Correctness of Model Synchronization Based on Triple Graph Grammars - Extended Version

Frank Hermann[1,2,⋆], Hartmut Ehrig[1], Fernando Orejas[3], Krzysztof Czarnecki[4], Zinovy Diskin[4], and Yingfei Xiong[4]

[1] Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, Germany, {frank,ehrig}@cs.tu-berlin.de
[2] Interdisciplinary Center for Security, Reliability and Trust, Université du Luxembourg
[3] Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, orejas(at)lsi.upc.edu
[4] Generative Software Development Lab, University of Waterloo, Canada
{kczarnec, zdiskin, yingfei}@gsd.uwaterloo.ca

Abstract. Triple graph grammars (TGGs) have been used successfully to analyze correctness and completeness of bidirectional model transformations, but a corresponding formal approach to model synchronization has been missing. This paper closes this gap by providing a formal synchronization framework with bidirectional update propagation operations. They are generated from a TGG, which specifies the language of all consistently integrated source and target models.
As a main result, we show that the generated synchronization framework is correct and complete, provided that forward and backward propagation operations are deterministic. Correctness essentially means that the propagation operations preserve consistency. Moreover, we analyze the conditions under which the operations are inverse to each other. All constructions and results are motivated and explained by a small running example using concrete visual syntax and abstract syntax notation based on typed attributed graphs.

Keywords: Model Synchronization, Correctness, Bidirectional Model Transformation, Triple Graph Grammars

1 Introduction

Bidirectional model transformations are a key concept for model generation and synchronization within model driven engineering (MDE, see [27,24,2]). Triple graph grammars (TGGs) have been successfully applied in several case studies for bidirectional model transformation, model integration and synchronization [21,26,11,10], and in the implementation of QVT [14]. Inspired by Schürr et al. [25,26], we started to develop a formal theory of TGGs [8,17], which allows us to handle correctness, completeness, termination, and functional behavior of model transformations.

The main goal of this paper is to provide a TGG framework for model synchronization with correctness guarantees, which is based on the theory of TGGs, work on incremental synchronization by Giese et al. [11,10], and the replica synchronization framework [4]. The main ideas and results are the following:

⋆ Supported by the National Research Fund, Luxembourg (AM2a)

1. Models are synchronized by propagating changes from a source model to a corresponding target model using forward and backward propagation operations. The operations are specified by a TGG model framework, inspired by symmetric replica synchronizers [4] and realized by model transformations based on TGGs [8]. The specified TGG also defines consistency of source and target models.
2. Since TGGs define, in general, non-deterministic model transformations, the derived synchronization operations are, in general, non-deterministic. But we are able to provide sufficient static conditions based on TGGs to ensure that the operations are deterministic.
3. The main result shows that a TGG synchronization framework with deterministic synchronization operations is correct, i.e., consistency preserving, and complete. We also give sufficient static conditions for invertability and weak invertability of the framework, where "weak" restricts invertability to a subclass of inputs.

Deriving a synchronization framework from a TGG has the following practical benefits. Consistency of related domains is defined declaritively and in a pattern-based style, using the rules of a TGG. After executing a synchronization operation, consistency of source and target models is always ensured (correctness) and the propagation operations can be performed for all valid inputs (completeness). The required static conditions of a TGG and the additional conditions for invertibility can be checked automatically using the existing tool support of AGG [28].

The next section presents our running example and Sec. 3 introduces the TGG model framework. Therafter, we briefly review model transformations based on TGGs in Sec. 4 and define the general synchronization process in Sec. 5. Section 6 presents the main result on the correctness of model synchronization. Finally, Secs. 7 and 8 discuss related work, conclusions, and future work. The proof of our main result and necessary technical constructions as well as results are given in App. A, while App. B presents the TGG of our case study together with the corresponding details of the analysis results.

2 Running Example

Throughout the paper, we use a simple running example, which is based on previous work [3]. The example considers the synchronization of two organizational diagrams as shown in Fig. 1. Diagrams in the first domain—depicted left—provides details about the salary components and is restricted to persons of the marketing department. The second domain provides additional information about birth dates (marked by "*") and does not show the salary components. Therefore, both domains contain exclusive information and none of them can be interpreted as a view—defined by a query—of the other. Both diagrams together with some correspondence structure build up an integrated model, where we refer by source model to the first and by target model to the second diagram. Such an integrated model is called *consistent*, if the diagrams coincide on names of corresponding persons and the salary values are equal to the sums of the corresponding base and bonus values.

Example 1 (Integrated Model). The first row of Fig. 1 shows a consistent integrated model M in visual notation. The source model of M consists of two persons belonging

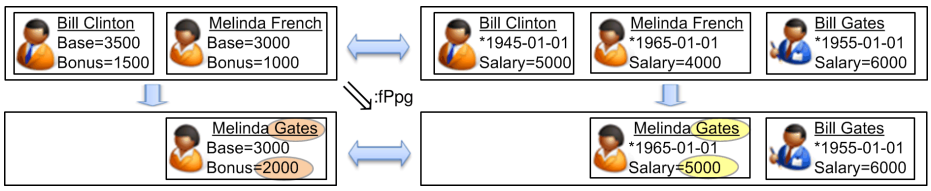


Fig. 1. Forward propagation

to the marketing department (depicted as persons without pencils) and the target model additionally contains the person "Bill Gates" belonging to the technical department (depicted as a person with pencil). The fifth row of Fig. 7 in Sec. 5 shows the corresponding underlying formal graph representation of the integrated model.

The synchronization problem is to propagate a model update in a way, such that the resulting integrated model is consistent. Looking at Fig. 1, we see a source model update that specifies the removal of person "Bill Clinton" and a change of attributes LastName and Bonus of person "Melinda French". The executed forward propagation (fPpg) removes person "Bill Clinton" and updates the attribute values of "Melinda French" in the target model, while preserving the unchanged birth date value.

3 Model Synchronization Framework Based on TGGs

Model synchronization aims to achieve consistency among interrelated models. A general way of specifying consistency for models of a source and a target domain is to provide a consistency relation that defines the consistent pairs (M^S, M^T) of source and target models. We argue that triple graph grammars (TGGs) are an adequate technique for this purpose. For this reason, we first review main concepts of TGGs [26,8].

In the framework of TGGs, an integrated model is represented by a triple graph consisting of three graphs G^S, G^C, and G^T, called source, correspondence, and target graphs, respectively, together with two mappings (graph morphisms) $s_G : G^C \to G^S$ and $t_G : G^C \to G^T$. Our triple graphs may also contain attributed nodes and edges [8,7]. The two mappings in G specify a *correspondence* $r : G^S \leftrightarrow G^T$, which relates the elements of G^S with their corresponding elements of G^T and vice versa. However, it is usually sufficient to have explicit correspondences between nodes only. For simplicity, we use double arrows ($\leftrightarrow$) as an equivalent shorter notation for triple graphs, whenever the the explicit correspondence graph can be omitted.

Triple graphs are related by triple graph morphisms $m : G \to H$ consisting of three graph morphisms that preserve the associated correspondences (i.e., the diagrams on the right commute).

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

Our triple graphs are typed. This means that a type triple graph TG is given (playing the role of a metamodel) and, moreover, every triple graph G is typed by a triple graph morphism $type_G : G \to TG$. It is required that morphisms between typed triple graphs

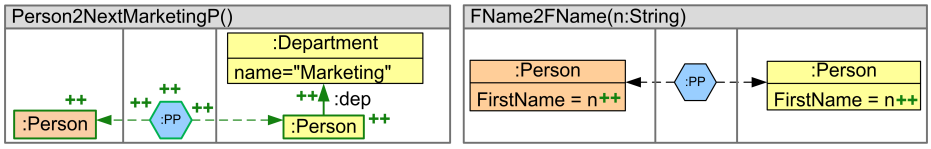


Fig. 2. Some triple rules of the TGG

preserve the typing. For $TG = (TG^S \leftarrow TG^C \rightarrow TG^T)$, we use $VL(TG)$, $VL(TG^S)$, and $VL(TG^T)$ to denote the classes of all graphs typed over TG, TG^S, and TG^T, respectively.

A TGG specifies a language of triple graphs, which are considered as consistently integrated models. The triple rules of a TGG are used to synchronously build up source and target models, together with the correspondence structures.

A triple rule tr, depicted on the right, is an inclusion of triple graphs, represented $L \hookrightarrow R$. Notice that one or more of the rule components tr^S, tr^C, and tr^T may be empty. In the example, this is the case for a rule concerning employees of the technical department within the target model. 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 $m : L \rightarrow G$. 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, as depicted above on the right. This triple graph transformation (TGT) step is denoted by $G \overset{tr,m}{\Longrightarrow} H$. Moreover, triple rules can be extended by negative application conditions (NACs) for restricting their application to specific matches [17]. A triple graph grammar $TGG = (TG, S, TR)$ consists of a triple type graph TG, a triple start graph S and a set TR of triple rules and generates the triple graph language $VL(TGG) \subseteq VL(TG)$.

$$\begin{array}{ccc} L = (L^S \xleftarrow{s_L} L^C \xrightarrow{t_L} L^T) & & L \overset{tr}{\hookrightarrow} R \\ tr \downarrow \quad tr^S \downarrow \quad tr^C \downarrow \quad tr^T \downarrow & & m \downarrow \quad (PO) \quad \downarrow n \\ R = (R^S \xleftarrow[s_R]{} R^C \xrightarrow[t_R]{} R^T) & & G \underset{t}{\hookrightarrow} H \end{array}$$

Example 2 (Triple Rules). Figure 2 shows some triple rules of our running example using short notation, i.e., left- and right-hand side of a rule are depicted in one triple graph and the elements to be created have the label "++". The first rule Person2NextMarketingP requires an existing marketing department. It creates a new person in the target component together with its corresponding person in the source component and the explicit correspondence structure. The TGG contains a similar rule (not depicted) for initially creating the marketing department together with one person, where an additional NAC ensures that none of the existing departments is called "Marketing". The second rule in Fig. 2 extends two corresponding persons by their first names. There are further similar rules for the handling of the remaining attributes. In particular, the rule for the attribute birth is the empty rule on the source component.

A TGG model framework specifies the possible correspondences between models and updates of models according to Def. 1 below. The framework is closely related to the abstract framework for diagonal replica synchronizers [4] and triple spaces [5]. In our context, a model update $\delta : G \rightarrow G'$ is specified as a *graph modification* $\delta : G \overset{i_1}{\leftarrow} I \overset{i_2}{\rightarrow} G'$. The relating morphisms $i_1 : I \hookrightarrow G$ and $i_2 : I \hookrightarrow G'$ are inclusions and

specify the elements in the interface I that are preserved by the modification. While graph modifications are also triple graphs by definition, it is conceptually important to distinguish between correspondences and updates δ.

Definition 1 (TGG Model Framework). *Let $TGG = (TG, \emptyset, TR)$ be a triple graph grammar with empty start graph. The derived* TGG model framework $MF(TGG) = (VL(TG^S), VL(TG^T), R, C, \Delta_S, \Delta_T)$ *consists of source domain $VL(TG^S)$, target domain $VL(TG^T)$, the set R of correspondence relations given by $R = VL(TG)$, the set C of consistent correspondence relations $C \subseteq R$ given by $C = VL(TGG)$, (i.e., R contains all integrated models and C all consistently integrated ones), and sets Δ_S, Δ_T of graph modifications for the source and target domains, given by $\Delta_S = \{a : G^S \to G'^S \mid G^S, G'^S \in VL(TG^S)$, and a is a graph modification$\}$ and $\Delta_T = \{b : G^T \to G'^T \mid G^T, G'^T \in VL(TG^T)$, and b is a graph modification$\}$, respectively.*

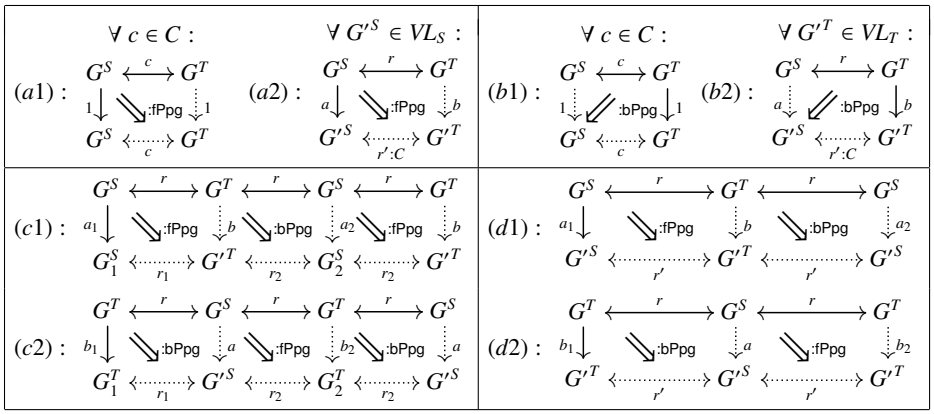


Fig. 3. Laws for correct and (weak) invertible synchronization frameworks

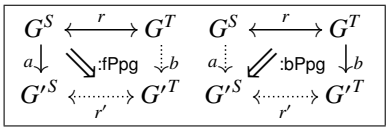


Fig. 4. Synchronization operations

Given a TGG model framework, the synchronization problem is to provide suitable forward and backward propagation operations fPpg and bPpg, which are total and deterministic (see Fig. 4, where we use solid lines for the inputs and dashed lines for the outputs). The required input for fPpg is an integrated model (correspondence relation) $G^S \leftrightarrow G^T$ together with a source model update (graph modification) $a : G^S \to G'^S$. In a common tool environment, both inputs are either available directly or can be obtained. For example, the graph modification of a model update can be derived via standard difference computation and the initial correspondence can be computed based on TGG integration concepts [6,21]. Note that determinism of fPpg means that the resulting correspondence $G'^S \leftrightarrow G'^T$ and target model update $b : G^T \to G'^T$ are uniquely determined. The propagation operations are *correct*, if they additionally preserve consistency as specified

by laws $(a1)$ – $(b2)$ in Fig. 3. Law $(a2)$ means that fPpg always produces consistent correspondences from consistent updated source models G'^S. Law $(a1)$ means that if the given update is the identity and the given correspondence is consistent, then fPpg changes nothing. Laws $(b1)$ and $(b2)$ are the dual versions concerning bPpg. Moreover, the sets VL_S and VL_T specify the *consistent source and target models*, which are given by the source and target components of the integrated models in $C = VL(TGG)$.

