which, as we would expect, are formed by three graph morphisms. More precisely, a triple graph morphism

$$m = (m_S, m_C, m_T) : G \to H$$

consists of

$$m_S : G_S \to H_S$$
$$m_C : G_C \to H_C$$
$$m_T : G_T \to H_T$$

such that

$$m_S \circ s_G = s_H \circ m_C$$
$$m_T \circ t_G = t_H \circ m_C.$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

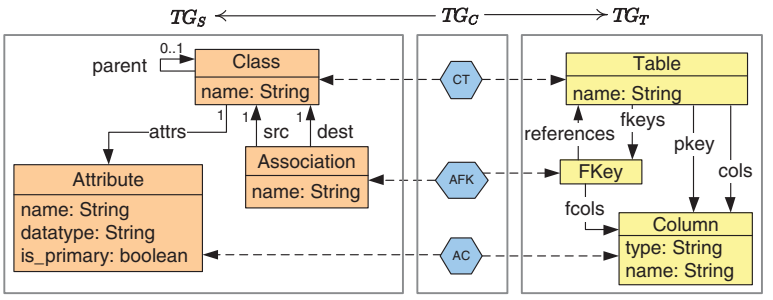



Fig. 1. (Colour online) Triple type graph for *CD2RDBM*

$$\begin{array}{ccccc} G = \left(G^S & \xleftarrow{s_G} & G^C & \xrightarrow{t_G} & G^T\right) \\ \downarrow m & & \downarrow m^S & & \downarrow m^C & & \downarrow m^T \\ H = \left(H^S & \xleftarrow[s_H]{} & H^C & \xrightarrow[t_H]{} & H^T\right) \end{array}$$

We can use any kind of graph inside a triple graph, as long as they form an adhesive (or $\mathcal{M}$-adhesive) category (Lack and Sobociński 2005; Ehrig *et al.* 2006; Ehrig *et al.* 2010). This means that we can have triple graphs (or, more accurately, triple structures) consisting of many kinds of graphical structures. In the current paper, we use attributed triple graphs based on E-graphs as presented in Ehrig *et al.* (2007). Moreover, our triple graphs are assumed to be typed over a given triple type graph $TG$. As usual, the typing is done by a triple graph morphism $type_G : G \to TG$.

**Example 2.1 (triple type graph).** Figure 1 shows the triple 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 the class diagrams and the target component specifies the structure of the relational database models. Classes correspond to tables, attributes to columns and associations to foreign keys. Throughout the example, which was originated in Ehrig *et al.* (2007), elements are arranged on the left-hand side, centre and right-hand side according to whether the component types are the source, correspondence or target. Attributes of structural nodes and edges are shown within their containing structural nodes and edges, respectively. Formally, attribute values are edges to additional data value nodes (see Appendix A). Note that the correspondence component is important for relating the source elements to their aligned target elements. For this reason, it is used in applications to navigate using the traceability links from the source structures to the target structures, or *vice versa*. The morphisms between the three components are represented by dashed arrows. The multiplicity constraints shown are ensured by the triple rules of the grammar in Figures 2–4. Moreover, the source language contains only those class diagrams in which the classes have unique primary attributes.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

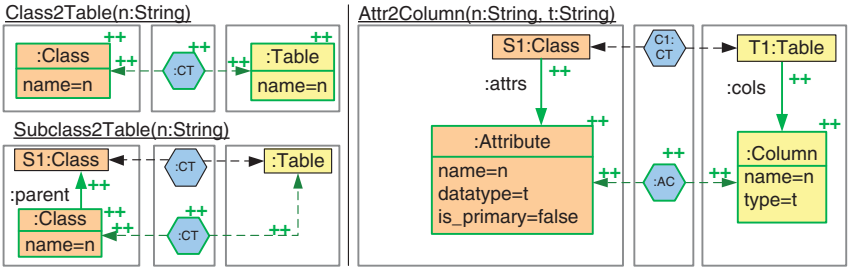



Fig. 2. (Colour online) Rules for the model transformation *CD2RDBM*, Part 1

A rule $tr$ in a triple graph grammar is called a triple rule and is an injective triple graph morphism

$$tr = \left(tr^S, tr^C, tr^T\right) : L \to R$$

and, without loss of generality, we can assume $tr$ to be an inclusion:

$$\begin{array}{ccccc} L = \left(L^S & \xleftarrow{s_L} & L^C & \xrightarrow{t_L} & L^T\right) \\ \downarrow{\scriptstyle tr} & & \downarrow{\scriptstyle tr^S} & & \downarrow{\scriptstyle tr^C} \quad \downarrow{\scriptstyle tr^T} \\ R = \left(R^S & \xleftarrow[s_R]{} & R^C & \xrightarrow[t_R]{} & R^T\right) \end{array}$$

A triple rule is applied to a triple graph $G$ by matching $L$ to some sub-triple-graph of $G$. Technically, a match is a morphism $m : L \to G$. The result of this application is the triple graph $H$, where $L$ is replaced by $R$ in $G$. Technically, the result of the transformation is defined by a pushout diagram with comatch $n : R \to H$ and transformation inclusion $t : G \hookrightarrow H$:

$$\begin{array}{ccc} L & \overset{tr}{\hookrightarrow} & R \\ {\scriptstyle m}\downarrow & (PO) & \downarrow{\scriptstyle n} \\ G & \underset{t}{\hookrightarrow} & H \end{array}$$

This triple graph transformation (TGT) step is denoted by $G \overset{tr,m}{\Longrightarrow} H$. 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 2.2 (triple rules and triple transformation steps).** The triple rules in Figure 2 are part of the rules of the grammar *TGG* for the *CD2RDBM* model transformation. They are presented in abbreviated notation, that is, the left- and right-hand sides of a rule are shown in one triple graph. Elements that are created by the rule are labelled with '++' and marked by green lines. The '*Class2Table*' rule synchronously creates a class with name 'n' together with the corresponding table in the relational database. Accordingly, subclasses are connected to the tables of their super classes by the '*Subclass2Table*' rule. Note that this rule creates the new class node together with an edge of type parent implying

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

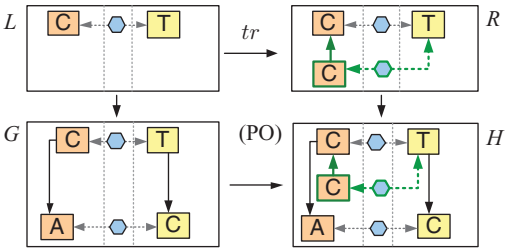



Fig. 3. (Colour online) Triple graph transformation step using the '*Subclass2Table*' rule

that our compact case study does not handle the case of multiple inheritance. Finally, the '*Attr2Column*' rule creates attributes with type 't' together with their corresponding columns in the database component. Figure 3 shows a triple graph transformation step $G \overset{tr,m}{\Longrightarrow} H$ using the $tr =$'*Subclass2Table*' rule, where we have ommitted the attribute values of the nodes and reduced the node types to the starting letters. The top line shows the rule with its left- and right-hand sides, and the bottom line shows the given triple graph $G$ and the resulting triple graph $H$. The effect of this step is the addition of a new subclass that is related to the existing table corresponding to the existing class.

From the point of view of applications, a model transformation should be injective on the structural part, that is, the transformation rules are applied along matches that do not identify structural elements. But it would be too restrictive to require injectivity of the matches on the data and variable nodes also because we must allow for the possibility that two different variables can be mapped to the same data value. For this reason, we introduce the notion of almost injective matches, which requires that matches are injective except for the data value nodes. In this way, attribute values can still be specified as terms within a rule and matched non-injectively to the same value. For the rest of the current paper, we will generally require almost injective matching for the transformation sequences.

**Definition 2.3 (almost injective match).** An attributed triple graph morphism $m : L \to G$ is called an *almost injective match* if it is non-injective at most for the set of variables and data values.

In graph transformations, negative application conditions (NACs for short) allow us to restrict the application of transformation rules when certain structures are present in the given object graph (see, for instance, Ehrig *et al.* (2006)). In the current paper, we consider NACs for triple rules, following Ehrig *et al.* (2009a). Moreover, for most case studies of model transformations source–target NACs, that is, either the source or the target NACs, are sufficient, and we will regard them as the standard case. These NACs prohibit the existence of certain structures either in the source or in the target part only, while general NACs may prohibit both at once, or even structures in the correspondence graph. See Golas *et al.* (2011) for model transformations with more general application conditions.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Definition 2.4 (triple rule with negative application conditions).** Given a triple rule

$$tr = (L \to R),$$

a negative application condition (NAC)

$$(n : L \to N)$$

consists of a triple graph $N$ and a triple graph morphism $n$. A NAC with

$$n = \left(n^S, id_{L^C}, id_{L^T}\right)$$

is called a *source NAC* and a NAC with

$$n = \left(id_{L^S}, id_{L^C}, n^T\right)$$

is called a *target NAC*.

A match $m : L \to G$ is NAC consistent if there is no injective $q : N \to G$ 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.

For the rest of the current paper, we will only consider source and target NACs and almost injective matches, which is sufficient for many practical case studies.

Given a triple type graph $TG$, a set of triple rules $TR$ and a start graph

$$\emptyset = (\emptyset \leftarrow \emptyset \to \emptyset)$$

(usually, the empty triple graph), we write $VL$ to denote the set of integrated models (that is, triple models including elements in the source, target and correspondence component) that are generated from $\emptyset$ using the rules in $TR$. The source language $VL_S$ and target language $VL_T$ of $VL$ are then derived by projections to the triple components, that is,

$$VL_S = proj_S(VL)$$
$$VL_T = proj_T(VL).$$

Moreover, we denote the set of all models typed over the source component $TG^S$ of the triple type graph $TG$ by $VL\left(TG^S\right)$ implying directly that

$$VL_S \subseteq VL\left(TG^S\right).$$

Analogously, we write $VL\left(TG^T\right)$ to denote the set of all target models typed over $TG^T$ and have

$$VL_T \subseteq VL\left(TG^T\right).$$

**Example 2.5 (triple rules with nacs).** The remaining triple rules of the '*CD2RDBM*' model transformation are shown in Figure 4. The '*PrimaryAttr2Column*' rule extends '*Attr2Column*' from Example 2.2 by creating a new link of type '*pkey*' for the column and by setting the attribute value 'is_primary=true'. This rule contains NACs, which are specified in abbreviated notation. The NAC-only elements are specified by red lines

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

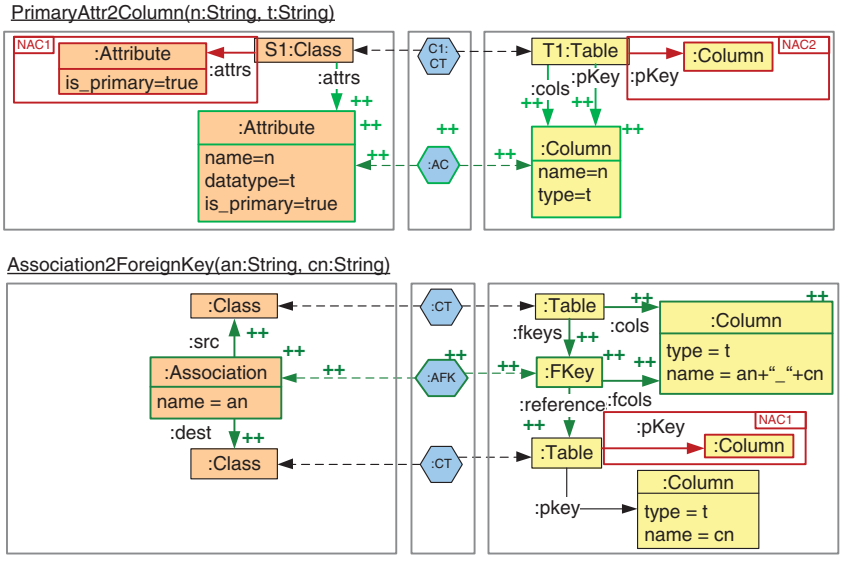



Fig. 4. (Colour online) Rules for the model transformation *CD2RDBM*, Part 2

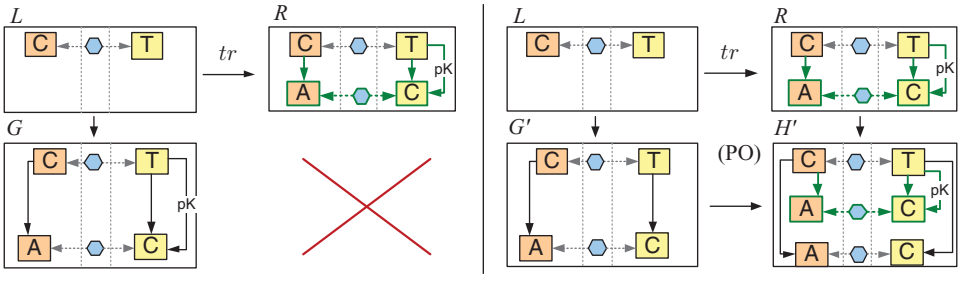



Fig. 5. (Colour online) Violation of NAC and satisfaction of NAC for the '*PrimaryAttr2Column*' rule

and with a surrounding frame with the label 'NAC'. A complete NAC is obtained by composing the left-hand side of a rule with the marked NAC-only elements. The source and a target NACs ensure that there is neither a primary attribute in the class diagram nor a primary key in the data base model when applying the rule. More formally, the NACs shown are actually NAC schemata (see Remark 2.6). The component on the left of Figure 5 shows a violation of the target NAC for the '*PrimaryAttr2Column*' rule, whose target NAC forbids the presence of an existing primary key at the matched table. The component on the right of the figure shows a NAC-consistent transformation step, where no primary key and no primary is present, and the existing attribute is assumed not to be a primary one. In a similar way to Figure 3, we use a compact notation for the transformation steps. The '*Association2ForeignKey*' rule in Figure 4 creates an association between two classes and the corresponding foreign key, where the parameters 'an' and 'cn' are used to set the names of the association and column nodes. The target NAC ensures that the primary key used for the foreign key in the data base component is unique.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

Note that the correspondence created by this rule implicitly assumes $m$ to 1 associations, because the foreign key uses the target primary key only. For the more general case of $m$ to $n$ associations, the example could be extended by an additional rule using a separate table that corresponds to an $m$ to $n$ association.

**Remark 2.6 (NACs for almost injective matches).** In order to simplify the specification of NACs for systems with almost injective matches, we interpret all specified NACs in a TGG as NAC schemata according to Hermann *et al.* (2014). A match $m : L \to G$ satisfies a NAC schema $n : L \to N$ effectively if there is no almost injective morphism $q : N \to G$, such that

$$q \circ n = m.$$

The difference compared with standard NACs is that the morphism $q$ is allowed to identify data values. According to Hermann *et al.* (2014, Fact 2.15), a NAC schema is equivalent to the set of all instantiated NACs, which are given by a structural copy of the NAC, but with an adapted data part for each possible data evaluation. This equivalence means that we can provide the formal results in the current paper using the standard notion of NAC satisfaction with injective morphism $q : N \to G$ according to Definition 2.4. Moreover, we can use the NAC schemata for the analysis and do not need to generate the instantiated NACs.

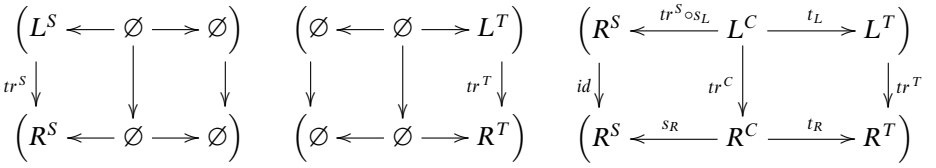
Triple graph grammars specify model transformations, but they do not directly solve the problem of how, given a source model (respectively, a target model), do we build its forward transformation (respectively, its backward transformation). However, as we will see in Sections 2.1 and 2.2, we can derive from a triple graph grammar the associated operational rules that are used for this task. In particular, in Section 2.1, we present model transformation in terms of forward (and backward) transformation rules, describing the main results (Schürr and Klar 2008; Ehrig *et al.* 2009a). Then, in Section 2.2, we present a more elaborate kind of rule, called forward (and backward) translation rules, based on the notion of translation attributes. These rules are the basis for the analysis of functional behaviour and information preservation in Section 3.