Definition 2 (Synchronization Problem and Framework). *Let* $MF = (VL(TG^S), VL(TG^T), R, C, \Delta_S, \Delta_T)$ *be a TGG model framework (see Def. 1). The* forward synchronization problem *is to construct an operation* $\mathsf{fPpg} : R \otimes \Delta_S \to R \times \Delta_T$ *leading to the left diagram in Fig. 4, called synchronization tile, where* $R \otimes \Delta_S = \{(r, a) \in R \times \Delta_S \mid r\colon G^S \leftrightarrow G^T, a\colon G^S \to G'^S\}$*, i.e., a and r coincide on* G^S*. The pair* $(r, a) \in R \otimes \Delta_S$ *is called premise and* $(r', b) \in R \times \Delta_T$ *is called solution of the forward synchronization problem, written* $\mathsf{fPpg}(r, a) = (r', b)$*. The* backward synchronization problem *is to construct an operation* bPpg *leading to the right diagram in Fig. 4. The operations* fPpg *and* bPpg *are called* correct *with respect to consistency function C, if axioms* $(a1)$ *and* $(a2)$ *resp.* $(b1)$ *and* $(b2)$ *in Fig. 3 are satisfied.*

Giventotal and deterministic propagation operations fPpg *and* bPpg*, the derived* synchronization framework *Synch*(TGG) *is given by Synch*$(TGG) = (MF, \mathsf{fPpg}, \mathsf{bPpg})$*. It is called* correct*, if* fPpg *and* bPpg *are correct; it is* weakly invertible *if axioms* $(c1)$ *and* $(c2)$ *in Fig. 3 are satisfied; and it is* invertible *if additionally axioms* $(d1)$ *and* $(d2)$ *in Fig. 3 are satisfied.*

Remark 1 (Correctness and Invertibility). Correctness of fPpg according to $(a1)$ means that for each consistent correspondence $c\colon G^S \leftrightarrow G^T$ and identity as modification $1 : G^S \to G^S$ we have an identical result, i.e. , $\mathsf{fPpg}(c, 1) = (c, 1)$. According to $(a2)$, we have for each general correspondence $r\colon G^S \leftrightarrow G^T$ and modification $a\colon G^S \to G'^S$ with consistent source model $G'^S \in VL_S$ a solution $(r', b) = \mathsf{fPpg}(r, a)$, where $r'\colon G'^S \leftrightarrow G'^T$ is consistent, i.e., $r' \in C$. Note that also for non-consistent $r\colon G^S \leftrightarrow G^T$ the result $r'\colon G'^S \leftrightarrow G'^T$ is consistent, provided that G'^S is consistent.

Weak invertibility (laws $(c1)$ and $(c2)$) imply that the operations are inverse of each other for a restricted set of inputs. Update b in $(c1)$ is assumed to be part of the result of a forward propagation and update a in $(c2)$ is assumed to be derived from a backward propagation. Invertibility ($(d1)$ and $(d2)$) means that the operations are essentially inverse of each other, although the interfaces of a_1 and a_2 (resp. b_1 and b_2) may be different. Invertibility requires effectively that all information in one domain is completely reflected in the other domain.

4 Model Transformation Based on TGGs

The *operational rules* for implementing bidirectional model transformations can be generated automatically from a TGG. The sets TR_S and TR_F contain all source and forward rules, respectively, and are derived from the triple rules TR as shown in the diagrams below. The rules are used to implement source-to-target transformations. The sets of target rules TR_T and backward rules TR_B are derived analogously and the generation of operational rules has been extended to triple rules with negative application conditions [8].

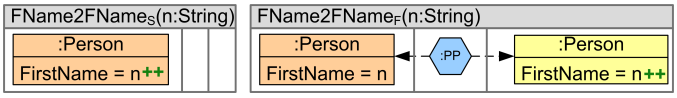


Fig. 5. Derived source and forward rules

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

triple rule tr

$$\begin{array}{c} (L^S \longleftarrow \varnothing \longrightarrow \varnothing) \\ {\scriptstyle tr^S}\downarrow \quad \downarrow \quad \downarrow \\ (R^S \longleftarrow \varnothing \longrightarrow \varnothing) \end{array}$$

source rule tr_S

$$\begin{array}{c} (R^S \xleftarrow{tr^S \circ s_L} L^C \xrightarrow{t_L} L^T) \\ {\scriptstyle id}\downarrow \quad {\scriptstyle tr^C}\downarrow \quad \downarrow{\scriptstyle tr^T} \\ (R^S \xleftarrow[s_R]{} R^C \xrightarrow[t_R]{} R^T) \end{array}$$

forward rule tr_F

Example 3 (Operational Rules). The rules in Fig. 5 are the derived source and forward rules of the triple rule FName2FName in Fig. 2.

The derived operational rules provide the basis for the definition of model transformations based on source-consistent forward transformation sequences [8,13]. *Source consistency* of a forward sequence $(G_0 \overset{tr_F^*}{\Longrightarrow} G_n)$ via TR_F is a control condition which requires that there is a corresponding source sequence $(\varnothing \overset{tr_S^*}{\Longrightarrow} G_0)$ via TR_S, such that matches of corresponding source and forward steps are compatible ($n^S_{i,S}(x) = m^S_{i,F}(x)$). The source sequence is obtained by parsing the given source model in order to guide the forward transformation. Moreover, source and forward sequences can be constructed simultaneously and backtracking can be reduced in order to derive efficient executions of model transformations [8,17]. Given a source model G^S, a *model transformation sequence* for G^S is given by $(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T)$, where G^T is the resulting target model derived from the source-consistent forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ with $G_0 = (G^S \leftarrow \varnothing \rightarrow \varnothing)$ and $G_n = (G^S \leftarrow G^C \rightarrow G^T)$.

Model transformations based on model transformation sequences are always syntactically correct and complete [8,13,17]. *Correctness* means that for each source model G^S that is transformed into a target model G^T there is a consistent integrated model $G = (G^S \leftarrow G^C \rightarrow G^T)$ in the language of consistent integrated models $VL(TGG)$ defined by the TGG. *Completeness* ensures that for each consistent source model there is a forward transformation sequence transforming it into a consistent target model.

The concept of *forward translation rules* [17] provides a simple way of implementing model transformations such that source consistency is ensured automatically. A forward translation rule tr_{FT} extends the forward rule tr_F by additional Boolean valued translation attributes, which are markers for elements in the source model and specify whether the elements have been translated already. Each forward translation rule tr_{FT} turns the markers of the source elements that are translated by this rule from **F** to **T** (i.e., the elements that are created by tr_S). The model transformation is successfully executed if the source model is completely marked with **T**. We indicate these markers in the examples by check marks in the visual notation and by bold font face in the graph representation. Similarly, from the triple rules, we can also create *consistency checking rules* [19], which, given an integrated model $(G^S \leftrightarrow G^T)$, simulate the creation of the

model by marking its elements. If all elements are marked with **T**, then ($G^S \leftrightarrow G^T$) belongs to $VL(TGG)$.

5 General Synchronization Process Based on TGGs

This section shows how to construct the operation fPpg of a TGG synchronization framework (see Def. 2) as a composition of auxiliary operations ⟨fAln, Del, fAdd⟩. Symmetrically, operations ⟨bAln, Del, bAdd⟩ are used to define the operation bPpg. As a general requirement, the given TGG has to provide *deterministic* sets of operational rules, meaning that the algorithmic execution of the forward translation, backward translation, and consistency checking rules ensures functional behavior (unique results) and does not require backtracking. For this purpose, additional policies can be defined that restrict the matches of operational rulesas presented in App. A by Fact 7 and as discussed in Ex. 5 in Sec. 6. Fact 1 in Sec. 6 provides sufficient conditions for deterministic operational rules. We provide additional static conditions and automated checks in the technical report [19].

The general synchronization process is performed as follows (see Fig. 6; we use double arrows (↔) for correspondence in the signature of the operations, and the explicit triple graphs for the construction details). Given two corresponding models G^S and G^T and an update of G^S via the graph modification $a = (G^S \xleftarrow{a_1} D^S \xrightarrow{a_2} G'^S)$ with $G'^S \in VL_S$, the forward propagation fPpg of δ_S is performed in three steps via the auxiliary operations fAln, Del, and fAdd. At first, the deletion performed in a is reflected into the correspondence relation between G^S and G^T by calculating the forward alignment remainder via operation fAln. This step deletes all correspondence elements whose elements in G^S have been deleted. In the second step, performed via operation Del, the two maximal subgraphs $G^S_k \subseteq G^S$ and $G^T_k \subseteq G^T$ are computed such that they form a consistent integrated model in $VL(TGG)$ according to the TGG. All elements that are in G^T but not in G^T_k are deleted, i.e., the new target model is given by G^T_k. Finally, in the last step (operation fAdd), the elements in G'^S that extend G^S_k are transformed to corresponding structures in G'^T, i.e., G^T_k is extended by these new structures. The result of fAdd, and hence also fPpg, is a consistent integrated model.

Definition 3 (Auxiliary TGG Operations). *Let $TGG = (TG, \emptyset, TR)$ be a TGG with deterministic sets of operational rules and let further $MF(TGG)$ be the derived TGG model framework.*

1. *The auxiliary operation* fAln *computing the forward alignment remainder is given by* $\mathsf{fAln}(r, a) = r'$*, as specified in the upper part of Fig. 6. The square marked by (PB) is a pullback, meaning that D^C is the intersection of D^S and G^C.*
2. *Let $r = (s, t)\colon G^S \leftrightarrow G^T$ be a correspondence relation, then the result of the auxiliary operation* Del *is the maximal consistent subgraph $G^S_k \leftrightarrow G^T_k$ of r, given by* $\mathsf{Del}(r) = (a, r', b)$*, which is specified in the middle part of Fig. 6.*
3. *Let $r = (s, t)\colon G^S \leftrightarrow G^T$ be a consistent correspondence relation, $a = (1, a_2) : G^S \to G'^S$ be a source modification and $G'^S \in VL_S$. The result of the auxiliary operation* fAdd*, for propagating the additions of source modification a, is a consistent model $G'^S \leftrightarrow G'^T$ extending $G^S \leftrightarrow G^T$, and is given by* $\mathsf{fAdd}(r, a) = (r', b)$*, according to the lower part of Fig. 6.*

Remark 2 (Auxiliary TGG Operations). Intuitively, operation fAln constructs the new correspondence graph D^C from the given G^C by deleting all correspondence elements in G^C whose associated elements in G^S are deleted via update a and, for this reason, do not occur in D^S. Operation Del is executed by applying consistency checking rules (cf. Sec. 4) to the given integrated model until no rule is applicable any more. If, at the end, $G^S \leftrightarrow G^T$ is completely marked, the integrated model is already consistent; otherwise, the result is the largest consistent integrated model included in $G^S \leftrightarrow G^T$. Technically, the application of the consistency checking rules corresponds to a maximal triple rule sequence as shown in the right middle part of Fig. 6 and discussed in more detail in App. A. Finally, fAdd is executed by applying forward translation rules (cf. Sec. 4) to $G'^S \leftrightarrow G^T$ until all the elements in G'^S are marked. That is, these TGT steps build a model transformation of G'^S extending G^T. Technically, the application of the forward translation rules corresponds to a source-consistent forward sequence from G_0 to G', as shown in the right lower part of Fig. 6. By correctness of model transformations [8], the sequence implies consistency of G' as stated above. The constructions for these auxiliary operations are provided in full detail in App. B.1.

Example 4 (Forward Propagation via Operation fPpg*).* Figure 7 shows the application of the three steps of synchronization operation fPpg to the visual models of our running example. After removing the dangling correspondence node of the alignment in the first step (fAln), the maximal consistent subgraph of the integrated model is computed (Del) by stepwise marking the consistent parts: consistent parts are indicated by grey boxes with checkmarks in the visual notation and by bold font faces in the graph representation. Note that node "Bill Gates" is part of the target graph in this maximal consistent subgraph, even though it is not in correspondence with any element of the source graph.

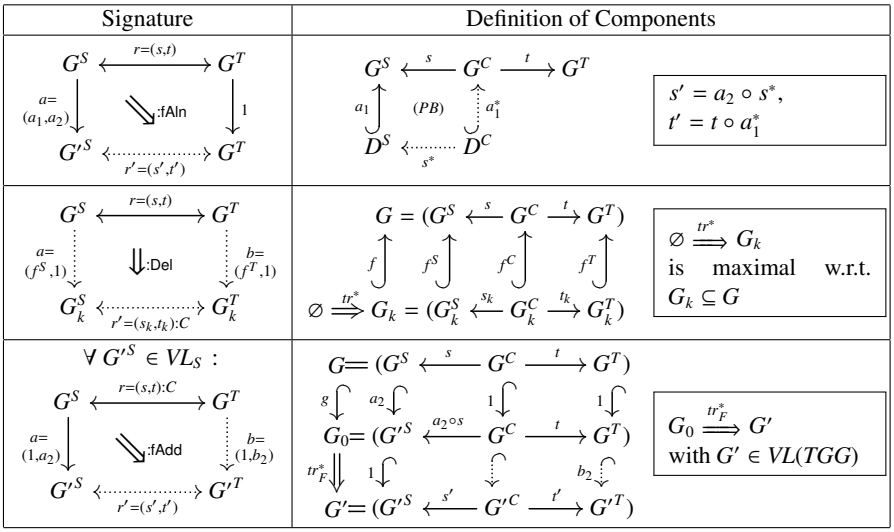


Fig. 6. Auxiliary operations fAln, Del and fAdd

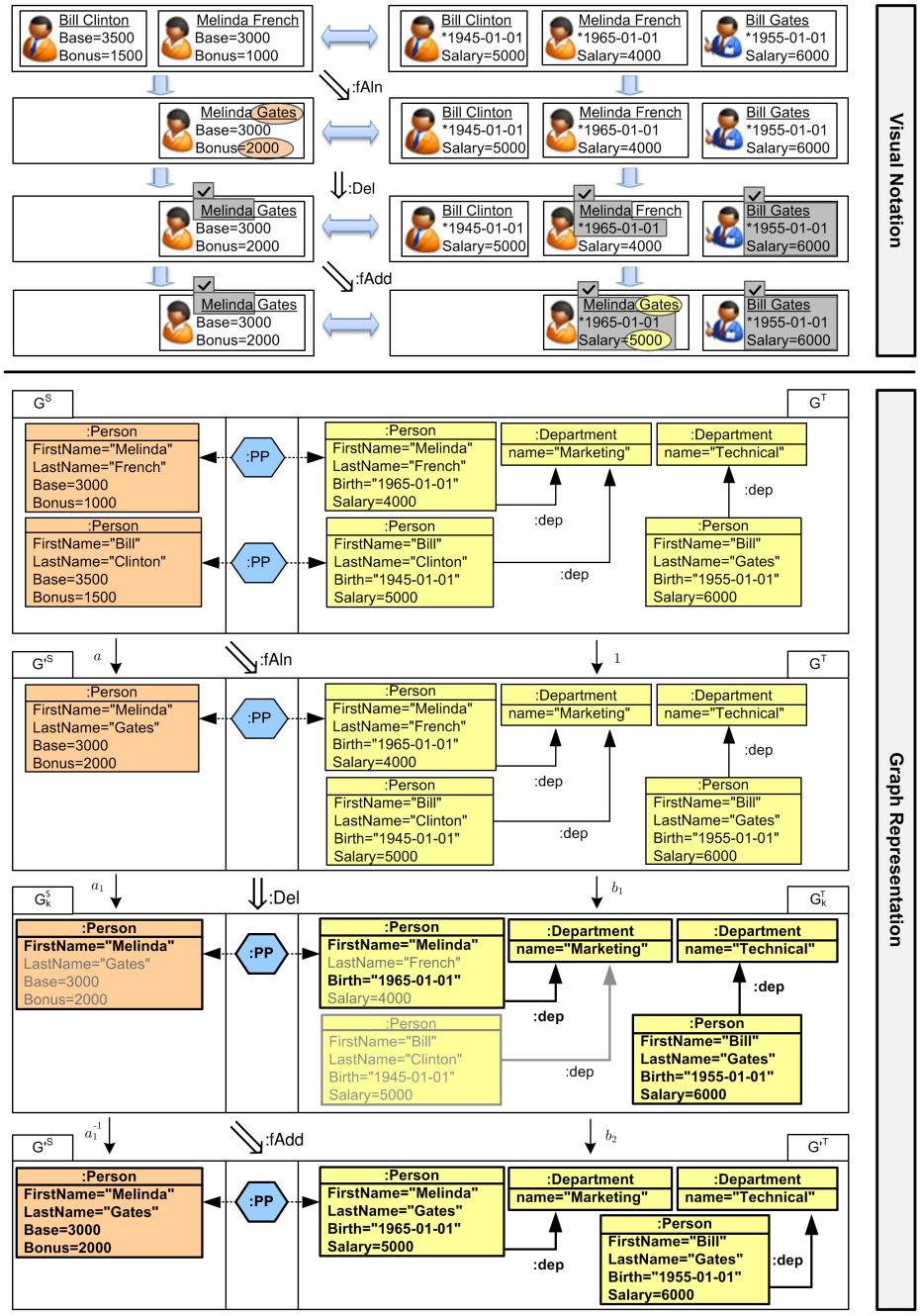
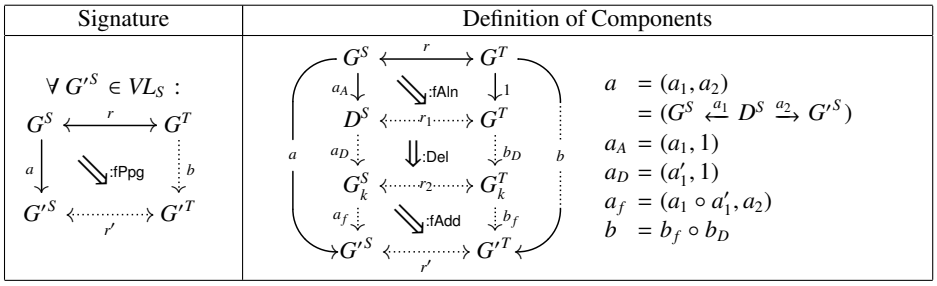


Fig. 7. Forward propagation in detail: visual notation (top) and graph representation (bottom)




```
1  /* == alignment remainder == */
2  forall(correspondence nodes without image in the source model){
3     delete these elements }
4  /* ==== delete === */
5  while(there is a triple rule p such that R\L is unmarked){
6     apply to G the consistency checking rule corresponding to p }
7  forall(unmarked nodes and edges from the target model){
8     delete these elements }
9  /* ===== add ===== */
10 while(there is a forward translation rule applicable to G){
11    apply to G the forward translation rule }
```

Fig. 8. Synchronization operation fPpg - top: formal definition, bottom: algorithm

In the final step (fAdd), the inconsistent elements in the target model are removed and the remaining new elements of the update are propagated towards the target model by model transformation, such that all elements are finally marked as consistent.

Definition 4 (Derived TGG Synchronization Framework). *Let* $TGG = (TG, \emptyset, TR)$ *be a TGG with deterministic sets of derived operational rules and with derived model framework* $MF(TGG)$*, then operation* fPpg *of the* derived TGG synchronization framework *is given according to Def. 2 by the composition of auxiliary operations (*fAln, Del, fAdd*) with construction in Rem. 3. Symmetrically—not shown explicitly—we obtain* bPpg *as composition of auxiliary operations (*bAln, Del, bAdd*).*

Remark 3 (Construction of fPpg *according to Fig. 8).* Given a not necessarily consistent integrated model $r\colon G^S \leftrightarrow G^T$ and source model update $a\colon G^S \to G'^S$ with $G'^S \in VL_S$, we compute $\mathsf{fPpg}(r, a)$ as follows. First, fAln computes the correspondence $(D^S \leftrightarrow G^T)$, where D^S is the part of G^S that is preserved by update a. Then, Del computes its maximal consistent integrated submodel $(G^S_k \leftrightarrow G^T_k)$. Finally, fAdd composes the embedding $G^S_k \to G'^S$ with correspondence $(G^S_k \leftrightarrow G^T_k)$ leading to $(G'^S \leftrightarrow G^T_k)$, which is then extended into the consistent integrated model $(G'^S \leftrightarrow G'^T)$ via forward transformation. If $G'^S \notin VL_S$, then the result is given by $b = (1, 1)\colon G^T \to G^T$ together with the correspondence relation $r' = (\emptyset, \emptyset)$ and additionally, an error message is provided. The bottom part of Fig. 8 describes this construction algorithmically in pseudo code, leaving out the error handling; marking is explained in Sec. 4.

6 Correctness of Model Synchronization Based on TGGs

Based on the derived TGG synchronization framework (Def. 4), we now show our main result concerning correctness, completeness, and invertibility. According to Def. 2, correctness requires that the synchronization operations are deterministic, i.e., they have functional behaviour and ensure laws ($a1$) - ($b2$). Concerning the first property, Fact 1 below provides a sufficient condition based on the notion of critical pairs [7], which is used in the automated analysis engine of the tool AGG [28]. A critical pair specifies a conflict between two rules in minimal context. Solving a conflict means to find compatible merging transformation steps, which is formalized by the notion of strict confluence [7]. The result is provided for almost injective matches, which means that matches are injective on the graph part and may evaluate different attribute expressions to the same values. *Completeness* requires that operations fPpg and bPpg can be successfully applied to all consistent source models $G'^S \in VL_S$ and target models $G'^T \in VL_T$, respectively. For this reason, additional propagation policies are defined in order to eliminate non-determinism. They can be seen as a kind of application conditions for the rules and are called *conservative*, if they preserve the completeness result. Fact 7 in App. A provides a sufficient static condition for checking this property.

Fact 1 (Deterministic Synchronization Operations). *Let TGG be a triple graph grammar and let matches be restricted to almost injective morphisms. If the critical pairs of the sets of operational rules are strictly confluent and the systems of rules are terminating, then the sets of operational rules are deterministic, which implies that the derived synchronization operations* fPpg *and* bPpg *are deterministic as well.*

Proof. The proof is given in App. A.5. □

Remark 4 (Termination). In order to ensure termination of the TGG constructions, we can check that each operational rule is modifying at least one translation attribute (cf. Sec. 4), which is a sufficient condition as shown by Thm. 1 in [17] for model transformation sequences.

Invertibility of propagation operations depends on additional properties of a TGG. For this purpose, we distinguish between different types of triple rules. By TR^{+s} we denote the triple rules of TR that are creating on the source component and by TR^{1s} those that are identical on the source component and analogously for the target component. A TGG is called *pure*, if $TR^{1s} \subseteq TR_T$ and $TR^{1t} \subseteq TR_S$ meaning that the source-identic triple rules are empty rules on the source and correspondence components and analogously for the target-identic triple rules. According to Thm. 1 below, weak invertibility is ensured if the TGG is pure and at most one set of operational rules is restricted by a conservative policy. In the more specific case that all triple rules of a TGG are creating on the source and target components ($TR = TR^{+s} = TR^{+t}$), then the TGG is called *tight*, because the derived forward and backward rules are strongly related. This additional property ensures invertibility meaning that fPpg and bPpg are inverse to each other when considering the resulting models only.

Theorem 1 (Correctness, Completeness, and Invertibility). *Let Synch(TGG) be a derived TGG synchronization framework, such that the sets of operational rules of TGG*

are deterministic. Then Synch(TGG) is correct and complete. If, additionally, TGG is pure and at most one set of operational rules was extended by a conservative policy, then Synch(TGG) is weakly invertible and if, moreover, TGG is tight and no policy was applied, then Synch(TGG) is also invertible.

Proof. The proof is given in App. A.5. □

Example 5 (Correctness, Completeness, Invertibility, and Scalability). The initially derived set of backward transformation rules for our running example is not completely deterministic because of the non-deterministic choice of base and bonus values for propagating the change of a salary value. Therefore, we defined a conservative policy for the responsible backward triple rule by fixing the propagated values of modified salary values to $bonus = base = 0.5 \times salary$. By Fact 7 in App. A, we provided a sufficient static condition for checking that a policy is conservative; we validated our example and showed that the derived operations fPpg and bPpg are deterministic. For this reason, we can apply Thm. 1 and verify that the derived TGG synchronization framework is correct and complete. Since, moreover, the TGG is pure and we used the conservative policy for the backward direction only, Thm. 1 further ensures that *Synch(TGG)* is weakly invertible. However, it is not invertible in the general sense, as shown by a counter example in App. A, which uses the fact that information about birth dates is stored in one domain only. The automated validation for our example TGG with 8 rules was performed in 25 seconds on a standard consumer notebook via the analysis engine of the tool AGG [28]. We are confident that the scalability of this approach can be significantly improved with additional optimizations.

In the case that the specified TGG does not ensure deterministic synchronization operations, there are still two options for synchronization that ensure correctness and completeness. On the one hand, the triple rules can be modified in a suitable way, such that the TGG can be verified to be deterministic. For this purpose, the critical pair analysis engine of the tool AGG [28] can be used to analyze conflicts between the generated operational rules. Moreover, backtracking can be reduced or even eliminated by generating additional application conditions for the operational rules using the automatic generation of filter NACs [17]. On the other hand, the TGG can be used directly, leading to nondeterministic synchronization operations, which may provide several possible synchronization results.

7 Related Work

Triple graph grammars have been successfully applied in multiple case studies for bidirectional model transformation, model integration and synchronization [21,26,11,10], and in the implementation of QVT [14]. Moreover, several formal results are available concerning correctness, completeness, termination [8,12], functional behavior [20,12], and optimization with respect to the efficiency of their execution [17,22,12]. The presented constructions for performing model transformations and model synchronizations are inspired by Schürr et al. [25,26] and Giese et al. [10,11], respectively. The constructions formalize the main ideas of model synchronization based on TGGs in order to

show correctness and completeness of the approach based on the results known for TGG model transformations.

Perdita Stevens developed an abstract state-based view on symmetric model synchronization based on the concept of constraint maintainers [27] and Diskin described a more general delta-based view within the *tile algebra* framework [4]. The constructions in the present paper are inspired by tile algebra and follow the general framework presented by Diskin et al. [5], where propagation operations are defined as the composition of two kinds of operations: alignment and consistency restoration. In the current paper, operations fAln and bAln take care of the alignment by removing all correspondence nodes that would be dangling due to deletions via the given model update. Then, operations Del and fAdd resp. bAdd provide the consistency restoration by first marking the consistent parts of the integrated model and then propagating the changes and deleting the remaining inconsistent parts.

Giese et al. introduced incremental synchronization techniques based on TGGs in order to preserve consistent structures of the given models by revoking previously performed forward propagation steps and their dependent ones [11]. This idea is generalized by the auxiliary operation Del in the present framework, which ensures the preservation of maximal consistent substructures and extends the application of synchronization to TGGs that are not tight or contain rules with negative application conditions. Giese et al. [10] and Greenyer et al. [15] proposed to extend the preservation of substructures by allowing for the reuse of any *partial* substructure of a rule causing, however, non-deterministic behavior. Moreover, a partial reuse can cause unintended results. Consider, e.g., the deletion of a person A in the source domain and the addition of a new person with the same name, then the old birth date of person A could be reused.

In order to improve efficiency, Giese et al. [11,10] proposed to avoid the computation of already consistent substructures by encoding the matches and dependencies of rule applications within the correspondences. In the present framework, operation Del can be extended conservatively by storing the matches and dependency information separately, such that the provided correctness and completeness results can be preserved as presented in App. A.

8 Conclusion and Future Work

Based on our formal framework for correctness, completeness, termination and functional behavior of model transformations using triple graph grammars (TGGs) [8,17], we have presented in this paper a formal TGG framework for model synchronization inspired by [11,10,25,26]. The main result (Thm. 1) shows correctness, completeness and (weak) invertibility, provided that the derived synchronization operations are deterministic. For this property, sufficient static conditions are provided in App. A based on general results for TGGs in [17].

In future work, the tool Henshin based on AGG [28] will be extended to implement the synchronization algorithm for forward propagation in Fig. 8. Moreover, the relationship with lenses [27] and delta based bidirectional transformations [5] will be studied in more detail, especially in view of composition of lenses leading to composi-

tion of synchronization operations. Furthermore, we will study synchronization based on non-deterministic forward and backward propagation operations in more detail.