### 2.1. *Model transformation based on forward rules*

As we have already said, in order to describe how given source models can be transformed into corresponding target models, we use so-called operational rules, which are derived from the triple rules $TR$ as shown below. From each triple rule $tr$, we derive a source rule $tr_S$ and a forward rule $tr_F$ for forward transformation sequences for, respectively, parsing and constructing a model of the source language. As can be seen, source rules essentially consist of the source part of triple rules, so they may be used to generate or parse the valid source graphs. However, note that the set of graphs that can be generated by the source rules includes, but does not in general coincide with $VL_S$, *viz.* the source part of the triple graphs generated by the triple rules. That is, there may be models generated by the source rules that do not have a valid transformation according to the triple rules. The reason for this is that, at a certain moment, it may be impossible to apply a given triple

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

rule because we cannot match the target or the correspondence part of its left-hand side, but it may be possible to match just the source part of the rule (that is, its associated source rule).



source rule $tr_S$ target rule $tr_T$ forward rule $tr_F$

The intuition behind forward rules is quite simple. Given a certain source model $G_S$, we are trying to find a target model $G_T$ such that there is a triple graph

$$\left(G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T\right)$$

that can be generated by the given set of triple rules. This means that $G_T$ can be generated by the target part of the triple rules. However, the problem is knowing which target rules should be used. Instead, we use forward rules that restrict the choice of which possible rules to use in this construction. In particular, given a triple rule $tr$, in its associated forward rule $tr_F$, the source part of its left-hand side, $R_S$, coincides with the source part of the right-hand side of $tr$. This means that if there is a match of $tr_F$ in $G_S$, its source part could have been generated by $tr$ or, conversely, if there is no match of the source part of $tr_F$ in $G_S$, we would be unable to use $tr$ to generate the triple graph

$$\left(G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T\right).$$

Furthermore, by using forward rules, we are not only able to build $G_T$, but also the correspondence between $G_S$ and $G_T$.

If the given triple rules include NACs, these NACs are inherited by the operational rules as follows. Each forward rule $tr_F$ inherits the target NACs of its associated triple rule $tr$ since target NACs restrict the construction of target models. Conversely, source NACs restrict the construction of source models. For this reason, they are inherited by source rules. We will write $TR_S$ and $TR_F$ to denote the sets of all source and forward rules derived from $TR$. Analogously, we can 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 $TR_B$.

As introduced in Ehrig *et al.* (2007) and Ehrig *et al.* (2009a), the derived operational rules provide the basis for defining model transformations based on forward transformation sequences that are executed using the formal control condition of source consistency, which we will now briefly explain. We know that $G^T$ is the transformation of $G^S$ if the triple graph

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right)$$

is in the class defined by the TGG, that is, if there is a sequence of transformations

$$\varnothing \overset{tr_1}{\Longrightarrow} G_1 \Rightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n = G.$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

However, as can be seen from Fact 2.9, this sequence of transformations can be decomposed into a sequence of transformations using the associated source rules, followed by a sequence of transformations using the associated forward rules

$$\emptyset \overset{tr_{1S}}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} G_{n0} = (G^S \leftarrow \emptyset \rightarrow \emptyset) \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G,$$

where the source and forward sequences are *match consistent*, meaning that the matches of the corresponding source and forward steps are compatible. Technically, source and forward match are compatible if they coincide for each mapped element on their source component, that is,

$$m_S^S(x) = m_F^S(x),$$

assuming that the trace morphisms of the transformation sequences are inclusions. Moreover, Fact 2.9 also tells us that for every match consistent sequence of transformations

$$\emptyset \overset{tr_{1S}}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} G_{n0} \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G,$$

there is a corresponding sequence of triple rule transformations

$$\emptyset \overset{tr_1}{\Longrightarrow} G_1 \Rightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n = G.$$

This means that if we want to compute the transformation of a certain source model $G^S$, we can look for a sequence of forward transformations

$$\left(G^S \leftarrow \emptyset \rightarrow \emptyset\right) \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G,$$

such that the corresponding sequence of match consistent source transformations generates $G^S$, that is,

$$\emptyset \overset{tr_{1S}}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} \left(G^S \leftarrow \emptyset \rightarrow \emptyset\right).$$

These forward sequences are said to be *source consistent*. In principle, in order to find a source-consistent forward sequence, we must first parse the source model, that is, we must find the match consistent source sequence that generates $G^S$. However, Ehrig *et al.* (2009a) showed that source and forward sequences can be constructed simultaneously.

We will now look at some of these concepts in more detail.

**Definition 2.7 (model transformation based on forward rules).** A *model transformation sequence*

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

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

$$G^S = G_0^S$$
$$G^T = G_n^T.$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

A *model transformation*

$$MT : VL\left(TG^S\right) \Rightarrow VL\left(TG^T\right)$$

is defined by all model transformation sequences

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

with

$$G^S \in VL\left(TG^S\right)$$
$$G^T \in VL\left(TG^T\right).$$

All the corresponding pairs $\left(G^S, G^T\right)$ define the *model transformation relation*

$$MTR_F \subseteq VL(TG^S) \times VL(TG^T)$$

based on $TR_F$.

In Ehrig *et al.* (2007) and Ehrig *et al.* (2009a) we proved that source consistency ensures the (syntactical) correctness and completeness of model transformations based on forward rules with respect to the language $VL$ of integrated models. Syntactical correctness means that every model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

using forward rules leads to an integrated model

$$G_n = \left(G^S \leftarrow G^C \rightarrow G^T\right)$$

contained in $VL$. In other words, source-consistent forward transformations generate correct model transformations, according to the class of transformations specified by the given TGG. Completeness means that for any integrated model

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right) \in VL,$$

there is a corresponding model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G, G^T\right).$$

Intuitively, this means that any valid transformation specified by a TGG can be implemented by a source consistence forward transformation.

Note that the model transformation relation $MTR_F$ is in general not a function from $VL(TG^S)$ to $VL(TG^T)$ – see Section 3 for more on functional behaviour.

**Definition 2.8 (syntactical correctness and completeness).** A model transformation

$$MT : VL\left(TG^S\right) \Rightarrow VL\left(TG^T\right)$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

based on forward rules is:

— *syntactically correct* if for each model transformation sequence

$$\left(G{\cdot}G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

there is $G \in VL$ with

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right)$$

implying further that

$$G^S \in VL_S$$
$$G^T \in VL_T.$$

— *complete* if for each $G^S \in VL_S$ there is

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right) \in VL$$

with a model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

and

$$G_n = G.$$

Conversely, for each $G^T \in VL_T$ there is

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right) \in VL$$

with a model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

and

$$G_n = G.$$

Note that we have defined syntactical correctness and completeness in terms of forward model transformations. If we consider Definition 2.8 for both directions of a bidirectional model transformation, that is, for the forward and backward directions, we derive a more specific definition. In that case, the conditions for correctness and completeness are both required for all source and target models.

In order to show syntactical correctness and completeness for model transformations based on TGGs by Theorem 2.10, we use the following composition and decomposition result for TGT-sequences, which was shown in Ehrig *et al.* (2007) and Ehrig *et al.* (2009b) for the case of rules without and with NACs, respectively.

**Fact 2.9 (composition and decomposition of TGT-sequences).**

(a) **Decomposition**:

For each TGT-sequence

$$G_0 \overset{tr_1}{\Longrightarrow} G_1 \Rightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n \tag{1}$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

based on triple rules, there is a corresponding match consistent TGT-sequence

$$G_0 = G_{00} \overset{tr_{1S}}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} G_{n0} \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G_n \tag{2}$$

based on the corresponding source and forward rules.

(b) **Composition:**
For each match consistent transformation sequence (2) there is a corresponding transformation sequence (1).

(c) **Bijective Correspondence:**
Composition and decomposition are inverse to each other (up to isomorphism).

**Theorem 2.10 (syntactical correctness and completeness).** Each model transformation

$$MT : VL\left(TG^S\right) \Rrightarrow VL\left(TG^T\right)$$

based on forward rules is syntactically correct and complete.

*Proof.*

— **Syntactical correctness**:
Given a model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right),$$

the source consistency of $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ implies a match consistent sequence

$$\varnothing \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n.$$

Using the composition part of Fact 2.9, we have a corresponding TGT-sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_n$. This implies for $G = G_n$ that $G \in VL$ with

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right)$$

and thus

$$G^S \in VL_S$$
$$G^T \in VL_T$$

too.

— **Completeness**:
Given $G^S \in VL_S$, by the definition of $VL_S$, we have some

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right) \in VL.$$

This means we have a TGT-sequence $\varnothing \overset{tr^*}{\Longrightarrow} G$, and by the decomposition part of Fact 2.9, we have a match consistent sequence

$$\varnothing \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G,$$

which defines a model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G, G^T\right)$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

using

$$G = (G^S \leftarrow G^C \rightarrow G^T).$$

We use Remark 2.11 for the converse. □

**Remark 2.11 (composition and decomposition for the backward case).** For each TGT-sequence $G_0 \overset{tr^*}{\Longrightarrow} G_n$, there is also a corresponding match consistent backward TGT-sequence

$$G_0 = G_{00} \overset{tr_{1,T}}{\Longrightarrow} G_{01} \Rightarrow \ldots \overset{tr_{n,T}}{\Longrightarrow} G_{0n} \overset{tr_{1,F}}{\Longrightarrow} G_{1n} \Rightarrow \ldots \overset{tr_{n,F}}{\Longrightarrow} G_{nn} = G_n$$

based on target and backward rules leading to a backward model transformation

$$MT_B : VL\left(TG^T\right) \Rrightarrow VL\left(TG^S\right)$$

with similar results to those for the forward case.

The termination of model transformations is considered in Fact 3.11.

## 2.2. *Model transformation based on forward translation rules*

A major difficulty in implementing the techniques described in Section 2.1 is related to how we can check source consistency in a reasonably efficient way. In this section we show an approach, introduced in Hermann *et al.* (2010c), that solves this problem in a relatively simple way. Moreover, this approach sets the basis for the analysis of model transformations in Section 3.1.

The basic idea is to use what we call *translation attributes*, which tell us, as described in Example 2.14, which elements of the given source model have already been translated or used to build the target and the correspondence models. More precisely, given a source model $G^S$, if we think of building in parallel the match consistent sequences of source and forward transformations,

$$\varnothing \overset{tr_{1S}}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} G_{n0} = \left(G^S \leftarrow \varnothing \rightarrow \varnothing\right)$$

and

$$\left(G^S \leftarrow \varnothing \rightarrow \varnothing\right) \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G,$$

at any point $i$, when we apply the source transformation

$$G_{(i-1)0} \overset{tr_{iS}}{\Longrightarrow} G_{i0}$$

and the forward transformation

$$G_{n(i-1)} \overset{tr_{iF}}{\Longrightarrow} G_{ni},$$

all the elements in the source graph that are included in $G_{(i-1)0}$ will have their translation attributes set to true and the rest of them set to false since the elements in $G_{(i-1)0}$ are the elements that have been translated by the forward transformations

$$(G^S \leftarrow \varnothing \rightarrow \varnothing) \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{(i-1)F}}{\Longrightarrow} G_{n(i-1)}.$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

This means that:

(1) We must first enrich the given source graph with translation attributes assigning one translation attribute to each element (that is, each node, edge and attribute) of the source graph.
(2) Before starting the transformation process, we must set all translation attributes to false, since initially no element has already been used in building the target model.
(3) When applying the forward rule $tr_{iF}$:
   — On the one hand, we need to check that the associated source rule $tr_{iS}$ could be applied using a consistent match. This is equivalent to checking whether all the elements of the source graph that are matched by the left-hand side of the source rule have their translation attributes set to true, and all the elements that will be added by the source rule have their translation attributes set to false. Moreover, if that source rule includes some NAC, we would need to check that the subgraph of the source graph consisting of the elements with true translation attributes satisfies that NAC with respect to the given match.
   — On the other hand, we have to set all the translation attributes of the elements of the source graph that would have been added by $tr_{iS}$ to true.
(4) Finally, to check that the source model has been completely transformed into a target (and a correspondence) model, we need to check that at the end of the transformation, all the translation attributes of the source model are set to true. In this case, we say that the transformation sequence is *complete*.

The above explanation of how we use translation attributes may give the impression that the management of translation attributes (that is, checking if we can apply a transformation rule and updating the attributes after each transformation step) is external to the transformation process in the sense that the model transformation process is still done using forward rules, but checking and updating the translation attributes is done in some metaprocess. However, this is not true. A second key idea of our approach is that we can integrate the management of translation attributes into the transformation process. We do this by using a variant of forward rules that we call forward translation rules. More precisely, given a triple rule

$$tr = (L \to R),$$

its associated forward translation rule $tr_{FT}$, as described in Example 2.17, enriches its associated forward rule in the following ways:

— In the source part of the left-hand side of the rule, every element in $L_S$ has an associated translation attribute set to true, and every element in $L_R \setminus L_S$ has an associated translation attribute set to false. In this way, the matching of the rule ensures that in the given source graph, all elements that are expected to have been already created by previous source transformations have a true translation attribute, and all the elements that are supposed to be translated in this transformation step have a false translation attribute.
— In the source part of the right-hand side of the rule, every element in $R_S$ has an associated translation attribute set to true. In this way, the transformation defined

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

by the rule takes care of updating the translation attributes of the elements that are supposed to be translated in this transformation step.†

— Every NAC $n : L \to N$ of $tr$ (not just target NACs) is included in $tr_{FT}$, but all the elements in the source part of the NAC (either in $L$ or in $N$) are included with an associated translation attribute set to true.

The main result in this section shows that model transformations based on source-consistent forward TGT-sequences are equivalent to those based on complete forward translation TGT-sequences – see Fact 2.21. The source consistency control condition is ensured by the completeness of forward translation TGT-sequences, which are based on the generated forward translation rules. For this reason, the source consistency check for forward TGT-sequences is reduced to a check of whether the model is completely translated, that is, whether all translation attributes are set to true.

We will now provide the technical details of this approach, together with some examples. Even if the basic ideas, as we have seen, are relatively simple, some basic definitions are a bit involved because of the details of handling the translation attributes so, in a superficial reading of this part, it may be a good idea to skip these definitions.

In our notation, the translation attribute of each node, edge and attribute of a graph is labelled with the prefix '`tr`'. Note that we use different font styles for a triple rule $tr$ (italic) and for the prefix of translation attributes '`tr`' (typewriter) in order to emphasise the difference. Given an attributed graph $AG = (G, D)$ and a family of subsets $M \subseteq |G|$ for the domains $|G|$ of $G$, we say $AG'$ is a graph with translation attributes over $AG$ if it extends $AG$ with one new Boolean-valued attribute `tr_`$x$ for each element $x$ (node or edge) in $M$, and one new Boolean-valued attribute `tr_`$x$`_`$a$ for each attribute $a$ associated with 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 Ehrig *et al.* (2006), where attributes are possible for both nodes and edges. Attributed graphs consist of a graph for the structural part and an algebra for the data values, together with the attributes, which are edges between the structural elements (nodes and edges) and the data values. Roughly speaking, an attribution edge of a node points to the assigned value for the specific attribute. See Appendix A for more information about attributed graphs.