References

1. Corradini, A., Hermann, F., Sobociński, P.: Subobject Transformation Systems. Applied Categorical Structures 16(3), 389–419 (February 2008), http://www.springerlink.com/content/1648jx522426p053/
2. Czarnecki, K., Foster, J., Hu, Z., Lämmel, R., Schürr, A., Terwilliger, J.: Bidirectional Transformations: A Cross-Discipline Perspective. In: Proc. ICMT'09. LNCS, vol. 5563, pp. 260–283. Springer (2009)
3. Diskin, Z., Xiong, Y., Czarnecki, K.: From State- to Delta-Based Bidirectional Model Transformations: the Asymmetric Case. Journal of Object technology 10, 6:1–25 (2011)
4. Diskin, Z.: Model Synchronization: Mappings, Tiles, and Categories. In: Generative and Transformational Techniques in Software Engineering III, LNCS, vol. 6491, pp. 92–165. Springer (2011)
5. Diskin, Z., Xiong, Y., Czarnecki, K., Ehrig, H., Hermann, F., Orejas, F.: From State- to Delta-based Bidirectional Model Transformations: The Symmetric Case. In: Proc. MODELS 2011. Springer (2011)
6. Ehrig, H., Ehrig, K., Hermann, F.: From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. EC-EASST 10 (2008)
7. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theor. Comp. Science, Springer (2006)
8. Ehrig, H., Ermel, C., Hermann, F., Prange, U.: On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars. In: Proc. MODELS'09. LNCS, vol. 5795, pp. 241–255. Springer (2009)
9. Ehrig, H., Ehrig, K., Ermel, C., Hermann, F., Taentzer, G.: Information preserving bidirectional model transformations. In: Proc. FASE'07. LNCS, vol. 4422, pp. 72–86. Springer (2007)
10. Giese, H., Hildebrandt, S.: Efficient Model Synchronization of Large-Scale Models . Tech. Rep. 28, Hasso Plattner Institute at the University of Potsdam (2009)
11. Giese, H., Wagner, R.: From model transformation to incremental bidirectional model synchronization. Software and Systems Modeling 8(1), 21–43 (2009)
12. Giese, H., Hildebrandt, S., Lambers, L.: Toward Bridging the Gap Between Formal Semantics and Implementation of Triple Graph Grammars . Tech. Rep. 37, Hasso Plattner Institute at the University of Potsdam (2010)
13. Golas, U., Ehrig, H., Hermann, F.: Formal Specification of Model Transformations by Triple Graph Grammars with Application Conditions. EC-EASST 39 (2011)
14. Greenyer, J., Kindler, E.: Comparing relational model transformation technologies: implementing query/view/transformation with triple graph grammars. Software and Systems Modeling (SoSyM) 9(1), 21–46 (2010)
15. Greenyer, J., Pook, S., Rieke, J.: Preventing information loss in incremental model synchronization by reusing elements. In: Proc. ECMFA 2011. LNCS, vol. 6698, pp. 144–159. Springer Verlag (2011)
16. Hermann, F.: Permutation Equivalence of DPO Derivations with Negative Application Conditions based on Subobject Transformation Systems. Electronic Communications of the EASST 16 (2009), http://tfs.cs.tu-berlin.de/publikationen/Papers09/Her09.pdf

17. Hermann, F., Ehrig, H., Golas, U., Orejas, F.: Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars. In: Proc. MDI'10 (2010)
18. Hermann, F., Ehrig, H., Golas, U., Orejas, F.: Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars - Extended Version. Tech. Rep. TR 2010-13, TU Berlin, Fak. IV (2010)
19. Hermann, F., Ehrig, H., Orejas, F., Czarnecki, K., Diskin, Z., Xiong, Y.: Correctness of Model Synchronization Based on Triple Graph Grammars - Extended Version. Tech. Rep. TR 2011-07, TU Berlin, Fak. IV (2011)
20. Hermann, F., Ehrig, H., Orejas, F., Golas, U.: Formal Analysis of Functional Behaviour of Model Transformations Based on Triple Graph Grammars. In: Proc. ICGT'10. LNCS, vol. 6372, pp. 155–170. Springer (2010)
21. Kindler, E., Wagner, R.: Triple graph grammars: Concepts, extensions, implementations, and application scenarios. Tech. Rep. TR-ri-07-284, Department of Computer Science, University of Paderborn, Germany (2007)
22. Klar, F., Lauder, M., Königs, A., Schürr, A.: Extended Triple Graph Grammars with Efficient and Compatible Graph Translators. In: Graph Transformations and Model Driven Enginering, LNCS, vol. 5765, pp. 141–174. Springer Verlag (2010)
23. Lambers, L.: Certifying Rule-Based Models using Graph Transformation. Ph.D. thesis, Technische Universität Berlin (2009)
24. Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Version 1.0 formal/08-04-03. http://www.omg.org/spec/QVT/1.0/ (2008)
25. Schürr, A.: Specification of Graph Translators with Triple Graph Grammars. In: Proc. WG'94. LNCS, vol. 903, pp. 151–163. Springer Verlag, Heidelberg (1994)
26. Schürr, A., Klar, F.: 15 Years of Triple Graph Grammars. In: Proc. ICGT'08. LNCS, vol. 5214, pp. 411–425 (2008)
27. Stevens, P.: Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions . Software and Systems Modeling 9, 7–20 (2010)
28. TFS-Group, TU Berlin: AGG (2011), `http://tfs.cs.tu-berlin.de/agg`

A Detailed Constructions and Proofs

This appendix A provides full technical details of the constructions for the auxiliary operations fAIn, Del and fAdd and shows the main result as stated by Thm. 1 in Sec. 6.

At first, we present in Sec. A.1 extended constructions and results for model transformations and consistency checks based on TGGs in general. Since negative application conditions (NACs) already allow the specification of expressive TGGs while still ensuring efficient executions we provide the results for transformation rules with NACs. The extension to the cases with nested application condition will be part of future work. Based on these general constructions and results, we provide in Sec. A.4 full details of the constructions concerning the auxiliary operations fAIn, Del and fAdd and show the main results.

A.1 TGG Operations and Analysis

This section first reviews further details on model transformations based forward rules and the equivalent approach based on forward translation rules, which extend forward rules by additional translation attributes. Forward translation rules are used for efficiently execute and analyze TGG model transformations as presented in [17].

According to Sec. 4 and Def. 5 below, model transformations $MT : VL(TG^S) \Rrightarrow VL(TG^T)$ based on forward rules map valid source models to target models by computing TGT sequences via the derived forward rules, such that the additional control condition source consistency is respected.

Definition 5 (Model Transformation based on Forward Rules). *A* model transformation sequence $(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T)$ *consists of a source graph* G^S*, a target graph* G^T*, and a source consistent forward TGT-sequence* $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ *with* $G^S = G_0^S$ *and* $G^T = G_n^T$*. A* model transformation $MT : VL(TG^S) \Rrightarrow VL(TG^T)$ *is defined by all model transformation sequences* $(G^S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G^T)$ *with* $G^S \in VL(TG^S)$ *and* $G^T \in VL(TG^T)$*.*

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

Definition 6 (Almost Injective Match). *An attributed triple graph morphism* $m : L \to G$ *is called* almost injective match*, if it is non-injective at most for the set of variables and data values.*

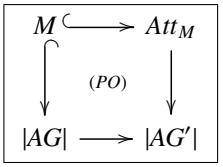
Using the concept of translation attributes we provide extended operational rules, such that transformations via these rules do not need to be controlled by a separate control condition. The operational rules TR_{FT} (forward translation rules), TR_{BT} (backward translation rules) and TR_{CC} (consistency creating rules) are derived from the set TR of TGG-triple rules by an extension with Boolean valued marking attributes with prefix "`tr`" as introduced for forward translation rules TR_{FT} in [17]. For this purpose, we review the constructions for triple graphs with translation attributes.

Given an attributed graph AG and a family of subsets $M \subseteq AG$ for nodes and edges, we call AG' a graph with translation attributes over AG if it extends AG with one Boolean-valued attribute `tr_`x for each element x (node or edge) in M and one Boolean-valued attribute `tr_`x`_`a for each attribute associated to such an element x in M. The family M together with all these additional translation attributes is denoted by Att_M. Note that we use the attribution concept of E-graphs as presented in [7], where attributes are possible for nodes and edges. An E-graph EG extends a directed graph $G = (V, E, (s, t : E \to V))$ by a set of attribute value nodes V_D together with sets of attribution edges E_{NA} and E_{EA} for assigning attribute values to structural graph nodes V and edges E. An attributed graph $AG = (G, D)$ is given by an E-graph G together with a data algebra D, such that the attribute values V_D are given by the disjoint union of the carrier sets of D.

Definition 7 (Family with Translation Attributes). *Given an attributed graph $AG = (G, D)$ we denote by $|AG| = (V_G^G, V_G^D, E_G^G, E_G^{NA}, E_G^{EA})$ the underlying family of sets containing all nodes and edges. Let $M \subseteq |AG|$ with $(V_M^G, V_M^D, E_M^G, E_M^{NA}, E_M^{EA})$, then a* family with translation attributes *for (AG, M) extends M by additional translation attributes and is given by $Att_M = (V_M^G, V_M^D, E_M^G, E^{NA}, E^{EA})$ with:*

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

Definition 8 (Graph with Translation Attributes). *Given an attributed graph $AG = (G, D)$ and a family of subsets $M \subseteq |AG|$ with $\{\mathbf{T}, \mathbf{F}\} \subseteq V_M^D$ and let Att_M be a family with translation attributes for (G, M) according to Def. 7. Then, $AG' = (G', D)$ is a* graph with translation attributes *over AG, where the domains $|AG'|$ of AG' are given by the gluing via pushout of $|AG|$ and Att_M over M and the source and target functions of G' are defined as follows:*



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

Att_M^v, where $v = \mathbf{T}$ or $v = \mathbf{F}$, denotes a family with translation attributes where all attributes are set to v. Moreover, we denote by $AG \oplus Att_M$ that AG is extended by the

translation attributes in Att_M, i.e. $AG \oplus Att_M = (G', D)$ for $AG' = (G', D)$ as defined above. We use the notion $AG \oplus Att^v_M$ for translation attributes with value v and we use the short notion $Att^v(AG) := AG \oplus Att^v_{|AG|}$.

The concept of forward translation rules, which we introduced in [20], extends the construction of forward rules by additional translation attributes in the source component. The translation attributes are used to keep track of the elements that have been translated so far. This way, we can ensure that each element in the source graph is not translated twice, but exactly once. At the beginning, the source model of a model transformation sequence is extended by translation attributes that are all set to "**F**" and they are set to "**T**" when their containing elements are translated by a forward translation rule. Whenever the model transformation stops at a model whose translation attributes are all set to "**T**", the sequence specifies a source consistent forward sequence by removing all translation attributes and a valid target model is obtained from the resulting triple graph. Due to the modification of the translation attributes, the rules are deleting and thus, the triple transformations are extended from a single (total) pushout to the classical double pushout (DPO) approach [7]. We call these rules forward translation rules, because pure forward rules need to be controlled by the additional control condition source consistency [9].

Consistency creating rules are used to compute maximal subgraphs G_k of a given triple graph G typed over TG, such that $G_k \in VL(TGG)$. In the special case that $G \in VL(TGG)$ we have that $G_k \cong G$. For this purpose, each consistency creating rule switches labels form false to true for those elements that would be created by the corresponding TGG-rule in TR. This means that elements in the left hand side $L_{CC} = R$ are labelled with true, if they are also contained in L, and they are labelled with false otherwise. Accordingly, all elements in the right hand side R_{CC} are labelled with true.

	main components	new NAC for each $n : L \to N$ of tr
tr_{CC}	$L_{CC} \xleftarrow{l_{CC}} K_{CC} \xhookrightarrow{r_{CC}} R_{CC}$ $\| \quad \| \quad \|$ $(R \oplus Att^{\mathbf{T}}_L \oplus Att^{\mathbf{F}}_{R\setminus L}) \quad (R \oplus Att^{\mathbf{T}}_L) \quad (R \oplus Att^{\mathbf{T}}_L \oplus Att^{\mathbf{T}}_{R\setminus L})$	$N_{CC} = (L_{CC} +_L N) \oplus Att^{\mathbf{T}}_{N\setminus L}$
tr_{FT}	$L_{FT} \xleftarrow{l_{FT}} K_{FT} \xhookrightarrow{r_{FT}} R_{FT}$ $\| \quad \| \quad \|$ $(L_F \oplus Att^{\mathbf{T}}_{L_S} \oplus Att^{\mathbf{F}}_{R_S\setminus L_S}) \quad (L_F \oplus Att^{\mathbf{T}}_{L_S}) \quad (R_F \oplus Att^{\mathbf{T}}_{L_S} \oplus Att^{\mathbf{T}}_{R_S\setminus L_S})$	$N_{FT} = (L_{FT} +_L N) \oplus Att^{\mathbf{T}}_{N_S\setminus L_S}$
tr_{BT}	$L_{BT} \xleftarrow{l_{BT}} K_{BT} \xhookrightarrow{r_{BT}} R_{BT}$ $\| \quad \| \quad \|$ $(L_B \oplus Att^{\mathbf{T}}_{L_T} \oplus Att^{\mathbf{F}}_{R_T\setminus L_T}) \quad (L_B \oplus Att^{\mathbf{T}}_{L_T}) \quad (R_B \oplus Att^{\mathbf{T}}_{L_T} \oplus Att^{\mathbf{T}}_{R_T\setminus L_T})$	$N_{BT} = (L_{BT} +_L N) \oplus Att^{\mathbf{T}}_{N_T\setminus L_T}$

Fig. 9. Components of derived operation rules

Definition 9 (Operational Rules). *Given a triple rule* $tr = (L \to R)$ *and its derived source rule* $tr_S = (L_S \to R_S)$*, target rule* $tr_T = (L_T \to R_T)$*, forward rule* $tr_F = (L_F \to R_F)$ *and backward rule* $tr_B = (L_B \to R_B)$*, the derived translation rules of tr are given by* consistency creating rule $tr_{CC} = (L_{CC} \xleftarrow{l_{CC}} K_{CC} \xrightarrow{r_{CC}} R_{CC})$*,* forward translation rule $tr_{FT} = (L_{FT} \xleftarrow{l_{FT}} K_{FT} \xrightarrow{r_{FT}} R_{FT})$*, and* backward translation rule $tr_{BT} = (L_{BT} \xleftarrow{l_{BT}} K_{BT} \xrightarrow{r_{BT}} R_{BT})$ *defined in Fig. 9 using the notation of Def. 8. By* TR_{CC}*,* TR_{FT}*,* TR_{BT} *we denote the sets of all derived consistency creating, forward translation and backward translation rules, respectively.*

Remark 5. Note that $(B +_A C)$ is the union of B and C with shared A, such that for instance $(L_{FT} +_L N)$ is the union of L_{FT} and N with shared L.

Since forward translation rules are deleting only on attribution edges, the gluing condition is always satisfied, because dangling points and identification points are preserved for almost injective matches. Now, we define model transformations based on forward translation rules in the same way as for forward rules in Def. 5, where source consistency of the forward sequence is replaced by completeness of the forward translation sequence. As shown in [17], model transformation sequences based on forward and forward translation rules, respectively, are equivalent. This ensures that the derived model transformation relations are the same.

Definition 10 (Complete Forward Translation Sequence). *A forward translation sequence* $G_0 \overset{tr^*_{FT}}{\Longrightarrow} G_n$ *with almost injective matches is called* complete *if* G_n *is* completely translated*, i.e. all translation attributes of* G_n *are set to true ("*$\mathbf{T}$*").*

Definition 11 (Model Transformation Based on Forward Translation Rules). *A* model transformation sequence $(G^S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G^T)$ *based on forward translation rules* 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* $G'_0 = (Att^{\mathbf{F}}(G^S) \leftarrow \varnothing \to \varnothing)$ *and* $G'_n = (Att^{\mathbf{T}}(G^S) \leftarrow G^C \to G^T)$.
A model transformation $MT : VL(TG^S) \Rrightarrow VL(TG^T)$ *based on* TR_{FT} *is defined by all model transformation sequences as above with* $G^S \in VL(TG^S)$ *and* $G^T \in VL(TG^T)$*. All the corresponding pairs* (G^S, G^T) *define the* model transformation relation $MTR_{FT} \subseteq VL(TG^S) \times VL(TG^T)$ *based on* TR_{FT}*. The model transformation is* terminating *if there are no infinite TGT-sequences via* TR_{FT} *starting with* $G'_0 = (Att^{\mathbf{F}}(G^S) \leftarrow \varnothing \to \varnothing)$ *for some source graph* $G^S \in VL(TG^S)$*.*

Consistency creating sequences as defined in Def. 12 below, are used for computing a maximal consistent part of a given triple graph, which is used for the auxiliary operation Del. A consistency creating sequence starts at a triple graph $G'_0 = Att^{\mathbf{F}}(G)$, i.e. at a triple graph where all elements are marked with $\mathbf{F}$. Each application of a consistency creating rule modifies some translation attributes of an intermediate triple graph G'_i from $\mathbf{F}$ to $\mathbf{T}$ and preserves the structural part G contained in G'_i. Therefore, the the resulting triple graph G'_n extends G with translation attributes only, i.e. some are set to $\mathbf{T}$ and the remaining ones to $\mathbf{F}$.

Definition 12 (Consistency Creating Sequence). *Given a triple graph grammar $TGG = (TG, \emptyset, TR)$, a triple graph G typed over TG and let TR_{CC} be the set of consistency creating rules of TR. A* consistency creating sequence $s = (G, G'_0 \overset{tr^*_{CC}}{\Longrightarrow} G'_n, G_n)$ *is given by a TGT sequence $G'_0 \overset{tr^*_{CC}}{\Longrightarrow} G'_n$ via TR_{CC} with $G'_0 = Att^{\mathbf{F}}(G)$ and $G'_n = G \oplus Att^{\mathbf{T}}_{G_n} \oplus Att^{\mathbf{F}}_{G \setminus G_n}$, where G_n is the subgraph of G derived from $G'_0 \overset{tr^*_{CC}}{\Longrightarrow} G'_n$ by restricting G'_n to all* **T***-marked elements. Consistency creating sequence s is called* terminated*, if there is no rule in TR_{CC} which is applicable to the result graph G'_n. In this case, the triple graph G'_n is called a maximal consistency marking of G. A triple graph G' is called* completely **T**-marked*, if $G' = Att^{\mathbf{T}}(G)$ for a given triple graph G, i.e. all translation attributes in G' are "***T***".*

Remark 6 (Termination). In order to ensure termination, the operational rules TR^{+s}_{FT} for forward transformations are derived from a subset TR^{+s} of source creating triple rules of TR given by $TR^{+s} = \{tr \in TR \mid tr^S \neq id\}$. Accordingly, we consider a subset TR^{+t} of target creating triple rules of TR for the operational rules TR^{+t}_{BT} for backward transformations, i.e. $TR^{+t} = \{tr \in TR \mid tr^T \neq id\}$. Let TR be a set of triple rules. We distinguish the following subsets.

- The set of *source creating* rules $TR^{+s} = \{tr \in TR \mid tr^S \neq id\}$,
- The set of *source identic* rules $TR^{1s} = \{tr \in TR \mid tr^S = id\}$,
- The set of *target creating* rules $TR^{+t} = \{tr \in TR \mid tr^T \neq id\}$,
- The set of *target identic* rules $TR^{1t} = \{tr \in TR \mid tr^T = id\}$, and
- The set of *identic* rules $TR^{1} = \{tr \in TR \mid tr = id\}$.

The consistency creating rules TR_{CC} are derived from $TR \setminus TR^1$. Since TR^{+s} and TR^{+t} usually differ from each other, there is usually only a loose correspondences between operational rules in TR^{+s}_{FT} and TR^{+t}_{BT}. In the special case that all triple rules in a TGG are creating on the source and target components we have $TR = TR^{+s} = TR^{+t}$. In particular, this means that for each triple rule tr there is a derived forward translation rule $tr_{FT} \in TR^{+s}_{FT}$ and a derived backward translation rule $tr_{BT} \in TR^{+t}_{BT}$. In this case, the TGG is called tight.

Definition 13 (Tight TGG). *A TGG $TGG = (TG, \emptyset, TR)$ is called* tight*, if the sets of source and target creating rules TR^{+s} and TR^{+t} coincide with the set of triple rules TR, i.e. $TR = TR^{+s} = TR^{+t}$.*

A.2 General Results for Transformation Systems

In order to compose the auxiliary operations Del and fPpg we first show two fundamental results for transformation systems based on the notion of sequential independence. At first we characterize sequential independence of rules. Sequential independence of a pair (p_1, p_2) of rules means that any two subsequence transformation steps with p_1 applied in the first and p_2 applied in the second step are sequentially independent, i.e. the rules are sequentially independent in any context concerning the specified order.

Definition 14 (Independence of Rules). *A pair of rules* (p_1, p_2) *is called* sequentially independent*, if any two transformation steps* $G_0 \overset{p_1,m_1}{\Longrightarrow} G_1 \overset{p_2,m_2}{\Longrightarrow} G_2$ *are sequentially independent.*

The following two facts are required in order to use AGG for static analysis of sequential independence based on critical pairs for parallel independence [7]. This is important, because we need to ensure sequential independence of the source-identic FT-rules TR^{1s}_{FT} towards the effective FT-rules TR^{+s}_{FT} in order to ensure termination of the propagation process. The key idea for this purpose is that the termination critical source identic FT-rules do not need to be applied, because they do not change any translation attribute and we require that they are sequentially independent from the effective FT-rules. Hence, we show by Fact 3 that sequentially independent rules can be shifted to the end.

Fact 2 (Characterization of Independent Rules). *A pair of rules* (p_1, p_2) *is sequentially independent if and only if there is no critical pair for* (p_1^{-1}, p_2)*, where* $p^{-1} = ((R \overset{r}{\leftarrow} K \overset{l}{\rightarrow} L), N')$ *denotes the inverted rule of the rule* $p = ((L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R), N)$ *and* N' *is obtained by shifting all NACs in* N *over the rule* p*.*

Proof. **Direction "⇐":** By contraposition, assume (r_1, r_2) is a pair of sequentially dependent rules. This implies that there are the sequentially dependent transformation steps $G_0 \overset{r_1,m_1}{\Longrightarrow} G_1 \overset{r_2,m_2}{\Longrightarrow} G_2$. This means that the steps $G_0 \overset{r_1^{-1},n_1}{\Longleftarrow} G_1 \overset{r_2,m_2}{\Longrightarrow} G_2$ are parallel dependent, where n_1 is the comatch of $G_0 \overset{r_1,m_1}{\Longrightarrow} G_1$. Thus, by completeness of critical pairs (Thm. 3.7.6 in [23]) there is a critical pair for (r_1^{-1}, r_2).
Direction "⇒": By contraposition, assume there is a critical pair for (r_1^{-1}, r_2), i.e. there are parallel dependent transformation steps $(P_1 \overset{r_1^{-1},m_1}{\Longleftarrow} K \overset{r_2,m_2}{\Longrightarrow} P_2)$. This implies that there are sequentially dependent transformation steps $(P_1 \overset{r_1,n_1}{\Longrightarrow} K \overset{r_2,m_2}{\Longrightarrow} P_2)$, where n_1 is the comatch of $K \overset{r_1^{-1},m_1}{\Longrightarrow} P_1$. Therefore, (r_1, r_2) is a pair of dependent rules.

Fact 3 (Shifting of Independent Steps). *Given two sets* P_1 *and* P_2 *of rules such that each pair* $(p_1, p_2) \in P_1 \times P_2$ *is sequentially independent. Then, there is a transformation sequence* $(G \overset{r^*}{\Longrightarrow} H)$ *via* $(P_1 \cup P_2)$ *if and only if there are transformation sequences:* $s_1 = (G \overset{p^*}{\Longrightarrow} G_1)$ *via* P_2 *and* $s_2 = (G_1 \overset{q^*}{\Longrightarrow} H)$ *via* P_1 *with same* G_1*.*

Proof. By Def. 14, each neighbouring pair of steps via first rule $q \in P_1$ and then $p \in P_2$ is sequentially independent. Therefore, we can stepwise apply the Local Church-Rosser Theorem and switch the steps via P_2 forward in s and derive the sequences s_1 and s_2. For the inverse direction, the sequences s_1 and s_2 can be composed leading to sequence s via $(P_1 \cup P_2)$.

A.3 Results for TGGs

Based on the following Fact 4, we can ensure termination for forward translation sequences, because the source-identic rules can be neglected, since they can be applied equivalently at the end and do not change the source model.

Fact 4 (Shifting of Independent FT-Steps). *Let $TGG = (TG, \emptyset, TR)$ be a triple graph grammar, $TR_1 \subseteq TR$ be a subset of triple rules and $\overline{TR_1} = TR \setminus TR_1$ be the complement of TR_1 with respect to TR. Let TR_{FT}, $TR_{1,FT}$ and $\overline{TR_1}_{,FT}$ be the derived forward translation rules of TR, TR_1 and $\overline{TR_1}$, respectively. If there is no pair $(\overline{tr}_{FT}, tr_{FT}) \in \overline{TR_1}_{,FT} \times TR_{1,FT}$ of sequentially dependent rules, then the following are equivalent for almost injective matches*

1. *There is a TGT-sequence $H'_k \overset{tr^*_{FT}}{\Longrightarrow} H'_n$.*
2. *There are TGT-sequences $H'_k \overset{tr^*_{1,FT}}{\Longrightarrow} H'_l$ via $TR_{1,FT}$ and $H'_l \overset{\overline{tr}^*_{1,FT}}{\Longrightarrow} H'_n$ via $\overline{TR_1}_{,FT}$.*

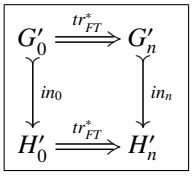
Proof. The result follows directly by Fact 3. □

Fact 5 (Completeness of Model Transformations Based on Reduced Operational Rules). *Let TGG be a triple graph grammar and TR^{+s}_{FT} and TR^{+t}_{BT} the operational rules derived from the source creating triple rules TR^{+s} resp. the target creating triple rules TR^{+t} of TR, such that the following conditions hold. There is no pair $(tr_1, tr_2) \in (TR^{1s}_{FT} \times TR^{+s}_{FT})$ of sequentially dependent rules and there is no pair $(tr_1, tr_2) \in (TR^{1t}_{BT} \times TR^{+t}_{BT})$ of sequentially dependent rules. Then, the derived model transformation relations $MTR^{+s}_{FT} \subseteq VL_S \times VL_T$ via TR^{+s}_{FT} and $MTR^{+t}_{BT} \subseteq VL_T \times VL_S$ via TR^{+t}_{BT} are left total.*

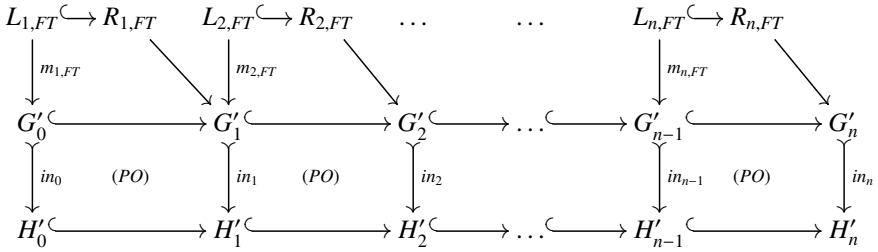
Proof. By Thm. 1 in [17] the model transformation relation MTR_{FT} derived from the complete set TR_{FT} of forward translation rules is left total. Let $(G^S, G^T) \in MTR_{FT}$, then there is a model transformation sequence $(G^S, G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n, G^T)$ via TR_{FT}. Using the preconditions concerning sequentially independence of the rules we can apply Fact 4 leading to the sequences $s_1 = (G'_0 \overset{tr^*_{1,FT}}{\Longrightarrow} G'_k)$ via TR^{+s}_{FT} and $s_2 = (G'_k \overset{\overline{tr}^*_{1,FT}}{\Longrightarrow} G'_n)$ via TR^{1s}_{FT}. Sequence s_1 is source consistent, because the complete sequence $(s_1; s_2)$ is source consistent and the corresponding source sequence for s_2 contains only identical steps. Therefore, (G^S, s_1, G^T_k) is a model transformation sequence, where $G'_k = (Att^{\mathbf{T}}(G^S) \leftarrow G^C_k \rightarrow G^T_k)$. This means that $(G^S, G^T_k) \in MTR^{+s}_{FT}$ and thus, MTR^{+s}_{FT} is left total.

The result for MTR^{+s}_{BT} follows analogously due to the symmetric definition of TGGs.

Fact 6 (Extension of FT-sequences). *Let $TGG = (TG, \emptyset, TR)$ be a triple graph grammar with derived forward translation rules TR_{FT}. Let $s_1 = (G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n)$ be a TGT-sequence via TR_{FT} with almost injective matches, where $G_0 = (G^S \leftarrow \emptyset \rightarrow \emptyset)$ $G'_0 = Att^{\mathbf{F}}(G_0)$, and $G'_n = G_n \oplus Att^{\mathbf{T}}_{G^S_0}$. Let $in_0 : G_0 \rightarrow H_0$ be an injective embedding with $H_0 = (H^S \leftarrow \emptyset \rightarrow \emptyset) \in VL(TG)$ and let $H'_0 = Att^{\mathbf{F}}(H_0)$. Then, there is a sequence of injective embeddings $(in_i : G'_i \rightarrow H'_i)_{(i=1..n)}$ and a TGT-sequence $s_2 = (H'_0 \overset{tr^*_{FT}}{\Longrightarrow} H'_n)$ via TR_{FT} with almost injective matches $m'_{i,FT} : L_{i,FT} \rightarrow H'_{i-1}$ given by $m'_{i,FT} = in_i \circ m_{i,FT}$, such that $H'^S_n = H^S_0 \oplus Att^{\mathbf{T}}_{|G^S_0|} \oplus Att^{\mathbf{F}}_{|H^S_n| \setminus |G^S_0|}$, $H'^C_n = G'^C_n$ and $H'^T_n = G'^T_n$.*



Proof. The almost injective match $m_{1,FT} : L_{1,FT} \to G'_0$ of the first step $G'_0 \overset{tr_{1,FT}, m_{1,FT}}{\Longrightarrow} G'_1$ is extended via the injective morphism $in_0 : G_0 \rightarrowtail H_0$ leading to an almost injective match $m'_{1,FT} = in_0 \circ m_{1,FT} : L_{1,FT} \to H_{0,FT}$.



We show that the rule $tr_{1,FT}$ is applicable at $m'_{1,FT}$. By Fact 2 in [17] the gluing condition is always satisfied for almost injective matches. Let $(n_{FT} : L_{1,FT} \to N)$ be a NAC of $tr_{1,FT}$. The NAC n_{FT} is satisfied by $m_{1,FT}$ and the NAC-only elements $(N \setminus n_{FT}(L_{1,FT}))$ in n_{FT} are labeled with "**T**" (see Fig. 9). The only way to derive a compatible occurrence of N in an extension of G'_0 would require that additional "**T**"-labeled elements are added. All translation attributes in $H'_0 \setminus in_0(G'_0)$ are set to "**F**". Therefore, $m'_{1,FT} \models n_{FT}$ leading to the transformation step $H'_0 \overset{tr_{1,FT}, m'_{1,FT}}{\Longrightarrow} H'_1$ with almost injective match $m'_{1,FT}$ and injective morphism $in_1 : G'_1 \rightarrowtail H'_1$. Moreover, all translation attributes in $H'_1 \setminus in_1(G'_1)$ are set to "**F**", because the match $m'_{1,FT}$ does not reach these elements and they are preserved.