**Definition 2.12 (family with translation attributes).** Given an attributed graph

$$AG = (G, D),$$

we write

$$|G| = \left(V_G^G, V_G^D, E_G^G, E_G^{NA}, E_G^{EA}\right)$$

to denote the underlying family of sets containing all nodes and edges, and let $M \subseteq |G|$ with

$$\left(V_M^G, V_M^D, E_M^G, E_M^{NA}, E_M^{EA}\right).$$

† Note that forward translation rules are not just inclusions (that is, non-deleting rules) since some translation attributes are modified by the rule. As a consequence, as can be seen in Definition 2.15, forward translation rules are spans of inclusions, as in the general case of graph transformation rules.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

Then a *family with translation attributes* for $(G, M)$ extends $M$ by additional translation attributes and is given by

$$Att_M = \left(V_M^G, V_M^D, E_M^G, E^{NA}, E^{EA}\right)$$

with:

$$E^{NA} = E_M^{NA} \cup \left\{\texttt{tr\_}x \mid x \in V_M^G\right\} \cup \left\{\texttt{tr\_}x\texttt{\_}a \mid a \in E_M^{NA}, src_G^{NA}(a) = x \in V_G^G\right\}$$
$$E^{EA} = E_M^{EA} \cup \left\{\texttt{tr\_}x \mid x \in E_M^G\right\} \cup \left\{\texttt{tr\_}x\texttt{\_}a \mid a \in E_M^{EA}, src_G^{EA}(a) = x \in E_G^G\right\}.$$

**Definition 2.13 (graph with translation attributes).** Given an attributed graph

$$AG = (G, D)$$

and a family of subsets $M \subseteq |G|$ with

$$\{\mathbf{T}, \mathbf{F}\} \subseteq V_M^D,$$

let $Att_M$ be a family with translation attributes for $(G, M)$ according to Definition 2.12. Then,

$$AG' = (G', D)$$

is a *graph with translation attributes* over $AG$, where the domains $|G'|$ of $G'$ are given by the gluing via pushout of $|G|$ and $Att_M$ over $M$

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

and the source and target functions of $G'$ are defined as follows:

$$src_{G'}^G = src_G^G$$
$$trg_{G'}^G = trg_G^G$$

$$src_{G'}^X(z) = \begin{cases} src_G^X(z) & z \in E_G^X \\ x & z = \texttt{tr\_}x \text{ or } z = \texttt{tr\_}x\texttt{\_}a \end{cases} \quad \text{for } X \in \{NA, EA\}$$

$$trg_{G'}^X(z) = \begin{cases} trg_G^X(z) & z \in E_G^X \\ \mathbf{T} \text{ or } \mathbf{F} & z = \texttt{tr\_}x \text{ or } z = \texttt{tr\_}x\texttt{\_}a \end{cases} \quad \text{for } X \in \{NA, EA\}.$$

We write $Att_M^v$, where $v = \mathbf{T}$ or $v = \mathbf{F}$, to denote a family with translation attributes where all attributes are set to $v$. Moreover, we write

$$AG \oplus Att_M$$

to denote the fact that $AG$ is extended by the translation attributes in $Att_M$, that is,

$$AG \oplus Att_M = (G', D)$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

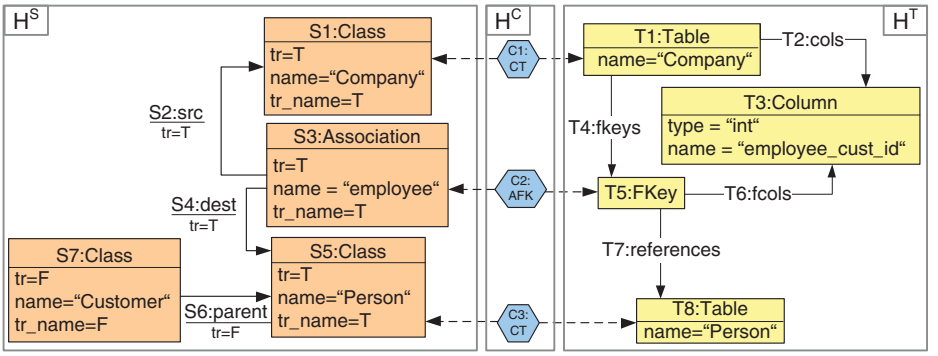



Fig. 6. (Colour online) Triple graph with translation attributes

for

$$AG' = (G', D)$$

as defined above. Analogously, we use the notion

$$AG \oplus Att^{v}_{M}$$

for translation attributes with value $v$, and we will use the abbreviated notation

$$Att^{v}(AG) := AG \oplus Att^{v}_{|G|}.$$

**Example 2.14 (triple graph with translation attributes).** Figure 6 shows the triple graph

$$H = \left(H^S \leftarrow H^C \rightarrow H^T\right),$$

which is extended by some translation attributes in the source component. The translation attributes with value '**T**' indicate that the owning elements have been translated during a model transformation sequence using forward translation rules, which will be defined in Definition 2.15. The remaining elements (edge S6, node S7 and the attribute 'name' of S7) in the source component are still marked with translation attributes set to '**F**'. These elements can still be matched and will become translated at later steps. The translation attributes are used to specify explicitly those elements that have been translated up to a specific step during the execution of a model transformation.

The concept of forward translation rules, which we introduced in Hermann *et al.* (2010c), extends the construction of forward rules by additional translation attributes in the source component. As described in Example 2.14, the translation attributes are used to keep track of the elements that have been translated so far. Since triple rules may create new attributes for existing nodes by definition, we also have to keep track of the translation of the attributes. The separate handling of nodes and their attributes is used, for example, in synchronisation scenarios (Hermann *et al.* 2011). To begin with, the source model of a model transformation sequence is extended by translation attributes that are all set to '**F**' and, step by step, they are set to '**T**' when their containing elements are translated by a forward translation rule.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Definition 2.15 (forward translation rule).** Given a triple rule

$$tr = (L \to R),$$

the *forward translation rule* of $tr$ is given by

$$tr_{FT} = \left(L_{FT} \xleftarrow{l_{FT}} K_{FT} \xrightarrow{r_{FT}} R_{FT}\right)$$

defined as follows using the forward rule

$$\left(L_F \xrightarrow{tr_F} R_F\right)$$

and the source rule

$$\left(L_S \xrightarrow{tr_S} R_S\right)$$

of $tr$, where we assume without loss of generality that $tr$ is an inclusion:

$$\begin{aligned} L_{FT} &= L_F \oplus Att^{\mathbf{T}}_{L_S} \oplus Att^{\mathbf{F}}_{R_S \setminus L_S} \\ K_{FT} &= L_F \oplus Att^{\mathbf{T}}_{L_S} \\ R_{FT} &= R_F \oplus Att^{\mathbf{T}}_{L_S} \oplus Att^{\mathbf{T}}_{R_S \setminus L_S} \\ &= R_F \oplus Att^{\mathbf{T}}_{R_S}, \end{aligned}$$

and $l_{FT}$ 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 an inclusion with

$$N_{FT} = (L_{FT} +_L N) \oplus Att^{\mathbf{T}}_{N_S \setminus L_S}.$$

**Remark 2.16.** Note that $(L_{FT} +_L N)$ is the union of $L_{FT}$ and $N$ with shared $L$ (formally a pushout), and, for a target NAC $n$, the forward translation NAC $n_{FT}$ does not contain any additional translation attributes because $N_S = L_S$. Given a set of triple rules $TR$, we write $TR_{FT}$ to denote the set of all $tr_{FT}$ with $tr \in TR$.

**Example 2.17 (derived forward translation rules).** The '*Subclass2Table*$_{FT}$' rule in Figure 7 is the derived forward translation rule of the '*Subclass2Table*' triple rule in Figure 2. Note that to improve readability in the figures, we abbreviate '`tr_`$x$' for an item (node or edge) $x$ by '`tr`' and '`tr_`$x$`_`$a$' by '`tr_`*type*$(a)$'. The compact notation of forward translation rules specifies the modification of translation attributes by '[**F** ⇒ **T**]', meaning that the attribute is matched with the value '**F**' and set to '**T**' during the transformation step. The detailed complete notation of a forward translation rule is shown on the right of Figure 7 for '*Subclass2Table*$_{FT}$'.

Figure 8 shows the forward translation rule with '*PrimaryAttr2Column*$_{FT}$' NACs derived from the triple '*PrimaryAttr2Column*' in Figure 4. According to Definition 2.15, the source

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

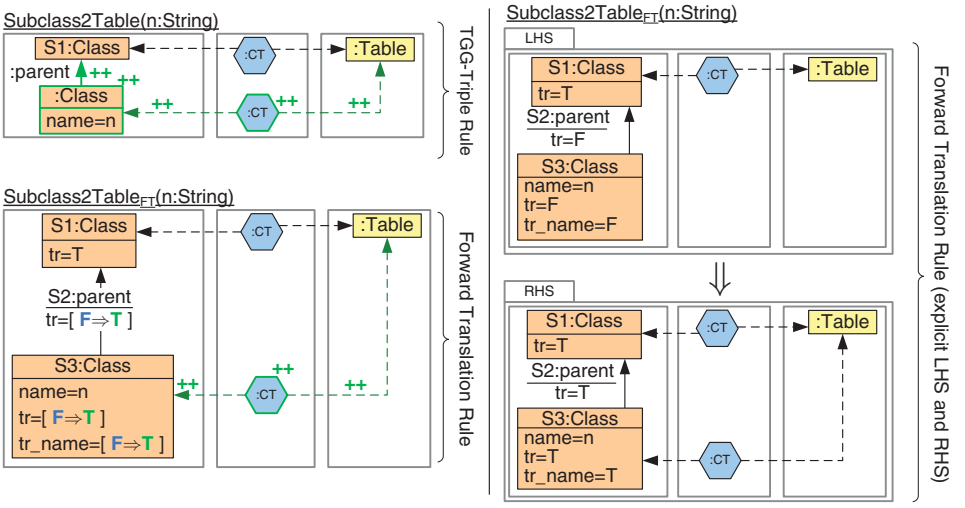



Fig. 7. (Colour online) Forward translation rule $Subclass2Table_{FT}(n : String)$

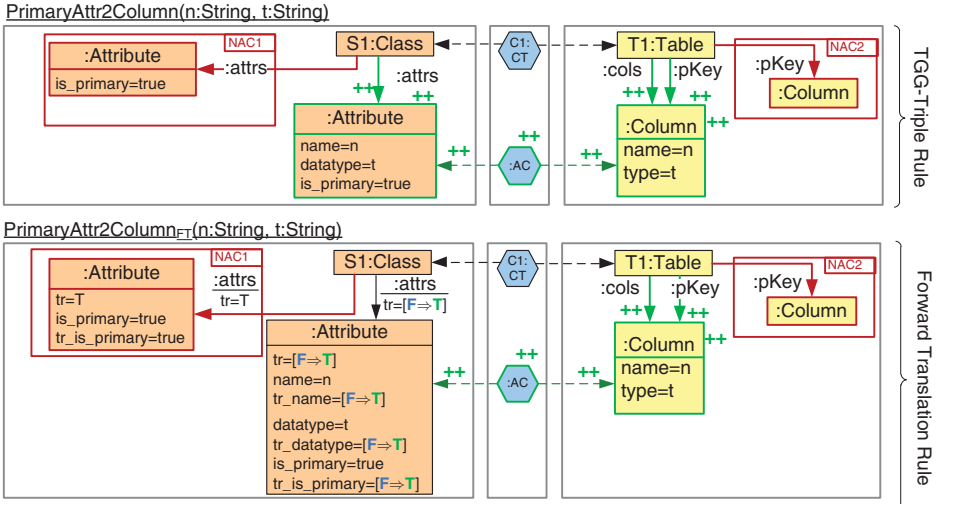



Fig. 8. (Colour online) Forward translation rule with NACs

elements of the triple rule 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 forward translation rule contains both the source and target NACs of the triple rule, where the NAC-only elements in the source NACs are extended by translation attributes set to '**T**'. Thus, a source NAC is only concerned with elements that have already been translated.

Since forward translation rules are deleting only on attribution edges, according to Hermann *et al.* (2010a, Fact 1), each NAC-consistent match is applicable. Note that in the general case of deleting rules, the additional gluing condition has to be checked (Ehrig

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

*et al.* 2006) in order to ensure, for example, that edges do not become dangling due to the deletion of nodes.

We will now define model transformations based on forward translation rules in the same way as for forward rules in Definition 2.7, where source consistency of the forward sequence is replaced by completeness of the forward translation sequence.

**Definition 2.18 (complete forward translation sequence).** A forward translation sequence $G_0 \overset{tr^*_{FT}}{\Longrightarrow} G_n$ with almost injective matches is said to be *complete* if no further forward translation rule is applicable and $G_n$ is *completely translated*, that is, all translation attributes of $G_n$ are set to true ('**T**').

**Definition 2.19 (model transformation based on forward translation rules).** A *model transformation sequence*

$$\left(G^S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G^T\right)$$

based on forward translation rules $TR_{FT}$ consists of a source graph $G^S$, a target graph $G^T$ and a complete TGT-sequence $G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n$ typed over

$$TG' = TG \oplus Att^{\mathbf{F}}_{|TG^S|} \oplus Att^{\mathbf{T}}_{|TG^S|}$$

based on $TR_{FT}$ with

$$G'_0 = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \varnothing \rightarrow \varnothing\right)$$
$$G'_n = \left(Att^{\mathbf{T}}\left(G^S\right) \leftarrow G^C \rightarrow G^T\right).$$

A *model transformation*

$$MT : VL\left(TG^S\right) \Rrightarrow VL\left(TG^T\right)$$

based on $TR_{FT}$ is defined by all model transformation sequences as above with

$$G^S \in VL\left(TG^S\right)$$
$$G^T \in VL\left(TG^T\right).$$

All the corresponding $\left(G^S, G^T\right)$ pairs define the *model transformation relation*

$$MTR_{FT} \subseteq VL\left(TG^S\right) \times VL\left(TG^T\right)$$

based on $TR_{FT}$. The model transformation is *terminating* if there are no infinite TGT-sequences through $TR_{FT}$ starting with

$$G'_0 = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \varnothing \rightarrow \varnothing\right)$$

for some source graph $G^S$.

**Example 2.20 (model transformation using forward translation rules).** Figure 9 shows the triple graph resulting from a forward translation sequence. The execution starts by extending the source model $G^S$ with translation attributes according to Definition 2.19, that is,

$$G'_0 = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \varnothing \rightarrow \varnothing\right).$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

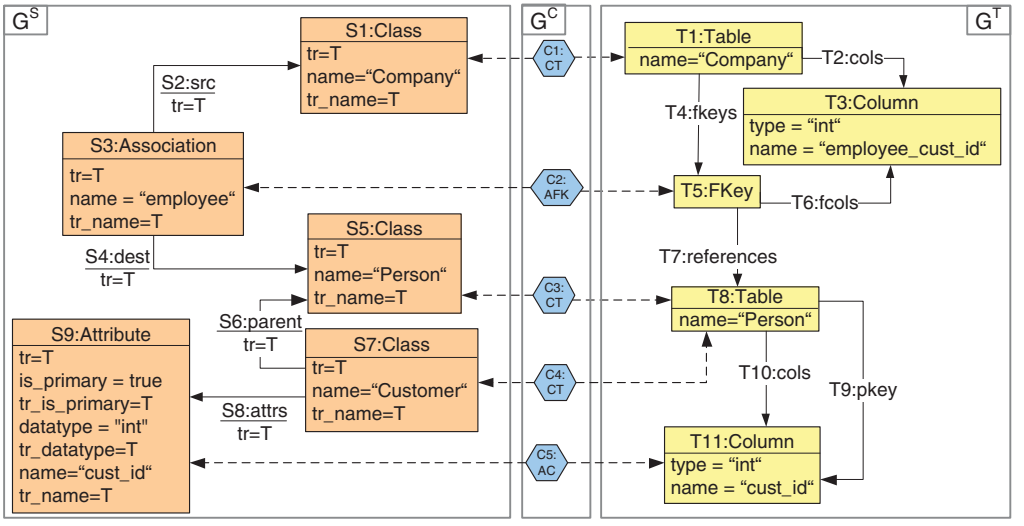



Fig. 9. (Colour online) Triple graph instance with translation attributes for *CD2RDBM*

We can execute the forward translation sequence using the following sequence of forward translation steps.

$$G'_0 \overset{Class2Table_{FT}}{\Longrightarrow} G'_1 \overset{Class2Table_{FT}}{\Longrightarrow} G'_2 \overset{Subclass2Table_{FT}}{\Longrightarrow} G'_3 \overset{PrimaryAttr2Col_{FT}}{\Longrightarrow} G'_4 \overset{Association2FKey_{FT}}{\Longrightarrow} G'_5,$$

with $G'_5$ being the graph $G$ in Figure 9. The triple graph $G'_5$ has now been completely translated because all translation attributes are set to '**T**'. No further forward translation rule is applicable, and we derive the resulting target model $G^T$ by restricting $G'^5$ to its target component, that is,

$$G^T = G'^T_5.$$

According to the equivalence of the model transformation concepts based on forward and forward translation rules in Fact 2.21, we can further conclude that $G^T$ can be equivalently obtained using a source-consistent forward transformation sequence based on forward rules without translation attributes.

We will show in Fact 2.21 that the model transformation sequences based on forward translation rules are one-to-one with model transformation sequences based on forward rules, that is, based on source-consistent forward sequences. For this reason, we can equivalently use either concept, and chose one of them according to the particular needs of the situation. While the concept based on source consistency has advantages in formal proofs, the concept based on forward translation rules has advantages when it comes to analysis and efficiency, as we will show in Section 3.1. The proof of Fact 2.21 is given in Appendix B.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Fact 2.21 (equivalence of forward transformation and forward translation sequences).** Given a source model $G^S \in VL(TG^S)$, the sets of forward rules $TR_F$ and the corresponding forward translation rules $TR_{FT}$, the following are equivalent for almost injective matches:

(1) There is a model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right)$$

based on $TR_F$ with

$$G_0 = \left(G^S \leftarrow \emptyset \rightarrow \emptyset\right)$$
$$G_n = \left(G^S \leftarrow G^C \rightarrow G^T\right).$$

(2) There is a model transformation sequence

$$\left(G^S, G_0' \overset{tr_{FT}^*}{\Longrightarrow} G_n', G^T\right)$$

based on $TR_{FT}$ with

$$G_0' = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \emptyset \rightarrow \emptyset\right)$$
$$G_n' = \left(Att^{\mathbf{T}}\left(G^S\right) \leftarrow G^C \rightarrow G^T\right).$$

Moreover, the model transformation relation $MTR_F$ for the model transformation based on forward rules coincides with the model transformation relation $MTR_{FT}$ for the model transformation based on forward translation rules, that is,

$$MTR_F = MTR_{FT}.$$

## 3. Analysis of functional behaviour and information preservation

As we showed in Section 2, we can ensure syntactical correctness and completeness for model transformations based on forward rules, and equivalently for those based on forward translation rules using Fact 2.21. This section concentrates on an analysis of functional behaviour and information preservation.

### 3.1. *Functional behaviour and efficient execution*

Functional behaviour of a model transformation means that each model of the source domain specific language (DSL) $\mathcal{L}_S$ is transformed into a unique model of the target language, where we require $\mathcal{L}_S \subseteq VL_S$ in order to ensure correctness and completeness by Theorem 2.10. The source DSL can form any subset of $VL_S$, and it can be specified by the type graph $TG^S$ together with additional well-formedness constraints. In many cases, model transformations should ensure the crucial property of functional behaviour. Moreover, in order to ensure efficient executions of model transformations, backtracking should be reduced or eliminated. Backtracking is necessary because of the possible choice of a suitable forward rule and match used for the translation of a particular source element. Hence, backtracking is performed if a transformation sequence terminates and

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

is not completed successfully because some parts of the source model have not been translated. This means that an execution of $MT$ requires backtracking if there are terminating TGT-sequences

$$\left(Att^F\left(G^S\right) \leftarrow \varnothing \rightarrow \varnothing\right) \overset{tr^*_{FT}}{\Longrightarrow} G'_n$$

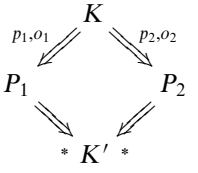
with

$$G'^S_n \neq Att^T\left(G^S\right).$$

The termination of a forward translation sequence means that the construction of this sequence ends at a graph to which no further forward translation rule is applicable. As we will show in Theorems 3.13 and 3.16, functional behaviour and the elimination of backtracking are closely related topics.

**Definition 3.1 (functional behaviour of model transformations).** Given a source DSL $\mathcal{L}_S \subseteq VL_S$, a model transformation $MT$ based on forward translation rules has *functional behaviour* if each execution of $MT$ starting at a source model $G^S \in \mathcal{L}_S$ leads to a unique target model $G^T \in VL_T$.

The standard way to analyse functional behaviour is to check whether the underlying transformation system is confluent, that is, all diverging derivation paths starting at the same model finally meet again. According to Newman's Lemma (Newman 1942), confluence can be shown by proving local confluence and ensuring termination. More precisely, local confluence means that whenever a graph $K$ can be transformed in one step into two graphs $P_1$ and $P_2$, these graphs can be transformed into a graph $K'$, as in




Local confluence can be shown by checking the confluence of all *critical pairs*

$$(P_1 \Leftarrow K \Rightarrow P_2),$$

which represent the minimal objects where a confluence conflict may occur. A critical pair describes a minimal conflict, where minimality means that only overlappings of the rule components are considered for graph $K$. The technique is based on two results (Ehrig *et al.* 2006). On the one hand, the *completeness* of critical pairs implies that for every confluence conflict given by a pair of diverging transformation steps $(G_1 \Leftarrow G \Rightarrow G_2)$, there is a critical pair

$$(P_1 \Leftarrow K \Rightarrow P_2)$$

that can be embedded into $(G_1 \Leftarrow G \Rightarrow G_2)$. On the other hand, the transformations

$$\left(P_1 \overset{*}{\Rightarrow} K' \overset{*}{\Leftarrow} P_2\right)$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

obtained by confluence of the critical pair can be embedded into the transformations

$$\left(G_1 \overset{*}{\Rightarrow} G' \overset{*}{\Leftarrow} G_2\right)$$

that solve the original confluence conflict.

However, as shown in Plump (1993) and Plump (2005), the confluence of critical pairs is not sufficient for this, and we need a slightly stronger version, called strict confluence, which adds the requirement that the preserved elements of the given steps are preserved in the merging steps. This means that elements that are not deleted by one of the original transformations steps must be preserved by the additional transformations that lead to confluence to ensure the applicability of the rules in the bigger context. This is necessary because, when extending such a transformation, a preserved node may be adjacent to an edge such that the deletion of this node would lead to dangling edges, that is, the additional transformations are not applicable in the larger context. This result is also valid for typed attributed graph transformation systems (Ehrig *et al.* 2006; Lambers 2009), and we will apply it to show functional behaviour of model transformations.

Furthermore, in the presence of NACs, we also have to ensure that the NAC-consistency of the merging steps is implied by the NAC-consistent diverging steps of the critical pair. Again, this property ensures that the confluent transformations of the critical pair can be embedded into a larger context. NAC-consistency of an embedding $k : G \to G'$ for a transformation step

$$G \overset{p,m}{\Longrightarrow} H$$

implies that there is a transformation step

$$G' \overset{p,m'}{\Longrightarrow} H'$$

with

$$m' = k \circ m$$

that satisfies the NACs of $p$. NAC-consistency for a transformation sequence can be checked by constructing the concurrent rule of the sequence (Lambers 2009), which combines the relevant NACs in a suitable way. If an embedding morphism fulfils the NACs of the original rules, the critical pair can be embedded into the larger context. To ensure the embedding of the additional transformations, the embedding morphism has to fulfil all occurring NACs to ensure the applicability of the transformation, otherwise we may have an embedding of a critical pair that is not confluent.

We will begin by recalling the basic notions for confluence of critical pairs according to Ehrig *et al.* (2006) and Lambers (2009).

**Definition 3.2 (NAC-strict confluence of critical pairs).** A critical pair

$$CP = \left(P_1 \overset{p_1,o_1}{\Longleftarrow} K \overset{p_2,o_2}{\Longrightarrow} P_2\right)$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

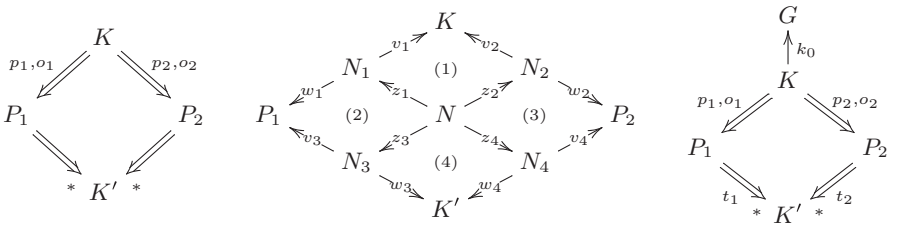



Fig. 10. NAC-strict confluence

is said to be *strictly confluent* if we have:

(1) *Confluence*:
The critical pair is confluent, that is, there are transformations

$$t_1 : P_1 \overset{*}{\Rightarrow} K'$$

$$t_2 : P_2 \overset{*}{\Rightarrow} K'$$

with derived spans

$$der(t_i) = \left(P_i \xleftarrow{v_{i+2}} N_{i+2} \xrightarrow{w_{i+2}} K'\right)$$

for $i = 1, 2$.

(2) *Strictness*:
Let

$$der\left(K \overset{p_i,o_i}{\Longrightarrow} P_i\right) = \left(K \xleftarrow{v_i} N_i \xrightarrow{w_i} P_i\right)$$

for $i = 1, 2$, and let $N$ be the pullback object of the pullback (1) in Figure 10. Then, there are morphisms $z_3$ and $z_4$ such that (2), (3) and (4) in Figure 10 commute.

(3) *NAC-consistency*:
For every injective morphism $k_0 : K \to G$ that is NAC consistent with respect to

$$K \overset{p_1,o_1}{\Longrightarrow} P_1$$

$$K \overset{p_2,o_2}{\Longrightarrow} P_2$$

in Figure 10, $k_0$ is also NAC consistent with respect to $t_1$ and $t_2$.

However, while it is quite easy to ensure the termination of model transformations based on forward rules or forward translation rules by checking that all *TGG*-triple rules are creating on the source component, this is not the case for local confluence. In fact, the system of forward translation rules of our *CD2RDBM* case study is not locally confluent, but we will show in Example 3.17 that the model transformation has functional behaviour. Indeed, the functional behaviour of a model transformation does not require general confluence of the underlying system of operational rules. Confluence only needs to be ensured for the transformation paths that lead to completely translated models. More precisely, derivation paths leading to a point for backtracking do not influence the functional behaviour. For this reason, we introduce so-called *filter NACs* that extend the model transformation rules in order to avoid misleading paths that cause backtracking,

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

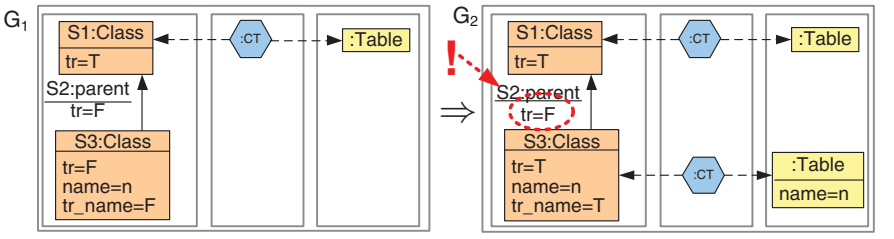



Fig. 11. (Colour online) Step $G_1 \overset{Class2Table_{FT}}{\Longrightarrow} G_2$ with misleading graph $G_2$

and in this way the backtracking for the extended system is substantially reduced. Using Fact 3.9, we will ensure that the overall behaviour of the model transformation with respect to the model transformation relation is still preserved. As a first important result, we will show in Theorem 3.13 that the functional behaviour of a model transformation is ensured by termination and strict confluence of all significant critical pairs of the system of forward translation rules enriched by filter NACs, where significant critical pairs are a subset of all critical pairs. Furthermore, we are able to characterise the strong functional behaviour of a terminating model transformation based on forward translation rules with filter NACs in Theorem 3.16 by the condition that there is no significant critical pair at all. Compared with functional behaviour, we can also ensure by strong functional behaviour that the model transformation sequences are unique up to switch equivalence.

Hence, the addition of filter NACs has two advantages. On the one hand, the analysis of functional behaviour is improved because the possible conflicts between the transformation rules are reduced, and we will show in this section that filter NACs allow us to verify functional behaviour for our *CD2RDBM* case study. On the other hand, filter NACs improve the efficiency of the execution by cutting off possible backtracking paths. Filter NACs are based on the following notion of misleading graphs, which can be viewed as the model fragments that are responsible for the backtracking of a model transformation.

**Definition 3.3 (translatable and misleading graphs).** A triple graph with translation attributes $G$ is *translatable* if there is a transformation sequence

$$G \overset{tr^*_{FT}}{\Longrightarrow} H$$

using forward translation rules such that $H$ is completely translated (see Definition 2.18). 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 3.4 (misleading graph).** Consider the transformation step shown in Figure 11. The resulting graph $G_2$ is misleading according to Definition 3.3 because the edge $S2$ is labelled with a translation attribute set to '**F**', but there is no rule that may change this attribute in any bigger context at any later stage of the transformation. The only rule that changes the translation attribute of a 'parent'-edge is '$Subclass2Table_{FT}$', but it requires that the source node $S3$ is labelled with a translation attribute set to '**F**'. However, forward

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

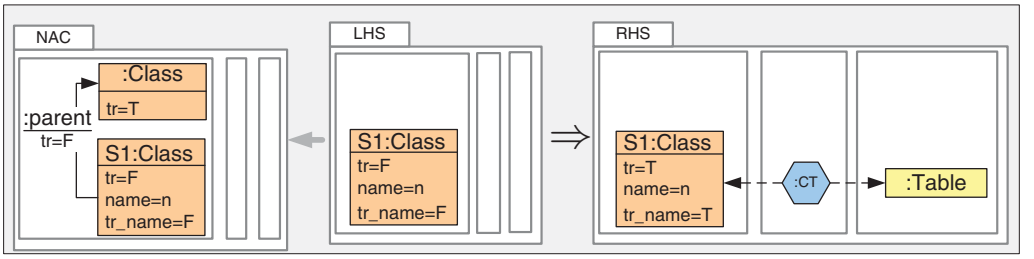



Fig. 12. (Colour online) A forward translation rule with filter NAC: $Class2Table_{FN}$

translation rules do not modify translation attributes if they are set to '**T**' already and do not change the structure of the source component.

**Definition 3.5 (filter NAC).** A filter NAC $n$ for a forward translation rule

$$tr_{FT} : L_{FT} \leftarrow K_{FT} \rightarrow R_{FT}$$

is given by a morphism $n : L_{FT} \rightarrow N$, such that there is a TGT step

$$N \overset{tr_{FT},n}{\Longrightarrow} M$$

with $M$ being misleading. The extension of $tr_{FT}$ by some set of filter NACs is called a forward translation rule $tr_{FN}$ with filter NACs.

**Example 3.6 (forward translation rule with filter NACs).** The $Class2Table_{FT}$ rule is extended by a filter NAC in Figure 12, which is obtained from the graph $G_1$ of the transformation step

$$G_1 \overset{Class2Table_{FT}}{\Longrightarrow} G_2$$

in Figure 11, where $G_2$ is misleading according to Example 3.4. In Fact 3.7, we will show how such filter NACs can be generated automatically. In Example 3.17, we will extend the rule by a further similar filter NAC with '$tr = \mathbf{T}$' for node '$S1$'.

A direct construction of filter NACs according to Definition 3.5 would be inefficient because the size of the considered graphs to be checked is unbounded. For this reason, we will now present some efficient techniques that support the generation of filter NACs and allow us to bound the size without losing generality. We will first present an automated technique for a subset of filter NACs and then an interactive generation technique leading to a much larger set of filter NACs. The first procedure in Fact 3.7 is based on a sufficient criterion for checking the misleading property. For our running example, this automated generation leads to the filter NAC shown in Figure 12 for the $Class2Table_{FT}$ rule for an incoming edge of type '*parent*'.

**Fact 3.7 (automated 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

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

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 $TR$ that creates an edge $(e : T_e)$ also creates its source node.
- If we extend $L_{FT}$ to $N$ by adding an outgoing edge $(e : T_e)$ at $x$ together with a target node and add a translation attribute for $e$ with value **F**, then 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*:
This is just the dual case of the above, but this time for an incoming edge $(e : T_e)$.