This procedure is repeated for the subsequent steps leading to $s_2 = (H'_0 \overset{tr^*_{FT}}{\Longrightarrow} H'_n)$, such that all translation attributes in $H'_n \setminus in_n(G'_n)$ are set to "**F**". Recall that $G'_n = G_n \oplus Att^{\mathbf{T}}_{G^S_0}$ by general assumption. This implies that $H'^S_n = H^S_0 \oplus Att^{\mathbf{T}}_{|G^S_0|} \oplus Att^{\mathbf{F}}_{|H^S_n| \setminus |G^S_0|}$. Each in^C_i and in^T_i is an isomorphism using that $in^C_0 = in^T_0 : \varnothing \to \varnothing$ and the preservation of isomorphisms along pushouts. This implies that $H'^C_n = G'^C_n$ and $H'^T_n = G'^T_n$. □

Since transformation systems are not deterministic in general, we introduce the concept of policies in order to obtain deterministic sets of operational rules for the synchronization operations. The main idea is to restrict the matches of a transformation rule using additional attribute conditions in order to eliminate ambiguous results.

An attribute condition *attCon* for a (triple) rule $tr : L \to R$ is a set of equations for attribute values. A match $m : L \to G$ satisfies *attCon*—written $m \models attCon$—if the evaluation of attribute values satisfies each equation. In our case study, we use one attribute condition.

Definition 15 (Policy for Operational Rules). *Given a TGG and let TR_{FT} be the derived set of forward translation rules. A* policy *$pol : TR_{FT} \to TR'_{FT}$ for restricting the applications of the rules in TR_{FT} maps each rule $tr_{FT} \in TR_{FT}$ to an extended rule $tr'_{FT} \in TR'_{FT}$, where tr'_{FT} is given by tr_{FT} extended by a set of additional attribute conditions $AttC_{pol}(tr_{FT})$. The policy pol is called* conservative*, if the derived model transformation relation $MTR'_{FT} \subseteq VL_S \times VL_T$ based on TR'_{FT} is left total and is contained in the model transformation relation MTR_{FT} derived from TR_{FT}, i.e. $MTR'_{FT} \subseteq MTR_{FT}$.*

A policy for backward translation rules TR_{BT} is defined analogously by replacing FT with BT and it is conservative if the derived model transformation relation $MTR'_{BT} \subseteq VL_T \times VL_T$ is left total and contained in MTR_{BT}.

Fact 7 (Conservative Policy). *Given a policy pol* : $TR_{FT} \to TR'_{FT}$*, such that for each rule* $tr'_{FT} = pol(tr_{FT})$ *in* TR'_{FT} *with* $tr : L \to R$ *the following condition holds.*

- *Given a match* $m : L \to G$ *for* tr_{FT}*, then there is also a match* $m' : L \to G$ *for* tr'_{FT} *satisfying* $AttC_{pol}$*.*
- *If* $AttC_{pol}(tr_{FT}) \neq \emptyset$*, then for each rule* $tr_2 \in TR_{FT}$ *with* $tr_{FT} \neq tr_2$ *the pair* (tr_{FT}, tr_2) *is sequentially independent according to Def. 14.*

Then, the policy pol is conservative (see Def. 15). A similar fact holds for a policy $pol : TR_{BT} \to TR'_{BT}$ *concerning backward translation rules.*

Proof. First of all, $MTR'_{FT} \subseteq MTR_{FT}$, because the additional attribute conditions only restrict the possible transformation sequences and no additional ones are possible. The relation MTR_{FT} is left total by the completeness result for TGGs according to Thm. 1 in [17]. It remains to show that $MTR'_{FT} \subseteq VL_S \times VL_T$ is left total as well.

Let $G^S \in VL_S$. Since MTR_{FT} is left total this implies that there is a complete forward translation sequence $s = (G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'_n)$ via TR_{FT} (Def. 10). This means that $G'_n = G'_n \oplus Att^{\mathbf{T}}(G_0)$, i.e., ${G'}^S_n$ is completely marked with **T**. It remains to show that there is a complete forward translation sequence $s' = (G'_0 \overset{tr^*_{FT}}{\Longrightarrow} \overline{H}'_n)$ via TR'_{FT}.

Let $pol(tr_{FT}) = tr'_{FT}$ with $AttC_{pol} \neq \emptyset$ and let $TR_{1,FT} = TR_{FT} \setminus \{tr_{FT}\}$. Using the second item of the precondition and Fact 4 we derive from sequence s two subsequences $s_1 = (G'_0 \overset{tr^*_{1,FT}}{\Longrightarrow} G'_k)$ via $TR_{1,FT}$ and $s_2 = (G'_k \overset{\overline{tr}^*_{1,FT}}{\Longrightarrow} G'_n)$ via $\{tr_{FT}\}$. By the first item of the precondition we derive for the last step in sequence s_2 that there is a corresponding step $G'_{n-1} \overset{tr'_{FT},m'}{\Longrightarrow} H'_n$ via tr'_{FT}, but with possibly different match m' leading to a new sequence s_3 by replacing in s_2 the last step with the new one. Moreover, the rules tr'_{FT} and tr_{FT} change the same amount of translation attributes from **F** to **T** and do not perform other changes on translation attributes. This implies that ${H'}^S_n = {G'}^S_n$, because the source component of a forward translation rule modifies translation attributes only. Therefore, ${H'}^S_n$ is completely marked with **T**, because ${G'}^S_n$ is completely marked with **T**.

Now, recall that the policy *pol* eliminates some possible matches for the forward translation rules, but does not lead to new ones, because it defines additional attribute conditions only. Therefore, no additional dependencies between rules are caused by *pol*. Thus, item 2 of the precondition implies the following property $(*)$: If $AttC_{pol}(tr_{FT}) \neq \emptyset$, then for each rule $tr_2 \in TR_{FT}$ with $tr_{FT} \neq tr_2$ the pair (tr_{FT}, tr_2) and the pair $(tr_{FT}, pol(tr_2))$ is sequentially independent according to Def. 14.

By property $(*)$, the other steps via tr_{FT} in s_2 can be shifted beyond the new step $(G'_{n-1} \overset{tr'_{FT},m'}{\Longrightarrow} H'_n)$ as described for the last step before. They are iteratively replaced with their corresponding steps via tr'_{FT} leading to a sequence s'_2 via $\{tr'_{FT}\}$. The resulting last triple graph of the sequence is again completely marked with **T** as explained before for the first replacement of a step. By composing sequences s_1 and s'_2 we derive the new sequence s'. This procedure for tr_{FT} can be repeated and applied to the derived new sequences s' for each rule in TR'_{FT} with $AttC_{pol} \neq \emptyset$. Therefore, we derive a sequence $\overline{s} : G'_0 \overset{tr'^*_{FT}}{\Longrightarrow} \overline{H}'_n$ via TR'_{FT}. Moreover, ${\overline{H}'}^S_n = {G'}^S_n$ is completely marked with **T**, because each modification of the sequence s preserved this property. By definition, this implies

that $\overline{s}$ is a complete forward translation sequence. Finally, $\overline{s}$ specifies a valid sequence via TR_{FT}, because only additional attribute conditions are added to the rules via *pol*. Therefore, $(G^S, H^T) \in MTR'_{FT}$.

Concerning backward translation rules the proof is analogous, where *FT* is replaced by *BT*. □

Definition 16 (Deterministic Sets of Operational Rules and Tight TGG). *Let $TGG = (TG, \emptyset, TR)$ be a triple graph grammar leading to the following derived sets of operational rules with translation attributes.*

- *Consistency creating rules TR_{CC} derived from TR*
- *Forward translation rules TR^{+s}_{FT} derived from TR^{+s}*
- *Backward translation rules TR^{+t}_{BT} derived from TR^{+t}*

Let the pairs $(TR^{1s}_{FT}, TR^{+s}_{FT})$ and $(TR^{1t}_{BT}, TR^{+t}_{BT})$ be sequentially independent acc. to Def. 14. Then the sets of operational rules TR_{CC}, TR_{FT}, and TR_{BT} (possibly extended by conservative policies) are called deterministic*, if they have functional behavior and do not require backtracking. A TGG is called* tight*, if each triple rule $tr \in TR$ is source and target creating, i.e. $TR = TR^{+s} = TR^{+t}$.*

In order to check that the sets of operational rules are deterministic we first describe how the preconditions of Def. 16 are checked using the tool AGG. Moreover, we can apply the presented results for showing that the derived model transformation relations are left total.

1. Sequential independence of the pairs $(TR^{1s}_{FT}, TR^{+s}_{FT})$ and $(TR^{1t}_{BT}, TR^{+t}_{BT})$: we can use the tool AGG for the analysis of rule dependencies based on the generation of critical pairs according to Fact 2.
2. Applied policies are conservative: We apply Fact 7. This requires that the additional application conditions according to the policy restricts the evaluation of attribute values only, i.e., the assignement of variables. We have to show that the existence of matches is preserved for each rule and that other rules are not sequentially dependent. For the latter, we can again use the tool AGG and validate that the corresponding table entries show the value 0. The preservation of the existence of matches can be ensured by checking that the affected variables are free in the unmodified rule (tr_{FT} or tr_BT), i.e. they are not part of a term that is connected to a node in the LHS (L_{FT} or L_{BT}).
3. Left totality of derived model transformation relations: By Fact 5 we can conclude that the derived model transformation relations MTR_{FT} (for TR^{+s}_{FT}) and MTR_{BT} (for TR^{+t}_{BT}) are left total. According to Def. 15, the conservative policy ensures that the derived model transformation relations MTR'_{FT} and MTR'_{BT} are left total.

We now show how to check that the derived sets of operational rules have functional behavior and do not require backtracking. For this purpose, we apply results known for the analysis of confluence of transformation systems. A transformation system is confluent, if it is locally confluent and terminating.

We generally assume that the input models are finite on the structure part, i.e. the carrier sets of the data values can be infinite, but the graph nodes and all sets of edges are

finite. As shown in [17] for forward translation rules, this allows to ensure termination if all operational rules modify at least one translation attribute. The reason is that an operational rule may change the value of a translation attribute from **F** to **T**, but not vice versa. Moreover, the amount of elements that are marked with translation attributes is not changed by any operational rule and the input models are finite.

Fact 8 (Termination). *Let $TGG = (TG, \emptyset, TR)$ be a triple graph grammar and $TR_{CC}.TR_{FT}, TR_{BT}$ are the derived sets of operational rules for consistency creating, forward translation and backward translation, respectively according to Def. 9 and possibly extended by some policies. Then, the transformation systems $TR_{CC}.TR_{FT}, TR_{BT}$ are terminating for any finite input triple graph.*

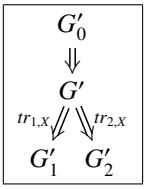
Proof. According to Def. 9, all triple rules that are identical on the translation attributes are neglected for the sets of operational rules. Therefore, each operational rule turns at last one translation attribute value from **F** to **T** and none of them creates a new translation attribute. Using the general assumption that all input graphs are finite on the graph part, we can deduce that termination is ensured, because each rule reduces the amount of translation attributes with value **F** and the start triple graph contains finitely many translation attributes.

In order to show local confluence, it is sufficient to analyze all critical pairs of the transformation system. A critical pair specifies a conflict of two rules in minimal context. The aim is to find a compatible solution for each conflict. This means that, given two dependent and diverging transformation steps described by the critical pair, then there shall be compatible merging transformation sequences leading to the same resulting triple graph.

In the context of model transformations it is sufficient to consider significant critical pairs only as shown in [17]. A critical pair is significant, if the diverging transformation steps can be embedded in transformation sequences starting at a graph that can be a possible input. For this purpose, we denote by $\mathcal{L}_S \subseteq VL(TG^S)$ the source domain language and by $\mathcal{L}_T \subseteq VL(TG^T)$ the target domain language. We usually assume $\mathcal{L}_S \subseteq VL_S$ and $\mathcal{L}_T \subseteq VL_T$.

Definition 17 (Significant Critical Pair). *A critical pair $p = (P_1 \overset{tr_{1,X}}{\Longleftarrow} K \overset{tr_{2,X}}{\Longrightarrow} P_2)$ for a set TR_X of operational triple rules is called* significant*, if it can be embedded into a parallel dependent pair $(G'_1 \overset{tr_{1,X}}{\Longleftarrow} G' \overset{tr_{2,X}}{\Longrightarrow} G'_2)$, such that one of the following conditions hold depending on X.*

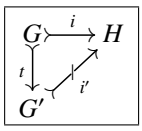
- *Consistency creating rules (X = CC): no restriction*
- *Forward translation rules (X = FT): there is $G^S \in \mathcal{L}_S$ and $G'_0 \overset{tr^*_{FT}}{\Longrightarrow} G'$ with $G'_0 = (Att^{\mathbf{F}}(G^S) \leftarrow \emptyset \rightarrow \emptyset)$*
- *Backward translation rules (X = BT): there is $G^T \in \mathcal{L}_T$ and $G'_0 \overset{tr^*_{BT}}{\Longrightarrow} G'$ with $G'_0 = (\emptyset \leftarrow \emptyset \rightarrow Att^{\mathbf{F}}(G^T))$*



Fact 9 (Functional Behavior and Backtracking). *Let $TGG = (TG, \emptyset, TR)$ be a triple graph grammar and $TR_{CC}.TR_{FT}, TR_{BT}$ are the derived sets of operational rules for consistency creating, forward translation and backward translation, respectively, such that termination is ensured for finite triple graphs. The computation of a consistency creating sequence (Def. 12) based on TR_{CC} has functional behavior and does not require backtracking, if all significant critical pairs of TR_{CC} are strictly NAC confluent. Let $X \in \{FT, BT\}$, then the model transformation based on TR_X has functional behavior and does not require backtracking, if all significant critical pairs of TR_X are strictly NAC confluent.*

Proof. The fact is shown for forward translation rules in [17]. By the symmetric character of TGGs we directly derive the result for backward translation rules as well. We now consider the consistency creating rules. All critical pairs are also significant by Def. 17. This allows to apply the local confluence theorem for transformation rules with NACs (Thm. 3.10.8 in [23]) and we can conclude that the system is locally confluent. By combining local confluence and termination from the precondition we derive that the system is confluent. This means that the system has functional behavior. Moreover, confluence implies that the computation of consistency creating markings based on TR_{CC} does not require backtracking. In particular, according to Def. 12, each terminated TGT sequence via TR_{CC} provides a maximal consistency marking, i.e. no terminating sequence is omitted or declared as invalid. □

Definition 18 (Maximal Triple Sequence). *Given an injective morphism $i : G \to H$, then a TGT sequence $s = (\emptyset \overset{tr^*}{\Longrightarrow} G)$ is i-maximal, if for any extended TGT sequence $s' = (\emptyset \overset{tr^*}{\Longrightarrow} G \overset{tr,m}{\Longrightarrow} G')$ with derived transformation inclusion $t : G \to G'$ there is no injective morphism $i' : G' \to H$ compatible with i, i.e. with $i' \circ t = i$.*



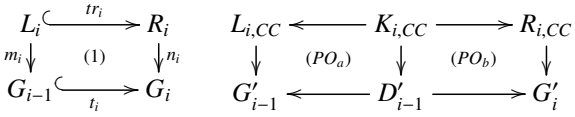
We first proof the equivalence of consistency creating sequences via TR_{CC} and TGT sequences via TR for single steps as stated by Lemma 1.