— $TR_{FN}$ is the extension of $TR_{FT}$ by all filter NACs constructed above.

*Proof.* See Appendix B. □

The following interactive technique for deriving filter NACs is based on the generation of critical pairs that define conflicts of rule applications in a minimal context. By the completeness of critical pairs (Ehrig *et al.* 2006, Lemma 6.22), we know that for each pair of parallel dependent transformation steps, there is a critical pair that can be embedded. If a critical pair

$$P_1 \xleftarrow{tr_{1,FT}} K \xrightarrow{tr_{2,FT}} P_2$$

contains a misleading graph $P_1$, we use the overlapping graph $K$ as a filter NAC of the rule $tr_{1,FT}$. However, checking the misleading property needs manual interaction, though in some cases, the manual results from identifying misleading graphs can be reused for more general static conditions. Indeed, the conditions used in Fact 3.7 were inspired by our application of the interactive method to our case study. Moreover, we are currently working on a technique that uses a sufficient criterion to check the misleading property automatically, and we are confident that this approach will provide a powerful generation technique.

**Fact 3.8 (interactive generation of filter NACs).** Given a set of forward translation rules, 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

$$K \xrightarrow{tr_{1,FN},m_1} P_1$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

violates the filter NAC. Hence, the critical pair for $tr_{1,FT}$ and $tr_{2,FT}$ is no longer a critical pair for $tr_{1,FN}$ and $tr_{2,FT}$. However, 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.* See Appendix B. □

Using the flattening construction presented in Ehrig *et al.* (2008), we can derive an equivalent plain graph transformation system from the system of forward translation rules. Since the system of forward translation rules ensures source consistency for complete transformation sequences by construction, the derived flattened grammar also ensures source consistency for complete transformation sequences. For this reason, we do not need to extend the analysis techniques for critical pairs and can use the AGG critical pair analysis engine (AGG 2011).

For our *CD2RDBM* case study, the interactive generation terminates after the second round, which is typical for practical applications because the number of already translated elements in the new occurring critical pairs usually decreases. Furthermore, several NACs can be combined if they only differ on some translation attributes. According to Fact 3.9, filter NACs do not change the behaviour of model transformations, and their only effect is to filter out derivation paths that would lead to misleading graphs, that is, to backtracking for the computation of the model transformation sequence. This means that the filter NACs filter out backtracking paths.

**Fact 3.9 (equivalence of transformations with filter NACs).** Given a triple graph grammar

$$TGG = (TG, \emptyset, TR)$$

with forward translation rules $TR_{FT}$ and filter NACs leading to $TR_{FN}$. Let

$$G_0 = \left(G^S \leftarrow \emptyset \rightarrow \emptyset\right)$$

be a triple graph typed over $TG$ and

$$G_0' = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \emptyset \rightarrow \emptyset\right).$$

Then the following are equivalent for almost injective matches:

(1) There is a complete TGT-sequence $G_0' \overset{tr_{FT}^*, m_{FT}^*}{\Longrightarrow} G'$ using $TR_{FT}$.

(2) There is a complete TGT-sequence $G_0' \overset{tr_{FN}^*, m_{FT}^*}{\Longrightarrow} G'$ using $TR_{FN}$.

*Proof.* See Appendix B. □

According to Hermann *et al.* (2010a, Theorem 1), we have the following Fact 3.11 for the termination of a system of forward translation rules according to Definition 3.10.

**Definition 3.10 (termination).** A system of forward translation rules $TR_{FT}$ is terminating if each transformation sequence using $TR_{FT}$ is terminating, that is, the sequence ends at a graph to which no further forward translation rule can be applied.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Fact 3.11 (termination).** Given $TR_{FN}$ and $TR_{FT}$ as in Fact 3.9, $TR_{FN}$ is terminating if $TR_{FT}$ is terminating. A sufficient condition for termination of $TR_{FT}$ is that all graphs are finite on the graph part and each rule modifies at least one translation attribute from false to true. Termination of $TR_{FN}$ with strict confluence of critical pairs implies unique normal forms by the local confluence theorem in Lambers (2009).

In order to analyse functional behaviour, we generate the critical pairs for the system of forward translation rules and show by Theorem 3.13 that strict confluence of 'significant' critical pairs ensures functional behaviour. A critical pair is significant if it can be embedded into two transformation sequences using forward translation rules that start at the same source model $G^S$, which belongs to the source domain specific language $\mathcal{L}_S$. This implies that a critical pair containing a misleading graph is automatically not significant. For this reason, some of the non-significant critical pairs can already be eliminated using the automatic and interactive techniques for generating filter NACs presented in Facts 3.7 and 3.8.

**Definition 3.12 (significant critical pair).** A critical pair

$$\left(P_1 \xleftarrow{tr_{1,FN}} K \xrightarrow{tr_{2,FN}} P_2\right)$$

for a set of forward translation rules with filter NACs $TR_{FN}$ is said to be significant if it can be embedded into a parallel dependent pair

$$\left(G_1' \xleftarrow{tr_{1,FN}} G' \xrightarrow{tr_{2,FN}} G_2'\right)$$

such that there is $G^S \in \mathcal{L}_S \subseteq VL_S$ and $G_0' \xRightarrow{tr_{FN}^*} G'$ with

$$G_0' = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \emptyset \rightarrow \emptyset\right).$$

$$G_0' \xRightarrow{tr_{FN}^*} G' \begin{array}{l} \xRightarrow{tr_{1,FN}} G1' \\ \xRightarrow{tr_{2,FN}} G_2' \end{array}$$

**Theorem 3.13 (functional behaviour).** Let $MT_{FT}$ be a model transformation based on forward translation rules $TR_{FT}$ with model transformation relation $MTR_{FT}$ and source DSL $\mathcal{L}_S$, and let $TR_{FN}$ extend $TR_{FT}$ with filter NACs such that $TR_{FN}$ is terminating and all significant critical pairs are strictly confluent. Then $MT_{FT}$ has functional behaviour. Moreover, the model transformation $MT_{FN}$ based on $TR_{FN}$ does not require backtracking and $MT_{FN}$ defines the same model transformation relation, that is,

$$MTR_{FN} = MTR_{FT}.$$

*Proof.* For the functional behaviour of the model transformation, we have to show that each source model $G^S \in \mathcal{L}_S$ 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, $G^T \in VL_T$.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

For $G^S \in \mathcal{L}_S \subseteq VL_S$, we have, by the definition of $VL$, that there is a $G^T \in VL_T$ and a TGT-sequence

$$\varnothing \overset{tr^*}{\Longrightarrow} \left(G^S \leftarrow G^C \rightarrow G^T\right)$$

using $TR$, and from the decomposition theorem with NACs in Ehrig *et al.* (2009b), we obtain a match consistent TGT-sequence

$$\varnothing \overset{tr_S^*}{\Longrightarrow} \left(G^S \leftarrow \varnothing \rightarrow \varnothing\right) \overset{tr_F^*}{\Longrightarrow} \left(G^S \leftarrow G^C \rightarrow G^T\right),$$

and, by Fact 2.21, a complete TGT-sequence

$$G_0' = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \varnothing \rightarrow \varnothing\right) \overset{tr_{FT}^*}{\Longrightarrow} \left(Att^{\mathbf{T}}\left(G^S\right) \leftarrow G^C \rightarrow G^T\right) = G'.$$

This means that

$$\left(G^S, G_0' \overset{tr_{FT}^*}{\Longrightarrow} G', G^T\right)$$

is a model transformation sequence based on $TR_{FT}$. We now assume that we also have a complete forward translation sequence

$$G_0' = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \varnothing \rightarrow \varnothing\right) \overset{\overline{tr}_{FT}^*}{\Longrightarrow} \left(Att^{\mathbf{T}}\left(G^S\right) \leftarrow \overline{G^C} \rightarrow \overline{G^T}\right) = \overline{G}'.$$

By Fact 3.9, we also have the complete TGT-sequences

$$\left(G^S, G_0' \overset{tr_{FN}^*}{\Longrightarrow} G', G^T\right)$$

$$G_0' \overset{tr_{FN}^*}{\Longrightarrow} G'$$

$$G_0' \overset{tr_{FN}^*}{\Longrightarrow} \overline{G}'.$$

Using the assumption that $TR_{FN}$ is terminating and all significant critical pairs are strictly confluent, we will now show that all diverging transformation sequences can be merged again. Consider the possible transformation sequences starting at $G_0'$ (which form a graph of transformation steps) and two diverging steps

$$\left(G_{i+1}' \overset{p_1,m_1}{\Longleftarrow} G_i' \overset{p_2,m_2}{\Longrightarrow} G_{i+1}''\right).$$

If they are parallel independent, we can apply the local Church–Rosser theorem (LCR) (Lambers 2009) and derive the merging steps

$$\left(G_{i+1}' \overset{p_2,m_2'}{\Longrightarrow} H \overset{p_1,m_1'}{\Longleftarrow} G_{i+1}''\right).$$

If they are parallel dependent diverging steps, we know by the completeness of critical pairs (Lambers 2009, Theorem 3.7.6) that there is a critical pair, and by Definition 3.12, we know that this pair is significant because we consider transformations sequences starting at $G_0'$. This pair is strictly confluent by assumption, so these steps can be merged again. Any new diverging situation can now be merged by either LCR for parallel independent steps or by strict confluence of critical pairs for parallel dependent steps. By assumption, the system is terminating. In combination, this implies that $G' \cong \overline{G}'$, and thus $G^T \cong \overline{G}^T$.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

Backtracking is not required because termination of $TR_{FN}$ with strict confluence of significant critical pairs implies unique normal forms as shown above. Therefore, any terminating TGT-sequence

$$\left(Att^F\left(G^S\right) \leftarrow \emptyset \rightarrow \emptyset\right) \overset{tr^*_{FN}}{\Longrightarrow} G'_n$$

leads to a unique $G'_n$ up to isomorphism, and by correctness and completeness (Theorem 2.10 and Fact 2.21), we have that

$$G'^S_n = Att^T\left(G^S\right).$$

The same holds for the model transformation relation because we have the equivalence of the model transformation sequences by Fact 3.9. □

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 Theorem 3.13 that the corresponding system $TR_{FT}$ without filter NACs has functional behaviour. Moreover, from an efficiency point of view, the set of rules should be compact in order to minimise the effort of pattern matching. In the optimal case, the rule set ensures that each transformation sequence of the model transformation is itself unique up to switch equivalence, meaning that it is unique up to the order of sequentially independent steps. For this reason, we introduce the notion of strong functional behaviour with respect to a given source domain specific language (DSL) $\mathcal{L}_S$. Note that two transformation sequences are said to be switch equivalent if they can be obtained from each other by switching consecutive sequentially independent transformation steps, which is possible according to the Local Church–Rosser Theorem (Ehrig *et al.* 2006; Lambers 2009).

**Definition 3.14 (strong functional behaviour of model transformations).** A model transformation based on forward translation rules $TR_{FN}$ with filter NACs and the source DSL $\mathcal{L}_S \subseteq VL_S$ has *strong functional behaviour* if for each $G^S \in \mathcal{L}_S$, there is a $G^T \in VL_T$ and a model transformation sequence

$$\left(G^S, G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'_n, G^T\right)$$

based on forward translation rules, and:

— any partial TGT-sequence

$$G'_0 \overset{tr^{i,*}_{FN}}{\Longrightarrow} G'_i$$

terminates, that is, there are finitely many extended sequences

$$G'_0 \overset{tr^{i,*}_{FN}}{\Longrightarrow} G'_i \overset{tr^{j,*}_{FN}}{\Longrightarrow} G'_j;$$

and

— each pair of TGT-sequences

$$G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'_n$$

$$G'_0 \overset{\overline{tr}^*_{FN}}{\Longrightarrow} \overline{G}'_m$$

with completely translated graphs $G'_n$ and $\overline{G}'_m$ are switch equivalent up to isomorphism.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Remark 3.15 (strong functional behaviour).**

(1) The fact that sequences are terminating means that there is no longer any applicable rule in $TR_{FN}$. However, it is not required that the sequences are complete, that is, 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

$$G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'_n$$
$$G'_0 \overset{\overline{tr}^*_{FN}}{\Longrightarrow} \overline{G}'_m$$

are terminating TGT-sequences.
(3) Two sequences

$$t1 : G_0 \Rightarrow^* G_1$$
$$t2 : G_0 \Rightarrow^* G_2$$

are said to be 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 (Lambers 2009). The sequences $t1$ and $t2$ are said to be switch equivalent up to isomorphism if

$$t1 : G_0 \Rightarrow^* G_1$$

has an isomorphic sequence

$$t1' : G_0 \Rightarrow^* G_2$$

(using the same sequence of rules) with $i : G_1 \xrightarrow{\sim} G_2$, written

$$trace(t1') = i \circ trace(t1),$$

such that $t1' \approx t2$. In particular, this means that the rule sequence in $t2$ is a permutation of the rule sequence in $t1$.

We will now give the third main result of the current paper, which shows that strong functional behaviour of model transformations based on forward translation rules with filter NACs can be completely characterised by the absence of significant critical pairs.

**Theorem 3.16 (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, and thus polynomial time complexity, if and only if $TR_{FN}$ has no significant critical pair.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

*Proof.*

($\Leftarrow$) We assume that $TR_{FN}$ has no significant critical pair. As in the proof of Theorem 3.13, we obtain for each $G^S \in \mathcal{L}_S$ a $G^T \in VL_T$ and a complete TGT-sequence $G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'$ and a model transformation

$$\left(G^S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G', G^T\right)$$

based on $TR_{FT}$ underlying $TR_{FN}$. By Fact 3.9, we also have a complete TGT-sequence $G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'$, and hence also a model transformation

$$\left(G^S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G', G^T\right)$$

based on $TR_{FT}$ underlying $TR_{FN}$. In order to show strong functional behaviour let

$$G'_0 \overset{tr^*_{FN}}{\Longrightarrow} G'_n$$
$$G'_0 \overset{\overline{tr}^*_{FN}}{\Longrightarrow} \overline{G}'_m$$

be two terminating TGT-sequences with $m, n \geqslant 1$. We have to show that they are switch equivalent up to isomorphism. We will 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

$$t1 : G'_0 \overset{tr_{FN},m}{\Longrightarrow} G'_1$$
$$\overline{t1} : G'_0 \overset{\overline{tr}_{FN},\overline{m}}{\Longrightarrow} \overline{G}'_1.$$

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 $t1 \approx i \circ \overline{t1}$. Otherwise, $t1$ and $\overline{t1}$ are parallel independent because otherwise we would have a significant critical pair by the completeness of critical pairs in Lambers (2009). By the local Church–Rosser theorem (Lambers 2009), we have

$$t2 : G'_1 \overset{\overline{tr}_{FN}}{\Longrightarrow} G'_2$$
$$\overline{t2} : \overline{G}'_1 \overset{tr_{FN}}{\Longrightarrow} G'_2$$

such that

$$t2 \circ t1 \approx \overline{t2} \circ \overline{t1} : G'_0 \Rightarrow^* G'_2.$$

We now assume that for

$$t1 : G'_0 \Rightarrow^* G'_{n-1}$$
$$\overline{t1} : G'_0 \Rightarrow^* \overline{G}'_m,$$

we have extensions

$$t2 : G'_{n-1} \Rightarrow^* H$$
$$\overline{t2} : \overline{G}'_m \Rightarrow^* H,$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

such that

$$t2 \circ t1 \approx \overline{t2} \circ \overline{t1}.$$

$$\begin{array}{ccccc} G'_0 & \overset{t1}{\Longrightarrow}{}^{*} & G'_{n-1} & \overset{t}{\Longrightarrow} & G'_n \\ {\scriptstyle \overline{t1}}\Big\downarrow{\scriptstyle *} & & \Big\downarrow{\scriptstyle t2}{\scriptstyle *} & & \Big\downarrow{\scriptstyle t3}{\scriptstyle *} \\ \overline{G}'_m & \underset{\overline{t2}}{\Longrightarrow}{}^{*} & H & \underset{\overline{t3}}{\Longrightarrow}{}^{*} & K \end{array}$$

For a step $t : G'_{n-1} \Rightarrow G'_n$, we now have to show that $t \circ t1$ and $\overline{t1}$ can be extended to switch-equivalent sequences. By the induction hypothesis and the definition of significant critical pairs, $t$ and $t2$ can also 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

$$t3 \circ t \circ t1 \approx \overline{t3} \circ \overline{t2} \circ \overline{t1} : G'_0 \Rightarrow^{*} K,$$

which completes the induction proof.
We now use the fact 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

$$G'_0 \overset{tr^{*}_{FN}}{\Longrightarrow} G'_n$$
$$G'_0 \overset{\overline{tr}^{*}_{FN}}{\Longrightarrow} \overline{G}'_m$$

are switch equivalent up to isomorphism.

($\Rightarrow$) We will prove this direction by showing a contradiction. So we assume that $TR_{FN}$ has strong functional behaviour and that $TR_{FN}$ has a significant critical pair.
Let

$$P_1 \overset{tr_{1,FN}}{\Longleftarrow} K \overset{tr_{2,FN}}{\Longrightarrow} P_2$$

be the significant critical pair, which 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 $G^S \in \mathcal{L}_S$ with $G'_0 \overset{tr^{*}_{FN}}{\Longrightarrow} G'$ and

$$G'_0 = \left(Att^{\mathbf{F}}\left(G^S\right) \leftarrow \emptyset \rightarrow \emptyset\right).$$

Since $TR_{FN}$ is terminating, we have terminating sequences

$$G_1 \Rightarrow^{*} G_{1n}$$
$$G_2 \Rightarrow^{*} G_{2m}$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

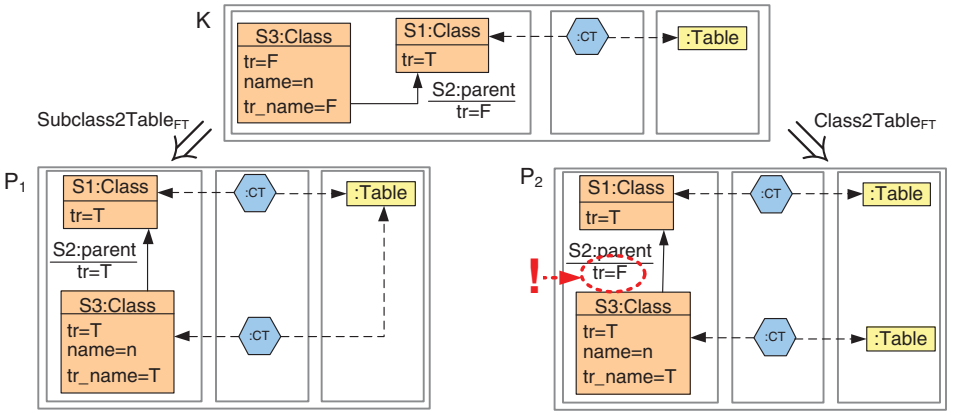



Fig. 13. (Colour online) Critical pair for the rules $Subclass2Table_{FT}$ and $Class2Table_{FT}$

through $TR_{FN}$. By composition, we have the following terminating TGT-sequences

$$G_0' \overset{tr_{FN}}{\Longrightarrow} G' \overset{tr_{1,FN}}{\Longrightarrow} G_1 \Rightarrow^* G_{1n} \tag{1}$$

$$G_0' \overset{tr_{FN}}{\Longrightarrow} G' \overset{tr_{2,FN}}{\Longrightarrow} G_2 \Rightarrow^* G_{2m}. \tag{2}$$

Since $TR_{FN}$ has strong functional behaviour, both are switch equivalent up to isomorphism. For simplicity, we assume $G_{1n} = G_{2m}$ instead of $G_{1n} \cong G_{2m}$. This implies $n = m$ and

$$G' \overset{tr_{1,FN}}{\Longrightarrow} G_1 \Rightarrow^* G_{1n}$$

is switch equivalent to

$$G' \overset{tr_{2,FN}}{\Longrightarrow} G_2 \Rightarrow^* G_{1n}.$$

This means $tr_{2,FN}$ occurs in $G_1 \Rightarrow^* G_{1n}$ and can be shifted in

$$G' \overset{tr_{1,FN}}{\Longrightarrow} G_1 \Rightarrow^* G_{1n},$$

to give

$$G' \overset{tr_{2,FN}}{\Longrightarrow} G_2 \Rightarrow^* G_{1n}.$$

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

The fact that strong functional behaviour implies that backtracking is not required is a direct consequence of Theorem 3.13: since we have no significant critical pair, all of them are strictly confluent. □

**Example 3.17 (functional and strong functional behaviour).** In this example, we analyse the functional behaviour of the *CD2RDBM* model transformation. By Fact 3.11, *CD2RDBM* is terminating because all *TGG*-triple rules are creating in the source component. In order to analyse local confluence, we use the AGG tool (AGG 2011) for the generation of

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

critical pairs. The set of derived forward translation rules from the rules $TR$ in Figures 2 and 4 is given by

$$TR_{FT} = \{Class2Table_{FT}, Subclass2Table_{FT}, Attr2Column_{FT}, PrimaryAttr2Column_{FT}, Association2ForeignKey_{FT}\}.$$

We now replace the $Class2Table_{FT}$ forward translation rule by the extended rule with filter NACs, $Class2Table_{FN}$, as shown in Figure 12, and additionally extend it by a further filter NAC obtained by the automated generation according to Fact 3.7. We used AGG (version 2.0) to generate the critical pairs. AGG detects three critical pairs for conflicts of the '*PrimaryAttr2Column*' rule with itself. The corresponding overlapping graphs $K$ of the critical pairs contain two primary attribute nodes, which belong to classes that are connected to the same table. This implies that the resulting graphs $P_1$ and $P_2$ of each critical pair ($P_1 \Leftarrow K \Rightarrow P_2$) are misleading because the remaining untranslated primary attribute of the first two cannot be translated in any bigger context because of the source NAC of the rule, and because no other rule translates a primary attribute. Hence, all critical pairs lead to additional filter NACs by the interactive generation of filter NACs in Fact 3.8. For the resulting system of forward translation rules with filter NACs, AGG does not generate any critical pair. Thus, we can apply Theorem 3.16 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 Theorem 3.13, we can conclude that the model transformation based on the forward translation rules $TR_{FT}$ without filter NACs has *functional behaviour*. As an example, Figure 9 shows the resulting triple graph of a model transformation starting with the class diagram $G^S$.

### 3.2. *Information preservation*

Model transformations are information preserving if their corresponding backward transformations can be used to derive parts of the given source model from a target model that was derived through a forward transformation. In fact, several TGG tools do not support backtracking and use optimisations in a way that means they cannot ensure completeness. This implies that for some target models, the execution of backward transformations may stop without creating a valid source model (Giese *et al.* 2010; Schürr and Klar 2008; Klar *et al.* 2010). This section provides results for analysing and ensuring information preservation for TGG model transformations according to Section 2. In particular, we analyse whether and how a source model can be reconstructed from the computed target model.

To do this, we distinguish forward and backward model transformations. Interestingly, it turns out that complete information preservation, that is, the complete reconstruction of the source model, is ensured by the functional behaviour of the backward model transformation. We will present the techniques for model transformations based on forward rules. According to the equivalence result in Fact 2.21, we also know that these techniques provide the same results for model transformations based on forward

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

translation rules. Moreover, because of the symmetric definition of TGGs, the results can be applied dually for backward model transformations.

**Definition 3.18 (information preserving model transformation).** A forward model transformation based on forward rules is *information preserving* if for each forward model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right),$$

there is a backward model transformation sequence

$$\left(G^T, G_0' \overset{tr_B'^*}{\Longrightarrow} G_m', G'^S\right)$$

with $G^S = G'^S$, that is, the source model $G^S$ can be reconstructed from the resulting target model $G^T$ using a target-consistent backward transformation sequence.

The following theorem shows that model transformations based on forward rules are information preserving.

**Theorem 3.19 (information preserving model transformation).** Each forward model transformation based on forward rules is information preserving.

*Proof.* We assume a set of triple rules $TR$ with derived forward rule $TR_F$ and backward rules $TR_B$. By Fact 2.9 and Remark 2.11 applied to the source-consistent forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ through $TR_F$, we can derive the target-consistent backward transformation

$$G_0' = \left(G^T \leftarrow \varnothing \rightarrow \varnothing\right) \overset{tr_B^*}{\Longrightarrow} G_n$$

using $TR_B$ with $G_n^S = G^S$. This means that we have a backward model transformation sequence

$$\left(G^T, G_0' \overset{tr_B^*}{\Longrightarrow} G_n, G'^S\right)$$

with $G^S = G'^S$. □

**Example 3.20 (information preserving model transformation *CD2RDBM*).** The *CD2RDBM* model transformation is information preserving because it consists of model transformation sequences based on forward rules, which ensure source consistency of the forward sequences by definition. Hence, the source model $G^S$ of the triple graph in Figure 9 can be reconstructed by a target-consistent backward transformation sequence starting at the model

$$G_0' = \left(\varnothing \leftarrow \varnothing \rightarrow G^T\right).$$

However, there are several possible target-consistent backward transformation sequences starting at $G_0'$ because the $Subclass2Table_B$ rule can be applied arbitrarily often without having any influence on the target consistency since the rule is identical on the target component. This means that the inheritance information within a class diagram has no explicit counterpart within a relational data base model.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

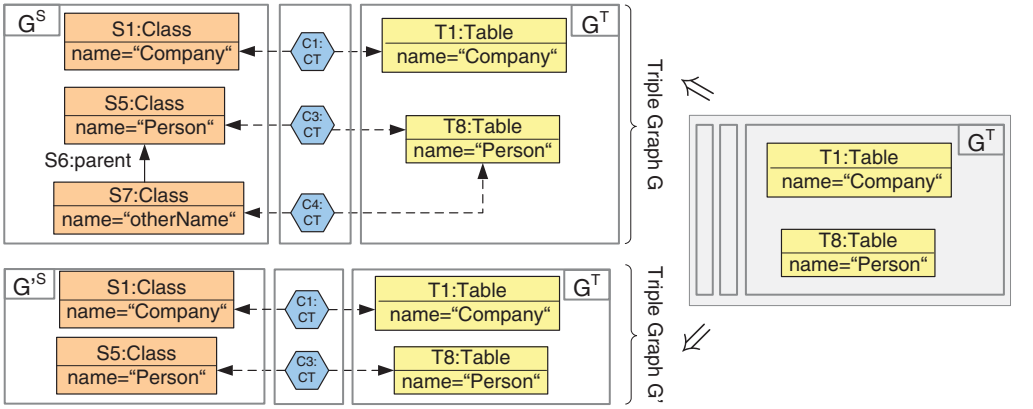



Fig. 14. (Colour online) Two possible target-consistent backward transformations

There are many possible target-consistent backward transformation sequences for the same derived target model $G^T$ – two of them are presented in Figure 14. The source model $G^S$ can be transformed into

$$G = \left(G^S \leftarrow G^C \rightarrow G^T\right).$$

However, starting with $G^T$, both of the backward transformation sequences shown are possible and target consistent, but the resulting source graphs $G^S$ and $G'^S$ differ with respect to the class node $S7$ and the edge $S6$ in $G^S$. Hence, some information about $G^S$ cannot be reconstructed uniquely and is thus partially lost in the target model $G^T$.

According to Theorem 3.19, each model transformation based on forward rules is information preserving. However, in general, the reconstruction of a corresponding source model from a derived target model is not unique. In order to ensure uniqueness of the reconstruction, we will now introduce the notion of complete information preservation. This stronger notion ensures that all information contained in a source model of a source domain specific language (DSL) can be reconstructed from the derived target model itself. More precisely, starting with the target model, each backward model transformation sequence will produce the original source model. This ensures that only one backward model transformation sequence has to be constructed. Intuitively, this means that the model transformation is invertible.

**Definition 3.21 (complete information preservation).** A forward model transformation with source DSL $\mathcal{L}_S$ is *completely information preserving* if it is information preserving and, given a source model $G^S \in \mathcal{L}_S$ and the resulting target model $G^T$ of a forward model transformation sequence, each partial backward transformation sequence starting with $G^T$ terminates and produces the given source model $G^S$ as the result.

We can verify complete information preservation by showing the functional behaviour of the corresponding backward model transformation with respect to the derived target models

$$\mathcal{L}'_T \subseteq MT(\mathcal{L}_S) \subseteq VL_T.$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Theorem 3.22 (completely information preserving model transformation).** A forward model transformation $MT$ is completely information preserving if the corresponding backward model transformation according to Remark 2.11 has functional behaviour with respect to the target language $\mathcal{L}'_T = MT(\mathcal{L}_S)$.

*Proof.* By Theorem 3.19, we know that $MT$ is information preserving. For a model transformation sequence

$$\left(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T\right).$$

We also know that $G^T \in VL_T$ by Theorem 2.10, and that

$$G^T \in \mathcal{L}'_T = MT(\mathcal{L}_S).$$

Using the functional behaviour of the corresponding backward model transformation according to Definition 3.1 for the language $\mathcal{L}'_T$, we know that for each model $H^T$, the backward model transformation yields a unique $H^S \in VL_S$. Therefore, each backward model transformation sequence

$$\left(G^T, G'_0 \overset{tr_B^*}{\Longrightarrow} G'_n, G'^S\right)$$

leads to a unique $G'^S \in VL_S$. Furthermore, there is a backward model transformation sequence

$$\left(G^T, G''_0 \overset{tr_B^*}{\Longrightarrow} G''_n, G^S\right)$$

by Theorem 3.19 implying $G^S \cong G'^S$, that is, the model transformation is completely information preserving. □

**Example 3.23 (complete information preservation).** The model transformation

$$MT_1 = CD2RDBM$$

is not completely information preserving. Consider, for example, the source model $G^S$ of Example 3.20 in Figure 14, where there are two possible backward model transformation sequences starting with the same derived target model $G^T$. This means that the backward model transformation has no functional behaviour with respect to

$$MT_1(\mathcal{L}_S) = MT(VL_S) = VL_T = \mathcal{L}_T.$$

However, we can also consider the inverse model transformation, that is, swapping the forward and backward direction leading to the model transformation

$$MT_2 = RDBM2CD$$

from relational data base models to class diagrams. In this case, the model transformation is completely information preserving, meaning that each relational data base model $M_{DB}$ can be transformed into a class diagram $M_{CD}$, and each data base model $M_{DB}$ can be completely and uniquely reconstructed from its derived class diagram $M_{CD}$. In other words, each class diagram resulting from a model transformation sequence of *RDBM2CD* contains

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

all the information present in the given data base model. According to Example 3.17, we know that the *CD2RDBM* model transformation has functional behaviour, so the backward model transformation of *RDBM2CD* has functional behaviour with respect to $VL_T$ being equal to the source language $VL_S$ of *CD2RDBM*. For this reason, we can apply Theorem 3.22 and get that *RDBM2CD* is completely information preserving. In particular, foreign keys are completely represented by associations, and primary keys by primary attributes. There is no structure within the data base model that is not explicitly represented within the class diagram.

## 4. Related work

TGGs have been successfully applied for model transformations for different purposes in a variety of domains (Guerra and de Lara 2006a; Guerra and de Lara 2006b; Kindler and Wagner 2007; Königs and Schürr 2006; Taentzer *et al.* 2005). The formal construction and analysis of model transformations based on TGGs was initiated in Ehrig *et al.* (2007), which analysed the information preservation of bidirectional model transformations. This work was continued in Ehrig *et al.* (2008), Ehrig and Prange (2008), Ehrig *et al.* (2009a), Ehrig *et al.* (2009b) and Hermann *et al.* (2010c), where model transformations based on TGGs are compared with those on plain graph grammars in Ehrig *et al.* (2008). TGGs with specification NACs were analysed in Ehrig *et al.* (2009b), and an efficient on-the-fly construction was introduced in Ehrig *et al.* (2009a). Pattern-based model-to-model transformations were introduced in de Lara and Guerra (2008), and the corresponding correctness, completeness and termination results given in Orejas *et al.* (2009). However, there results were limited in comparison with the results of the current paper.