Lemma 1 (Equivalence of Consistency Creating Step and Triple Step). *Let TR be a set of triple rules with $tr_i \in TR$ and let TR_{CC} be the derived set of consistency creating rules. Given a triple graph G_{i-1}, a triple graph H, and an inclusion $g_{i-1} : G_{i-1} \hookrightarrow H$. Let further $G'_{i-1} = H \oplus Att^{\mathbf{T}}_{G_{i-1}} \oplus Att^{\mathbf{F}}_{H \setminus G_{i-1}}$. Then the following are equivalent:*

1. $\exists$ TGT-*step* $G_{i-1} \overset{tr_i, m_i}{\Longrightarrow} G_i$ *with trace morphism (inclusion)* $t_i : G_{i-1} \hookrightarrow G_i$ *and inclusion* $g_i : G_i \hookrightarrow H$, *such that* $g_i \circ t_i = g_{i-1}(*)$.
2. $\exists$ *consistency creating* TGT-*step* $G'_{i-1} \overset{tr_{i,CC}, m_{i,CC}}{\Longrightarrow} G'_i$

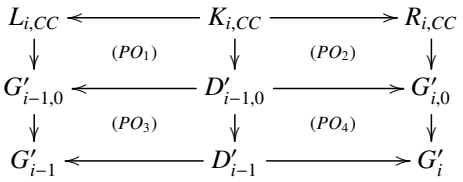
Moreover, the equivalent steps correspond via $G'_i = H \oplus Att^{\mathbf{T}}_{G_i} \oplus Att^{\mathbf{F}}_{H \setminus G_i}$.

Proof. At first we consider the transformation steps without NACs. For simpler notation we assume w.l.o.g. that rule morphisms are inclusions and matches are inclusions except for the data value component.
Constructions:

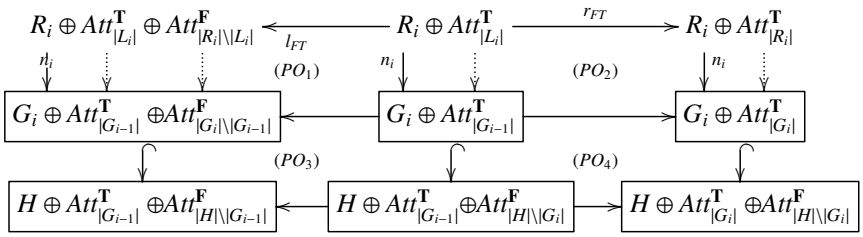


1. TGT-step $G_{i-1} \overset{tr_i, m_i}{\Longrightarrow} G_i$ is given by pushout (1) above.
2. Consistency creating TGT-step $G'_{i-1} \overset{tr_{i,CC}, m_{i,CC}}{\Longrightarrow} G'_i$ is given by (PO_a) and (PO_b) above with
   $L_{i,CC} = R_i \oplus Att^{\mathbf{T}}_{L_i} \oplus Att^{\mathbf{F}}_{R_i \setminus L_i}$
   $K_{i,CC} = R_i \oplus Att^{\mathbf{T}}_{L_i}$
   $R_{i,CC} = R_i \oplus Att^{\mathbf{T}}_{L_i} \oplus Att^{\mathbf{T}}_{R_i \setminus L_i} = R_i \oplus Att^{\mathbf{T}}_{R_i}$ according to Fig. 9.

Direction 1. $\Rightarrow$ 2. : We construct (PO_1), (PO_2), (PO_3), and (PO_4) as follows from diagram (1):



These pushouts consist of the following components concerning translation attributes, where $n_i \colon R_i \to G_i$ is defined by PO (1) and $g_i : G_i \hookrightarrow H$ is given by assumption.

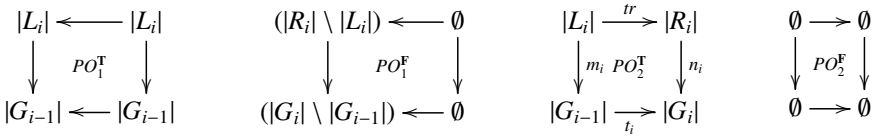


The match $m_{i,CC}$ is constructed as follows:

$$m_{i,CC}(x) = \begin{cases} m_i(x), & x \in L_i \\ \texttt{tr\_}m_i(y), & x = \texttt{tr\_}y, src_{L_{CC}}(x) = y \\ \texttt{tr\_}m_i(y)\texttt{\_}a, & x = \texttt{tr\_}y\texttt{\_}a, src_{L_{CC}}(x) = y \end{cases}$$

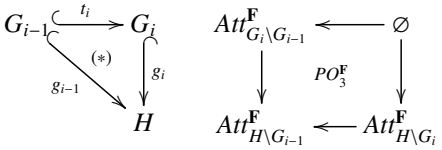
The match m_i is injective except for the data value nodes. For this reason, the match $m_{i,CC}$ is an almost injective match, i.e. possibly non-injective on the data values.

Disregarding translation attributes, diagrams (PO_1) and (PO_2) are trivially pushouts along l_{FT} resp. r_{FT}, which are identities in this restricted view. We now consider the translation attributes.
Thus, we have the following pushouts for the translation attributes:



Pushout $(PO_1^{\mathbf{T}})$ is a trivial pushout. $(PO_1^{\mathbf{F}})$ is a pushout, because (1) is a pullback (pushout along an injective morphism). $(PO_2^{\mathbf{T}})$ is a pushout by (1) and $(PO_2^{\mathbf{F}})$ is a trivial pushout.

By precondition $(*)$ we know that $g_i \circ t_i = g_{i-1}$, where all morphisms are inclusions (see diagram $(*)$ below). This implies that the sets $|G_i| \setminus |G_{i-1}|$ and $|H| \setminus |G_i|$ do not overlap, i.e., $(|G_i| \setminus |G_{i-1}|) \cap (|H| \setminus |G_i|) = \emptyset$. Therefore, diagram $PO_3^{\mathbf{F}}$ is a pushout in sets and thus, diagram (PO_3) is a pushout. Moreover, diagram (PO_4) is trivially a pushout.



Finally, pushouts (PO_a) and (PO_b) are derived by composition: $(PO_a) = (PO_1) + (PO_3)$ and $(PO_b) = (PO_2) + (PO_4)$.

Direction 2. $\Rightarrow$ 1. : Since $G'_{i-1} = H \oplus Att^{\mathbf{T}}_{G_{i-1}} \oplus Att^{\mathbf{F}}_{H \setminus G_{i-1}}$ by general assumption and $m_{CC} : L_{CC} \to G'_{i-1}$ is an almost injective match we can conclude that there is an inclusion $i : m_{CC}(R) \hookrightarrow G_{i-1}$, because $L_{CC} = Att^{\mathbf{T}}(R)$. Moreover, the inclusion $g_{i-1} : G_{i-1} \hookrightarrow H$ is compatible with match $m_{CC} : L_{CC} \to G'_{i-1}$, i.e. $m_{CC} = g_{i-1} \circ i \circ m_0$ with $m_0 : L_{CC} \to m_{CC}(L_{CC})$ given by $m_0(x) = m_{CC}(x)$. This leads to pushouts (PO_1), (PO_2), (PO_3), and (PO_4) as depicted before using the Restriction Thm. (Thm. 6.18 in [7] and Thm. 3.7.5 for the case with NACs in [23]).

We now construct pushout (1) from pushout (PO_2), where we assume w.l.o.g. that $t'_i : D'_{i,0} \to G'_i$ is an inclusion. $(PO_2^{\mathbf{T}})$ and $(PO_2^{\mathbf{F}})$ are pushouts for families of sets and they do not overlap, because the have different types according to the construction of the type graph with attributes by Def. 8. The match is almost injective and thus, it is injective on all components except the data value nodes. Therefore, (1) is a pushout for families of sets, where morphsms n_i, m_i, t_i are derived uniquely from the attribute values. We show that the morphisms are graph morphisms. Since G_{i-1} is given as graph and $t_i : G_{i-1} \to G_i$ is an inclusion, we have that t_i is a graph morphism. Using additional graph morphisms tr_i and n_i together with commutativity of $(PO_2^{\mathbf{T}})$ we derive that m_i is a graph morphism, because t_i is injective. Therefore, (1) is a pushout in **Graphs**. Finally, graph morphism $g_i : G_i \to H$ is given by inclusion $G'_{i,0} \subseteq H$ in (PO_4) and compatibility with t_i and g_{i-1} is ensured by commutativity of (PO_4).

NACs: For each step, we have transformations $G_{i-1} \overset{tr_i, m_i}{\Longrightarrow} G_i$, $G'_{i-1} \overset{tr_{i,CC}, m_{i,CC}}{\Longrightarrow} G'_i$ with $G'_{i-1} = H \oplus \oplus Att^{\mathbf{T}}_{G_{i-1}} Att^{\mathbf{F}}_{H \setminus G_{i-1}}$, $G'_i = H \oplus Att^{\mathbf{F}}_{H \setminus G_i} \oplus Att^{\mathbf{T}}_{G_i}$, and $m_{i,CC}|_{L_i} = m_i$. For a NAC $n : L \to N$ we have to show that, $m_i \models n$ iff $m_{i,CC} \models n_{CC}$.

If $m_{i,CC} \not\models n_{CC}$, there is an injective morphism $q' : N' \to G'_{i-1}$ with $q' \circ n_{CC} = m_{i,CC}$. We can restrict q' to N and G_{i-1} leading to the injective morphism $q : N \to G_{i-1}$. This implies that $q \circ n = m_i$, i.e. $m_i \not\models n$. Vice versa, if $m_i \not\models n$, there is an injective morphism q with $q \circ n = m_i$. Now, let q' be defined with $q'(x) = m_{i,CC}(x)$ for $x \in L_{i,CC}$, $q'(x) = q(x)$

for $x \in N \backslash L_i$, and for each $x \in N \backslash L_i$ we have that $q(x) \in G_{i-1}$. From the definition of G'_{i-1} by general assumption it follows that the corresponding translation attributes $\texttt{tr\_}x$ and $\texttt{tr\_}x\texttt{\_}a$ are set to $\mathbf{T}$ in G'_{i-1}. Thus, q' is well-defined and $q' \circ n_{CC} = m_{i,CC}$, i.e. $m_{i,CC} \not\models n_{CC}$. □

Fact 10 (Equivalence of Triple and Extended Consistency Creating Sequences). *Let $TGG = (TG, \varnothing, TR)$ be a triple graph grammar with derived consistency creating rules TR_{CC} and given $G \in VL(TG)$. Then, the following are equivalent for almost injective matches*

1. *There is a TGT-sequence $s = (\varnothing \overset{tr^*}{\Longrightarrow} G_k)$ via TR with injective embedding $f : G_k \to G$.*
2. *There is a consistency creating sequence $s' = (G'_0 \overset{tr^*_{CC}}{\Longrightarrow} G'_k)$ via TR_{CC} with $G'_0 = Att^{\mathbf{F}}(G)$.*

Moreover, the sequences correspond via $G'_k = H \oplus Att^{\mathbf{T}}_{G_k} \oplus Att^{\mathbf{F}}_{H \backslash G_k}$.

Proof. The equivalence of both sequences follows by applying stepwise Lem. 1. Moreover, stepwise application of Lem. 1 ensures the correspondence of both sequences via $G'_k = H \oplus Att^{\mathbf{T}}_{G_k} \oplus Att^{\mathbf{F}}_{H \backslash G_k}$. □

Fact 11 (Equivalence of Maximal Triple and Complete Extended Consistency Creating Sequences). *Let $TGG = (TG, \varnothing, TR)$ be a triple graph grammar with derived consistency creating rules TR_{CC} and given $G \in VL(TG)$. Then, the following are equivalent for almost injective matches*

1. *There is a TGT-sequence $s = (\varnothing \overset{tr^*}{\Longrightarrow} G_k)$ via TR with injective embedding $f : G_k \to G$, such that s is f-maximal.*
2. *There is a terminated consistency consistency creating sequence $s' = (G'_0 \overset{tr^*_{CC}}{\Longrightarrow} G'_k)$ via TR_{CC} with $G'_0 = Att^{\mathbf{F}}(G)$.*

Moreover, the sequences correspond via $G'_k = H \oplus Att^{\mathbf{T}}_{G_k} \oplus Att^{\mathbf{F}}_{H \backslash G_k}$.

Proof. **Direction** $1 \Rightarrow 2$: We derive consistency creating sequence s' by Fact 10. Assume that s' is not terminated, i.e. there is a further TGT step $G'_k \overset{tr_{CC}}{\Longrightarrow} G'_{k+1}$. This implies by Lem. 1 that there is an extension of sequence s with step $G_k \overset{tr}{\Longrightarrow} G_{k+1}$ with inclusion $g_{k+1} : G_{k+1} \to G$ compatible with $f = g_k$. This is a contradiction to the precondition that s is f-maximal. Therefore, the assumption is incorrect and s' is complete.
Direction $2 \Rightarrow 1$: We derive consistency creating sequence s by Fact 10. Assume that s is not f-maximal, i.e. there is a further TGT step $G_k \overset{tr}{\Longrightarrow} G_{k+1}$ with inclusion $g_{k+1} : G_{k+1} \to G$ compatible with $f = g_k$. This implies by Lem. 1 that there is an extension of sequence s' with step $G'_k \overset{tr_{CC}}{\Longrightarrow} G'_{k+1}$. This is a contradiction to the precondition that s' is complete. Therefore, the assumption is incorrect and s is f-maximal.