Ehrig and Prange (2008) presented a first approach to analysing functional behaviour for restricted TGGs with distinguished kernels, but a more general approach based on forward translation rules was given in Hermann *et al.* (2010a) and Hermann *et al.* (2010c). The concept of forward translation rules was inspired by the translation algorithm in Schürr and Klar (2008), which uses a set for storing the elements that have been translated during a transformation. The results in the current paper for model transformations based on forward translation rules with specification and filter NACs are based on results in most of these papers. In particular, we extended their formal results by providing a less restrictive condition for functional behaviour and a sufficient condition for complete information preservation.

Ehrig *et al.* (2007) presented a similar case study based on forward rules, but without using NACs. The grammar with NACs in the current paper handles primary keys and foreign keys in a more appropriate way, and allows us to show strong functional behaviour.

Giese *et al.* (2010) presented a more restrictive condition for ensuring functional behaviour, which requires the complete absence of all critical pairs, while the condition given in the current paper only requires strict confluence of the significant critical pairs after optimising the rules by the automatic and interactive generation of filter NACs. In order to reduce backtracking, Klar *et al.* (2010) proposed a concept similar to the automatic generation of filter NACs in Section 3.1. The effect of the filter NACs is

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

specified directly within the transformation algorithm, though the complete elimination of backtracking cannot be ensured.

There are several other approaches to model transformations, and, in particular, bidirectional transformations, where the general idea is to define one direction of the model transformation and then get the backward direction for free. This is different to the TGG case, where we define the triple rules that build up the language of consistently integrated models, and from these triple rules we derive both forward and backward rules. Hidaka *et al.* (2010) defined a bidirectional language using structural recursion on graphs. Bohannon *et al.* (2006) introduced lenses, which are basically a pair of functions – 'get' for forward transformation and 'put' for backward transformation – obeying certain behavioural laws. Foster (2009) used these lenses to propose a bidirectional language for model transformations for updating views that ensures that changes are propagated back to the underlying model. Stevens (2008) discussed different important properties for model transformations including specification, composition and the maintenance of model transformations, as well as verification and correctness properties, and some corresponding laws for lenses. With their main focus on updates, lenses seem to be a particularly good fit for realising views, but their usefulness for general model transformations with very different source and target models, and the application to graphs and other high-level structures requires further analysis.

## 5. Conclusions

### 5.1. *Summary of main results*

In the current paper, we have studied model transformations based on triple graph grammars (TGGs) with negative application conditions (NACs) in order to improve the analysis and execution performance 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. Our first main result shows the correctness and completeness of model transformations for forward transformations, and also for forward translations, by combining Theorem 2.10 and Fact 2.21. Our second main result provides a sufficient condition for functional behaviour (Theorem 3.13) based on the analysis of critical pairs for forward translation rules with filter NACs. The generation of filter NACs improves the analysis of functional behaviour for model transformations based on critical pair analysis (using the AGG tool (AGG 2011)) by filtering out backtracking paths, and in this way, some critical pairs. 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 (Theorem 3.16). Moreover, Theorems 3.13 and 3.16 also show that the strict confluence of significant critical pairs ensures that backtracking is not required for the execution of the model transformation, which implies polynomial time complexity. Finally, we show in Theorem 3.19 that TGG-model transformations are information preserving, and in Theorem 3.22, that forward transformations are completely information preserving if the corresponding backward transformation has functional

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

behaviour. For our *CD2RDBM* case study, we have shown that backtracking can be eliminated and strong functional behaviour can be obtained by an automatic optimisation based on filter NACs. Moreover, this leads to complete information preservation for the derived backward transformation.

A major challenge in applying our main results on (strong) functional behaviour and complete information preservation is to find suitable filter NACs such that we have a minimal number of critical pairs. To this end, we have provided automated and interactive techniques for the generation of filter NACs (see Facts 3.7 and 3.8).

### 5.2. *Practical relevance*

In this section we discuss how the results in the current paper can be used to meet the 'Grand Research Challenge of the TGG Community' as formulated in Schürr and Klar (2008). The main aims are the 'consistency', 'completeness', 'expressiveness' and 'efficiency' of model transformations:

(1) *Consistency*:
Model transformations are consistent with respect to the given TGG if whenever the algorithm translates a source model $G_S$ into a target model $G_T$, there is a triple graph

$$G = (G_S \leftarrow G_C \rightarrow G_T) \in VL$$

generated by the TGG. This property is shown in Theorem 2.10.

(2) *Completeness and termination*:
Completeness means that the execution of the model transformation translates every source model $G_S \in VL_S$ – this property subsumes termination. Both properties are ensured for our construction by Theorem 2.10 and Fact 3.11 if triple rules are creating on the source part.

(3) *Efficiency*:
Model transformations should have polynomial space and time complexity with the exponent $k$ being the maximal number of elements of a rule. This property can be ensured if we can show that a model transformation does not require backtracking and the TGG has a finite set of triple rules, which are creating on the source component, have finitely many NACs and have rule components that are finite. In this case, each execution of a model transformation has at most $n$ steps, with $n$ being the number of structural elements of the source model. As discussed in Schürr and Klar (2008), the bound $k$ then ensures polynomial time complexity. Moreover, we provided sufficient criteria and techniques for reducing and eliminating backtracking in Section 3.1, where they are used to analyse functional behaviour. Large TGGs with more than 50 rules and big input models may still slow down the execution, though in a current project where we are using a TGG with 50 triple rules in the Henshin tool (Arendt *et al.* 2010), our experience has been that the execution time for transforming models with several hundred model elements is less than 2 seconds on a standard PC. Moreover, the AGG tool (AGG 2011) provides automated analysis components, which we used to analyse the functional behaviour and information preservation of the case study in the current paper, as described in Section 3.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

(4) *Expressiveness*:
Finally, features that are very important for solving practical problems, like NACs and attribute conditions, should be captured. Both NACs and attributes are handled by our approach. Moreover, we have partially extended the results to the case of more general application conditions in Golas *et al.* (2011) in the sense of Habel and Pennemann (2009).

Summing up, the approach to model transformations based on triple graph grammars we have presented provides an intuitive, expressive, formally well-founded and efficient framework for bidirectional model transformations, and we have produced powerful results for analysis and optimisation using filter NACs. According to the achievements listed above, our approach offers several important advantages compared with other existing approaches, such as Schürr and Klar (2008), Königs and Schürr (2006), Kindler and Wagner (2007), Giese and Wagner (2009) and Giese and Hildebrandt (2009), which are mainly focused on software engineering, and hence do not provide similar formal results. However, these approaches are generally very similar, and in fact stimulated the development of some of our constructions, so, with some modification efforts, it may be possible to transfer the results we have presented here to other related approaches.

### 5.3. *Future work*

In the current paper we have considered functional behaviour with respect to unique target models – the more general notion given in Schürr and Klar (2008) regarding some semantic equivalence of target models will be the subject of further extensions of our techniques. Moreover, we will study additional static conditions for eliminating misleading execution paths, and we will develop extensions to layered model transformations and amalgamated rules. Finally, we have already applied some of the results for model transformation developed in the current paper to model synchronisation based on TGGs in order to ensure correctness (Hermann *et al.* 2011). However, there are several further problems in model synchronisation that will require new results: for example, concerning a notion of information preservation for partially related domain languages.

## Appendix A. Category of typed attributed garaphs

Typed attributed triple graphs are based on the underlying category of typed attributed graphs $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$, which is given by the slice category $(\mathbf{AGraph} \downarrow ATG, \mathcal{M})$ of directed attributed graphs over a type graph $ATG$.

In this appendix we review the main constructions for the $\mathcal{M}$-adhesive category of typed attributed graphs $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ according to Ehrig *et al.* (2006).

An attributed graph consists of an extended directed graph for the structural part, called the $E$-graph, together with an algebra for the specification of the carrier sets of the value nodes. An E-graph extends a directed graph by additional attribute value nodes and edges for the attribution of structural nodes and edges.

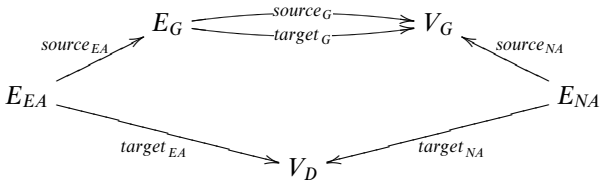
https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Definition A.1 (E-graph and E-graph morphism).** An *E-graph* $G$ with

$$G = (V_G, V_D, E_G, E_{NA}, E_{EA}, (source_j, target_j)_{j\in\{G,NA,EA\}})$$

consists of the sets:

— $V_G$ and $V_D$, which are called the graph and data nodes (or vertices), respectively;
— $E_G, E_{NA}$ and $E_{EA}$, which are called the graph, node attribute and edge attribute edges, respectively; and
— the source and target functions
  – $source_G : E_G \to V_G, target_G : E_G \to V_G$ for graph edges;
  – $source_{NA} : E_{NA} \to V_G, target_{NA} : E_{NA} \to V_D$ for node attribute edges; and
  – $source_{EA} : E_{EA} \to E_G, target_{EA} : E_{EA} \to V_D$ for edge attribute edges:




Consider the E-graphs $G^1$ and $G^2$ with

$$G^k = (V_G^k, V_D^k, E_G^k, E_{NA}^k, E_{EA}^k, (source_j^k, target_j^k)_{j\in\{G,NA,EA\}})$$

for $k = 1, 2$. An E-graph morphism $f : G^1 \to G^2$ is a tuple

$$(f_{V_G}, f_{V_D}, f_{E_G}, f_{E_{NA}}, f_{E_{EA}})$$

with

$$f_{V_i} : V_i^1 \to V_i^2$$
$$f_{E_j} : E_j^1 \to E_j^2$$

for

$$i \in \{G, D\}$$
$$j \in \{G, NA, EA\}$$

such that $f$ commutes with all source and target functions, for example

$$f_{V_G} \circ source_G^1 = source_G^2 \circ f_{E_G}.$$

The carrier sets of attribute values that form the single set $V_D$ of an $E$-graph are defined by an additional data algebra $D$, which also specifies the operations for generating and manipulating data values. The carrier sets $D_s$ of $D$ contain the data elements for each sort $s \in S$ according to a data signature

$$DSIG = (S_D, OP_D).$$

These carrier sets are combined by disjoint union and form the set $V_D$ of data elements.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

**Definition A.2 (attributed graph and attributed graph morphism).** Let

$$DSIG = (S_D, OP_D)$$

be a data signature with attribute value sorts $S'_D \subseteq S_D$. An attributed graph

$$AG = (G, D)$$

consists of an E-graph $G$ together with a $DSIG$-algebra $D$ such that

$$\cup_{s \in S'_D} D_S = V_D.$$

For two attributed graphs

$$AG^1 = (G^1, D^1)$$
$$AG^2 = (G^2, D^2),$$

an attributed graph morphism

$$f : AG^1 \to AG^2$$

is a pair $f = (f_G, f_D)$ with an E-graph morphism

$$f_G : G^1 \to G^2$$

and an algebra homomorphism

$$f_D : D^1 \to D^2$$

such that

$$\begin{array}{ccc} D^1_s & \xrightarrow{f_{D,s}} & D^2_s \\ \downarrow & (1) & \downarrow \\ V^1_D & \xrightarrow{f_{G,V_D}} & V^2_D \end{array}$$

commutes for all $s \in S'_D$, where the vertical arrows are inclusions.

The category of typed attributed graphs $\mathbf{AGraphs}_{ATG}$ has as objects all attributed graphs with a *typing morphism* to the attributed graph $ATG$ (type graph) and as arrows all attributed graph morphisms preserving the typing. Ehrig *et al.* (2006) showed that the category $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ is an adhesive HLR category, where the distinguished class of monomorphisms $\mathcal{M}$ contains all monomorphisms that are isomorphisms on the data part. For this reason, all results for adhesive HLR transformation systems presented in Ehrig *et al.* (2006) are valid. Since $\mathcal{M}$-adhesive categories (Ehrig *et al.* 2010) are a slight generalisation of weak adhesive and adhesive HLR categories, the category $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ is an $\mathcal{M}$-adhesive category.

## Appendix B. Proofs of additional technical results

In this section we provide proofs for Facts 2.21, 3.7, 3.8 and 3.9.

In order to prove Fact 2.21, we will use Definition B.1 and Lemma B.2 concerning the equivalence of single transformation steps using the on-the-fly construction of model transformations based on forward rules presented in Ehrig *et al.* (2009a). In this context,

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

forward sequences are constructed with an on-the-fly check for partial source consistency. Partial source consistency requires that the constructed forward sequence

$$G_0 \overset{tr_F^*}{\Longrightarrow} G_k$$

is partially match consistent, meaning that for each intermediate forward step

$$G_{k-1} \overset{tr_{k,F}}{\Longrightarrow} G_k,$$

the compatibility with the corresponding source step

$$G_{k-1,0} \overset{tr_{k,S}}{\Longrightarrow} G_{k,0}$$

of the simultaneously created source sequence

$$G_{00} \overset{tr_S^*}{\Longrightarrow} G_{k,0}$$

is checked. Compatibility requires that the forward match $m_{k,F}$ is forward consistent, which means that the comatch $n_{k,S}$ of the source step and the match $m_{k,F}$ of the forward step coincide on the source component with respect to the inclusion

$$G_{k-1,0} \hookrightarrow G_0 \hookrightarrow G_{k-1}.$$

The formal condition of a forward-consistent match is given in Definition B.1 by a pullback diagram where both matches satisfy the corresponding NACs, and intuitively, it specifies that the effective elements of the forward rule are matched for the first time in the forward sequence.

**Definition B.1 (forward-consistent match).** Given a partially match consistent sequence

$$\emptyset = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n-1,0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_{n-1},$$

a match $m_{n,F} : L_{n,F} \to G_{n-1}$ for $tr_{n,F} : L_{n,F} \to R_{n,F}$ is said to be *forward consistent* if there is a source match $m_{n,S}$ such that the diagram

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

is a pullback and the matches $m_{n,F}$ and $m_{n,S}$ satisfy the corresponding target and source *NACs*, respectively.

**Lemma B.2 (forward translation step).** Let $TR$ be a set of triple rules with $tr_i \in TR$ and $TR_F$ be the derived set of forward rules. We assume a partially match consistent forward sequence

$$\emptyset = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{i-1,0} \overset{g_{i-1}}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_{i-1}$$

and a corresponding forward translation sequence

$$G_0' \overset{tr_{FT}^*}{\Longrightarrow} G_{i-1}',$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

both with almost injective matches, such that

$$G'_{i-1} = G_{i-1} \oplus Att^{\mathbf{F}}_{G_0 \setminus G_{i-1,0}} \oplus Att^{\mathbf{T}}_{G_{i-1,0}},$$

Then the following are equivalent:

(1) There is a TGT-step $G_{i-1} \overset{tr_{i,F}, m_{i,F}}{\Longrightarrow} G_i$ with forward-consistent match $m_{i,F}$

(2) There is a forward translation TGT-step $G'_{i-1} \overset{tr_{i,FT}, m_{i,FT}}{\Longrightarrow} G'_i$.

Moreover, we have

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

*Proof.* The proof is given by the proof of Hermann *et al.* (2010b, Fact 1). □

**Fact 2.21 (equivalence of forward transformations and forward translation sequences).** See Section 2.2 for the statement.

*Proof.* We first show the equivalence of the sequences disregarding the NACs. Part 1 of the statement is equivalent to the existence of the sequence

$$G_0 \overset{tr_{1,F}, m_{1,F}}{\Longrightarrow} G_1 \overset{tr_{2,F}, m_{2,F}}{\Longrightarrow} G_2 \ldots \overset{tr_{n,F}, m_{n,F}}{\Longrightarrow} G_n$$

with $G^S_n = G^S$, where each match is forward consistent according to Definition B.1. Part 2 of the statement is equivalent to the existence of the complete forward translation sequence

$$G'_0 \overset{tr_{1,FT}, m_{1,FT}}{\Longrightarrow} G'_1 \overset{tr_{2,FT}, m_{2,FT}}{\Longrightarrow} G'_2 \ldots \overset{tr_{n,FT}, m_{n,FT}}{\Longrightarrow} G'_n$$

through $TR_{FT}$.

Disregarding the NACs, it remains to show that

$$G'^S_0 = Att^{\mathbf{F}}\left(G^S\right)$$
$$G'^S_n = Att^{\mathbf{T}}\left(G^S\right).$$

We apply Lemma B.2 for $i = 0$ with $G_{0,0} = \varnothing$ up to $i = n$ with $G_{n,0} = G_0$ and using $G^S_0 = G^S$ we have

$$\begin{aligned} G'^S_0 &= G^S_0 \oplus Att^{\mathbf{T}}_{G_{0,0}} \oplus Att^{\mathbf{F}}_{G^S_0 \setminus G^S_{0,0}} \\ &= G^S_0 \oplus Att^{\mathbf{F}}_{G^S_0} \\ &= G^S \oplus Att^{\mathbf{F}}_{G^S} \\ &= Att^{\mathbf{F}}\left(G^S\right) \end{aligned}$$

$$\begin{aligned} G'^S_n &= G^S_n \oplus Att^{\mathbf{T}}_{G^S_{n,0}} \oplus Att^{\mathbf{F}}_{G^S_0 \setminus G^S_{n,0}} \\ &= G^S_n \oplus Att^{\mathbf{T}}_{G^S_{n,0}} \\ &= G^S \oplus Att^{\mathbf{T}}_{G^S} \\ &= Att^{\mathbf{T}}\left(G^S\right). \end{aligned}$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

We will now show that the single steps are also NAC consistent. For each step, we have transformations

$$G_{i-1,0} \overset{tr_{i,S},m_{i,S}}{\Longrightarrow} G_{i,0}$$
$$G_{i-1} \overset{tr_{i,F},m_{i,F}}{\Longrightarrow} G_i$$
$$G'_{i-1} \overset{tr_{i,FT},m_{i,FT}}{\Longrightarrow} G'_i$$

with

$$G'_{i-1} = G_{i-1} \oplus Att^{\mathbf{F}}_{G_0 \setminus G_{i-1,0}} \oplus Att^{\mathbf{T}}_{G_{i-1,0}}$$
$$G'_i = G_i \oplus Att^{\mathbf{F}}_{G_0 \setminus G_{i,0}} \oplus Att^{\mathbf{T}}_{G_{i,0}}$$

and

$$m_{i,FT}|_{L_{i,F}} = m_{i,F}.$$

For a target NAC $n : L_i \to N$, we have to show that

$$m_{i,F} \models n \quad \text{if and only if} \quad m_{i,FT} \models n_{FT},$$

where $n_{FT}$ is the corresponding forward translation NAC of $n$. If $m_{i,FT} \not\models n_{F_T}$, we can find a monomorphism $q'$ with

$$q' \circ n_{FT} = m_{i,FT}.$$

Since $n = n_{FT}|_N$, we can define $q = q'|_N$, and it follows that

$$q \circ n = m_{i,F},$$

that is, $m_{i,F} \not\models n$.

Conversely, if $m_{i,F} \not\models n$, we can 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 $q' : N_{FT} \to G'_{i-1}$ such that $m_{i,FT} \not\models n_{FT}$.

Similarly, we have to show that for a source NAC $n : L \to N$,

$$m_{i,S} \models n \quad \text{if and only if} \quad m_{i,FT} \models n_{FT}.$$

As for target NACs, if $m_{i,FT} \not\models n_{FT}$, we can find a monomorphism $q'$ with

$$q' \circ n_{FT} = m_{i,FT},$$

and for the restriction to $L_i^S$ and $N^S$, it follows that

$$q^S \circ n^S = m^S_{i,FT},$$

that is, $m_{i,S} \not\models n$.

Conversely, if $m_{i,S} \not\models n$, we can find a monomorphism $q$ with

$$q \circ n = m_{i,S}.$$

We now define $q'$ by

$$q'(x) = m_{i,FT}(x) \qquad \text{for } x \in L_{FT}$$
$$q'(x) = q(x) \qquad \text{for } x \in N \setminus L_i,$$

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

and for each $x \in N^S \setminus L_i^S$, we have $q(x) \in G_{i-1,0}$. From the above characterisation of $G'_{i-1}$, it follows that the corresponding translation attributes `tr_x` and `tr_x_a` are set to **T** in $G'_{i-1}$. Thus, $q'$ is well defined and

$$q' \circ n_{FT} = m_{i,FT},$$

that is, $m_{i,FT} \not\models n_{FT}$.

The equality of the model transformation relations follows by the equality of the pairs $(G^S, G^T)$ in the model transformation sequences in both cases. □

**Fact 3.7 (automated generation of filter NACs).** See Section 3.1 for the statement.

*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 $N \overset{tr_{FT},n}{\Longrightarrow} M$ exists because the gluing condition is always satisfied for forward translation rules, as explained in Section 2.2, and the edge $e$ in $M$ is still labelled with a translation attribute set to '**F**', but $x$ is labelled 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 $H$, that is,

$$H' = H \oplus Att_{H_0}$$

for $H_0 \subseteq H'$, meaning that $H'$ has at most one translation attributes for each element in $H$ without translation attributes.

We will now show that $H'$ is not translatable, which implies that $M$ is misleading (Definition 3.3). Forward translation rules only modify translation attributes from '**F**' to '**T**'; they do not increase the number of translation attributes of a graph and no structural element is deleted. Thus, each graph $H_i$ in a TGT sequence $H' \overset{tr^*_{FT}}{\Longrightarrow} \overline{H}_n$ will contain the edge $e$ labelled with '**F**' because the rules that modify the translation attribute of $e$ are not applicable since $x$ is labelled 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, so $M$ is misleading. This means that $(n : L_{FT} \to N)$ is a filter NAC of $tr_{FT}$. By duality, the result also holds for a generated NAC with respect to an incoming edge. □

**Fact 3.8 (interactive generation of filter NACs).** See Section 3.1 for the statement.

*Proof.* The constructed NACs are filter NACs because the transformation step

$$K \overset{tr_{1,FT},m_1}{\Longrightarrow} P_1$$

contains the misleading graph $P_1$. Moreover, the procedure terminates since the number of critical pairs is bounded by the number of possible pairwise overlappings of the left-hand sides of the rules. The number of overlappings can be bounded by considering only constants and variables as possible attribute values. □

**Fact 3.9 (equivalence of transformations with filter NACs).** See Section 3.1 for the statement.

*Proof.* Sequence (1) consists of the same transformation diagrams as Sequence (2). The NAC-consistency of sequence (2) implies the NAC-consistency of sequence (1) because

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

each step in Sequence (2) involves a superset of the NACs for the corresponding step in Sequence (1).

For the inverse direction, consider a step

$$G_{i-1} \overset{tr_{(i,FT)},\mathrm{m}_{(i,FT)}}{\Longrightarrow} G_i,$$

which leads to the step

$$G_{i-1} \overset{tr_{(i,FN)},\mathrm{m}_{(i,FT)}}{\Longrightarrow} G_i$$

if NACs are not considered. We now assume that $m_{FT}$ does not satisfy some NAC of $tr_{FN}$. This implies that a filter NAC $(n : L_{i,FT} \to N)$ is not fulfilled since all the other NACs are fulfilled by the NAC-consistency of Sequence (1). Thus, there is a triple morphism $q : N \to G_{i-1}$ with $q \circ n = m_{i,FT}$. By Ehrig *et al.* (2006, Theorem 6.18) (the Restriction Theorem), we have that the transformation step

$$G_{i-1} \overset{tr_{(i,FN)},\mathrm{m}_{(i,FT)}}{\Longrightarrow} G_i$$

can be restricted to

$$N \overset{tr_{(i,FT)},n}{\Longrightarrow} H$$

with embedding $H \to G_i$. Now, by Definition 3.5 of filter NACs, we know that

$$N \overset{tr_{(i,FT)},n}{\Longrightarrow} H$$

and $H$ is misleading, which implies by Definition 3.3 that $G_i$ is not translatable. This contradicts the completely translated graph $G_n$ in sequence (1), so the filter NAC is fulfilled, which gives us the NAC-consistency of sequence (2). □

## References

AGG (2011) AGG. TFS-Group, TU Berlin. Homepage: `www.tfs.tu-berlin.de/agg`.

Arendt, T., Biermann, E., Jurack, S., Krause, C. and Taentzer, G. (2010) Henshin: Advanced concepts and tools for in-place EMF model transformations. In: Proceedings of the ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS'10). *Springer-Verlag Lecture Notes in Computer Science* **6394** 121–135.

Bisztray, D., Heckel, R. and Ehrig, H. (2009) Verification of architectural refactorings: Rule extraction and tool support. *Electronic Communications of the EASST* **16**.

Bohannon, A., Vaughan, J. A. and Pierce, B. C. (2006) Relational Lenses: A Language for Updateable Views. In: *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems – PODS '06* 338–347.

de Lara, J. and Guerra, E. (2008) Pattern-Based Model-to-Model Transformation. In: Ehrig, H., Heckel, R., Rozenberg, G. and Taentzer, G. (eds.) Proceedings of the 4th International Conference on Graph Transformations (ICGT 2008). *Springer-Verlag Lecture Notes in Computer Science* **5214** 426–441.

Ehrig, H., Ehrig, K., Ermel, C., Hermann, F. and Taentzer, G. (2007) Information Preserving Bidirectional Model Transformations. In: Proceedings FASE'07 – Fundamental Approaches to Software Engineering. *Springer-Verlag Lecture Notes in Computer Science* **4422** 72–86.

Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. (2006) *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science, Springer-Verlag.

https://doi.org/10.1017/S0960129512000370 Published online by Cambridge University Press

Ehrig, H., Ermel, C. and Hermann, F. (2008) On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars. In: *Proceedings GraMoT'08 – Graph and Model Transformation*, ACM 9–16.

Ehrig, H., Ermel, C., Hermann, F. and Prange, U. (2009a) On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars. In: Proceedings MODELS'09 – Model Driven Engineering Languages and Systems. *Springer-Verlag Lecture Notes in Computer Science* **5795** 241–255.

Ehrig, H., Golas, U. and Hermann, F. (2010) Categorical Frameworks for Graph Transformation and HLR Systems based on the DPO Approach. *Bulletin of the EATCS* **102** 111–121.

Ehrig, H., Hermann, F. and Sartorius, C. (2009b) Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions. *Electronic Communications of the EASST* **18**.

Ehrig, H., Pfender, M. and Schneider, H. (1973) Graph grammars: an algebraic approach. In: *14th Annual IEEE Symposium on Switching and Automata Theory* 167–180.

Ehrig, H. and Prange, U. (2008) Formal Analysis of Model Transformations Based on Triple Graph Rules with Kernels. In: Proceedings of the International Conference on Graph Transformation (ICGT'08). *Springer-Verlag Lecture Notes in Computer Science* **5214** 178–193.

Foster, J. (2009) *Bidirectional Programming Languages*, Dissertation, University of Pennsylvania.

Giese, H. and Hildebrandt, S. (2009) Efficient Model Synchronization of Large-Scale Models. Technical Report 28, Hasso Plattner Institute at the University of Potsdam.

Giese, H., Hildebrandt, S. and Lambers, L. (2010) Toward bridging the gap between formal semantics and implementation of triple graph grammars. Technical Report 37, Hasso Plattner Institute at the University of Potsdam.

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

Golas, U., Ehrig, H. and Hermann, F. (2011) Formal Specification of Model Transformations by Triple Graph Grammars with Application Conditions. In: Proceedings of GCM 2010. *Electronic Communications of the EASST* **39** 1–26.

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

Guerra, E. and de Lara, J. (2006b) Model View Management with Triple Graph Grammars. In: Proceedings of the International Conference on Graph Transformation (ICGT'06). *Springer-Verlag Lecture Notes in Computer Science* **4178** 351–366.

Habel, A. and Pennemann, K.-H. (2009) Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science* **19** (2) 245–296.

Hermann, F., Corradini, A. and Ehrig, H. (2014) Analysis of permutation equivalence in $\mathcal{M}$-adhesive transformation systems with negative application conditions. *Mathematical Structures in Computer Science* (this volume).

Hermann, F., Ehrig, H., Golas, U. and Orejas, F. (2010a) Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars. In: *Proceedings MDI'10 – Model Driven Interoperability*, ACM 22–31.

Hermann, F., Ehrig, H., Golas, U. and Orejas, F. (2010b) Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars – Extended Version. Technical Report 2010/13, FAk. IV, TU Berlin.

Hermann, F., Ehrig, H., Orejas, F., Czarnecki, K., Diskin, Z. and Xiong, Y. (2011) Correctness of Model Synchronization Based on Triple Graph Grammars. In: Proceedings of the

ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS'11). *Springer-Verlag Lecture Notes in Computer Science* **6981** 668–682.

Hermann, F., Ehrig, H., Orejas, F. and Golas, U. (2010c) Formal Analysis of Functional Behaviour of Model Transformations Based on Triple Graph Grammars. In: Proceedings of the International Conference on Graph Transformation (ICGT' 10). *Springer-Verlag Lecture Notes in Computer Science* **6372** 155–170.

Hermann, F., Hülsbusch, M. and König, B. (2010d) Specification and verification of model transformations. *Electronic Communications of the EASST* **30** 1–21.

Hidaka, S., Hu, Z., Inaba, K., Kato, H., Matsuda, K. and Nakano, K. (2010) Bidirectionalizing graph transformations. In: *Proceedings of the 15th ACM SIGPLAN International Conference on Functional programming (ICFP '10)* 205–216.

Kindler, E. and Wagner, R. (2007) Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios. Technical Report tr-ri-07-284, Department of Computer Science, University of Paderborn, Germany.

Klar, F., Lauder, M., Königs, A. and Schürr, A. (2010) Extended Triple Graph Grammars with Efficient and Compatible Graph Translators. In: Graph Transformations and Model Driven Enginering – Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday. *Springer-Verlag Lecture Notes in Computer Science* **5765** 144–177.

Königs, A. and Schürr, A. (2006) Tool Integration with Triple Graph Grammars – A Survey. In: Proceedings, SegraVis School on Foundations of Visual Modelling Techniques. *Electronic Notes in Theoretical Computer Science* **148** 113–150.

Lack, S. and Sobociński, P. (2005) Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications* **39** (2) 511–546.

Lambers, L. (2009) *Certifying Rule-Based Models using Graph Transformation*, Ph.D. thesis, Technische Universität Berlin.

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

Orejas, F., Guerra, E., de Lara, J. and Ehrig, H. (2009) Correctness, Completeness and Termination of Pattern-Based Model-to-Model Transformation. In: Kurz, A., Lenisa, M. and Tarlecki, A. (eds.) International Conference on Algebra and Coalgebra in Computer Science (CALCO'09). *Springer-Verlag Lecture Notes in Computer Science* **5728** 383–397.

Plump, D. (1993) Hypergraph Rewriting: Critical Pairs and Undecidability of Confluence. In: *Term Graph Rewriting: Theory and Practice*, John Wiley 201–213.

Plump, D. (2005) Confluence of Graph Transformation Revisited. In: Processes, Terms and Cycles: Steps on the Road to Infinity. *Springer-Verlag Lecture Notes in Computer Science* **3838** 280–308.

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

Schürr, A. (1994) Specification of Graph Translators with Triple Graph Grammars. In: Proceedings of the Workshop on Graph-Theoretic Concepts in Computer Science (WG'94). *Springer-Verlag Lecture Notes in Computer Science* **903** 151–163.

Schürr, A. and Klar, F. (2008) 15 years of triple graph grammars. In: *Proceedings of the International Conference on Graph Transformation (ICGT 2008)*, 411–425.

Stevens, P. (2008) A Landscape of Bidirectional Model Transformations. In: Proceedings of GTTSE 2008. *Springer-Verlag Lecture Notes in Computer Science* **5235** 408–424.

Taentzer, G. *et al.* (2005) Model Transformation by Graph Transformation: A Comparative Study. In: *Proceedings – Workshop Model Transformation in Practice*.