The correspondence of both sequences via $G'_k = H \oplus Att^{\mathbf{T}}_{G_k} \oplus Att^{\mathbf{F}}_{H \backslash G_k}$ follows by Fact 10. □

A.4 Constructions and Results for Auxiliary operations

Remark 7 (Detailed Construction of Auxiliary operation Del*).*

Let $r = (s,t) \in R = VL(TG)$ be a correspondence relation, then the result of the auxiliary operation Del for computing the maximal consistent subgraph and deleting the remainder is given by $\mathsf{Del}(r) = (a, r', b)$ according to Fig. 6 via the following construction. From r, we obtain a triple graph $G = (G^S \xleftarrow{s} G^C \xrightarrow{t} G^T)$. If $G \in VL(TGG)$, then we can directly define $G_k = G$. In the general case, we are able to construct a maximal TGT-sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_k$ with $G_k \in VL(TGG)$ and inclusion $f : G_k \to G$, where maximality means that there is no extension $\varnothing \overset{tr^*}{\Longrightarrow} G_k \overset{tr}{\Longrightarrow} G_{k+1}$ with $G_k \subseteq G_{k+1} \subseteq G$. Then, we have $\mathsf{Del}(r) = (a, r', b)$, where $a = (f^S, 1)$, $r' = (s_k, t_k)$ with consistent $r' \in C$ and $b = (f^T, 1)$.

For this purpose, we use the construction of consistency creating sequences according to Def. 12. Intuitively, we start with the triple graph G and extend it with translation attributes set to "**F**" and we call this initial graph G'_0. Thereafter, we apply the consistency creating rules as long as possible. The resulting marked graph G'_k then consists of G and additional translation attributes. We derive G_k by removing all translation attributes and additionally all elements from G that are labelled with "**F**" in G'_k.

In more detail, at first the initial triple graph is given by $G'_0 = Att^{\mathbf{F}}(G)$. Thereafter, the consistency creating rules $tr_{CC} \in TR_{CC}$ are applied as long as possible starting at G'_0 leading to a terminated consistency creating sequence $G'_0 \overset{tr^*_{CC}}{\Longrightarrow} G'_k$. The triple graph G_k is obtained by removing all elements in G'_k that are labelled with false and by removing all remaining translation attributes. More formally, by Fact 11 we derive the triple sequence $\varnothing \overset{tr^*}{\Longrightarrow} G_k$ with with injective embedding $f : G_k \to G$, such that s is f-maximal (see Def. 18). Therefore, $G_k \in VL(TGG)$ and an f-maximal consistent subgraph of G.

By Def. 3 the sets of operational rules are required to be deterministic. This implies by Def. 16 that no backtracking is necessary. Therefore, the triple graph G_k is unique. If additionally $G \in VL(TGG)$ then there is a triple sequence $\varnothing \overset{tr^*}{\Longrightarrow} G$ and by Fact 11 there is a corresponding consistency creating sequence $G'_0 \overset{tr^*_M}{\Longrightarrow} G'_k$ implying that $G_k = G$. This implies that operation Del is a total function (see Fact 12).

Fact 12 (Functional Behavior of TGG Operation Del**).** *Let TGG be triple graph grammar with deterministic sets of operational rules, then the execution of operation* Del *based on the derived consistency creating rules TR_{CC} has functional behavior, i.e. it terminates and leads to unique results.*

Proof. According to Rem. 7, the f-maximal TGT sequence $s = (\varnothing \overset{tr^*}{\Longrightarrow} G_k)$ with embedding $f : G_k \hookrightarrow G$ is constructed using the derived set TR_{CC} of consistency creating rules. By Fact 11, the existence of the f-maximal sequence s is equivalent to the existence of the corresponding complete consistency creating sequence via TR_{CC}. By Def. 16, the set TR_M has functional behavior. Therefore, the construction of sequence s terminates and yields a unique result. □

Remark 8 (Detailed Construction of Auxiliary operation fAdd*).*

Let $r = (s,t) \in C = VL(TGG)$ be a consistent correspondence relation, $a = (1, a_2) : G^S \to G'^S$ be a source modification and $G'^S \in VL_S$. The result of the auxiliary operation fAdd for propagating the additions of source modification a is given by $\mathsf{fAdd}(r,a) = (r', b)$ according to Fig. 6 via the following construction. Using $r = (s,t)\colon G^S \leftrightarrow G^T$ with $r \in C$ we obtain a consistent triple graph $G = (G^S \xleftarrow{s} G^C \xrightarrow{t} G^T)$. By completeness of model transformations via forward rules (Thm. 1 in [8]) there is a corresponding source consistent forward sequence $(G^S \leftarrow \varnothing \to \varnothing) \overset{tr^*_F}{\Longrightarrow} G$. Due to completeness of forward transformations via forward translation rules (Thm. 1 in [17]) there is a corresponding forward translation sequence $s_{1,FT}$. From $a = (1, a_2)\colon G^S \to G'^S$ we obtain $a_2 : G^S \to G'^S$, such that $G_0 = (G'^S \xleftarrow{a_2 \circ s} G^C \xrightarrow{t} G^T)$. Since backtracking is not necessary, there is a continuing forward translation sequence $s_{2,FT}$, which can be composed with $s_{1,FT}$ to a complete forward translation sequence $s_{3,FT}$. Due to equivalence of forward transformations via forward translation rules and forward rules (Fact 1 in [17]) there is a corresponding source consistent forward sequence for $s_{3,FT}$ with $G' = (G'^S \xleftarrow{s'} G'^C \xrightarrow{t'} G'^T)$, $b_2\colon G^T \to G'^T$, and $G' \in VL(TGG)$ due to correctness of model transformations via forward rules (Thm. 1 in [8]). Hence, we obtain a correspondence relation $G'^S \xrightarrow{r'=(s',t')} G'^T$ and $b = (1, b_2)\colon G^T \to G'^T$. $G' \in VL(TGG)$ implies that $r' = (s', t')\colon G'^S \leftrightarrow G'^T$ is consistent. Moreover, operation fAdd is a total function (see Fact 12).

Lemma 2 (Construction via Operation fAdd). *Let $Synch(TGG) = (MF(TGG), \mathsf{fPpg}, \mathsf{bPpg})$ be a derived TGG synchronization framework, where the sets of operational rules of TGG are deterministic. Let $r = (G^S \leftarrow G^C \to G^T) \in C = VL(TGG)$ be a consistent correspondence relation, $a = (G^S \leftarrow G^S \to G'^S)$ be a source model update. Then, the execution of operation fAdd yields a forward sequence $s_{F,2} = \langle G_0 \overset{tr^*_{F,2}}{\Longrightarrow} G_n\rangle$ with $G_0 = (G'^S \leftarrow G^C \to G^T)$, $G_n = G' = (G'^S \leftarrow G'^C \to G'^T)$, and $G' \in VL(TGG)$. Moreover, there is a forward sequence $s_{F,1} = (G'^S \leftarrow \varnothing \to \varnothing) \overset{tr^*_{F,1}}{\Longrightarrow} G_0$, such that the composed forward sequence $s_F = (s_{F,1}; s_{F,2})$ is source consistent.*

A similar result holds for operation bAdd, concerning an induced target consistent backward sequence.

Proof. At first we consider the operation fAdd and assume that no conservative policy is applied. Since $G \in VL(TGG)$ there is source consistent forward sequence $s_{F,0} = \langle G_{0,0} = (G^S \leftarrow \varnothing \to \varnothing) \overset{tr^*_{F,0}}{\Longrightarrow} (G^S \leftarrow G^C \to G^T) = G_{0,k}\rangle$ according to Thm. 2 in [13] (correctness). This sequence corresponds to a complete forward translation sequence $s_{FT,0} = G'_{0,0} \overset{tr^*_{FT,0}}{\Longrightarrow} G'_{0,k}$ according to Fact. 1 in [18] with $G'_{0,0} = (Att^{\mathbf{F}}(G^S) \leftarrow \varnothing \to \varnothing)$ and $G'_{0,k} = (Att^{\mathbf{T}}(G^S) \leftarrow G^C \to G^T)$.

Using Fact 6 (extension of FT sequences) and inclusion $g : G \to G'$ we derive the forward translation sequence $s_{FT,1} = \langle G'_{1,0} \overset{tr^*_{FT,1}}{\Longrightarrow} G'_{1,k}$ with $G'_{1,0} = (Att^{\mathbf{F}}(G'^S) \leftarrow \varnothing \to \varnothing)$ and $G'_{1,k} = (H^S \leftarrow G^C \to G^T)$, where $H^S = G'^S \oplus Att^{\mathbf{T}}_{G^S} \oplus Att^{\mathbf{F}}_{G'^S \setminus G^S}$. Now, we can extend the sequence by applying further forward translation rules of TR^{+s}_{FT} as long as possible leading to $s_{FT} = (s_{FT,1}; s_{FT,2}) = \langle G'_{1,0} \overset{tr^*_{FT,1}}{\Longrightarrow} G'_{1,k} \overset{tr^*_{FT,2}}{\Longrightarrow} G'_{2,n}$, where

termination is ensured by functional behavior of the sets of operational rules using the precondition that sets of the operational rules are deterministic. Moreover, the deterministic sets of operational rules ensure by Def. 16 that the pair $(TR^{1s}_{FT}, TR^{+s}_{FT})$ is sequentially independent acc. to Def. 14. By Fact 4 we can shift the steps via TR^{1s}_{FT} to the end leading to a sequence $s'_{FT} = (s_{FT,3}; s_{FT,4})$ with sequences $s_{FT,3} = G'_{1,0} \overset{tr^*_{FT,3}}{\Longrightarrow} H'_l$ via TR^{+s}_{FT} and $s_{FT,4} = H'_l \overset{tr^*_{FT,4}}{\Longrightarrow} G'_{2,n}$ via TR^{1s}_{FT}. Since the pair $(TR^{1s}_{FT}, TR^{+s}_{FT})$ is sequentially independent acc. to Def. 14 we can conclude that no further rule in TR^{+s}_{FT} is applicable to H'_l, because none was applicable to $G'_{2,n}$ in the sequence before. By Fact 5 we can conclude that $s_{FT,3}$ is a model transformation sequence via FT rules, i.e., $H'_l = (H'^S_l \leftarrow H'^C_l \rightarrow H'^T_l)$ with $H'^S_l = Att^{\mathbf{T}}(G'^S)$. Therefore, also $s'_{FT} = (s_{FT,3}; s_{FT,4})$ is a complete forward translation sequence, i.e., $G'_{2,n} = (Att^{\mathbf{T}}(G'^S \leftarrow G'^C_{2,n} \rightarrow G'^T_{2,n})$. Thus, we derive $G' = (G'^S \leftarrow G'^C_{2,n} \rightarrow G'^T_{2,n})$ by removing the translation attributes from $G'_{2,n}$. This allows us to apply Thm. 1 in [17] (correctness of forward translation) and we derive that $G' \in VL(TGG)$.

Moreover, sequences s_{FT} and s'_{FT} are shift equivalent implying that also s_{FT} is a complete forward translation sequence. By Fact 1 in [18] (equivalence to source consistent forward sequences) we conclude that there is a corresponding source consistent forward sequence s_F of s_{FT} with $s_F = (s_{F,1}; s_{F,2})$, where $s_{F,1}$ is the forward sequence corresponding stepwise to forward translation sequence $s_{FT,1}$ and $s_{F,2}$ is the forward sequence corresponding stepwise to forward translation sequence $s_{FT,2}$.

Now, since the modification of the rules via a conservative policy only restricts the model transformation relation MTR_{FT} and preserves the left totality property of MTR_{FT} according to Fact 7, we can conclude that the execution of operation fAdd ensures that the above sequences exist.

The result for operation bAdd follows by the symmetric definition of TGGs and of the dual nature of operational rules TR^{+t}_{BT} with respect to TR^{+s}_{FT}. □

Fact 13 (Functional Behavior of TGG Operation fAdd). *Let TGG be a triple graph grammar with deterministic sets of operational rules, then the execution of operations* fAdd *and* bAdd *based on the derived operational rules TR^{+s}_{FT} and TR^{+t}_{BT} has functional behavior, i.e. it terminates and leads to unique results.*

Proof. According to Lemma 2 we derive that the execution of operation fAdd terminates with a unique resulting correspondence $r' = G' \in C = VL(TGG)$. The resulting update b is given by $b = (G^T \leftarrow G^T \rightarrow G'^T)$ derived from the trace morphism of the forward sequence $G_0 \overset{tr^*_F}{\Longrightarrow} G_n$ computed by operation fAdd. Therefore, fAdd has functional behavior.

The result for operation bAdd follows by the symmetric definition of TGGs. □

A.5 Correctness of TGG Synchronization Operations

This section presents the proof of the main result of this paper: Thm. 1 concerning correctness and invertibility of model synchronizations based on TGGs. At the beginning, we discuss the notion of completeness for model synchronization inspired by the notion

of completeness for model transformations based on TGGs. Thereafter, we prove Fact 1 of Sec. 6 concerning the deterministic character of the synchronization operations. The subsequent Lemma 3 shows the correctness and completeness results and is used in the last part of this section to show our main result Thm. 1 concerning also invertibility.

Remark 9 (Completeness of Synchronization Operations). The notion of completeness according to Def. 19 requires the applicability of the synchronization operations to an arbitrary correspondence together with a given consistent result of an update in one domain. This means that the operations are total with respect to the possible inputs according to Def. 2. This notion corresponds to completeness for model transformations [8,17] based on the derived model transformation relations, where completeness additionally requires totality with respect to the possible output models $G^T \in VL_T$. The additional requirement is not considered in the case of model synchronization, because we require fPpg and bPpg to be functions and do not allow general relations. This would be a possible further generalization of our framework.

Definition 19 (Completeness of Synchronization Operations). *Let $TGG = (TG, \emptyset, TR)$ be a TGG with deterministic sets of derived operational rules and with derived TGG-synchronization framework $Synch(TGG) = (MF, \mathsf{fPpg}, \mathsf{bPpg})$. The synchronization operations* fPpg *and* bPpg *are called* complete*, if the following condition hold. The execution of forward propagation operation* fPpg *can be performed for each consistent source model $G^S \in VL_S$. Vice versa, the execution of backward propagation operation* bPpg *can be performed for each consistent target model $G^T \in VL_T$.*

Fact 1 (Deterministic Synchronization Operations) *Let TGG be a triple graph grammar and let matches be restricted to almost injective morphisms. If the critical pairs of the sets of operational rules are strictly confluent and the systems of rules are terminating, then the sets of operational rules are deterministic, which implies that the derived synchronization operations* fPpg *and* bPpg *are deterministic as well.*

Proof. First of all, the sets of operational rules are deterministic according to Fact 9. Operations fAln and bAln are given by pullback construction, which is unique up to isomorphism by definition. Therefore, they are deterministic. By Facts 12 and 13 for Del and fAdd we can conclude that fPpg has functional behavior. Operation bPpg has functional behavior by the symmetry of the definitions. □

Lemma 3 (TGG Synchronization Framework). *Let $Synch(TGG)$ be a derived TGG synchronization framework, such that the sets of operational rules of TGG are deterministic. Then $Synch(TGG)$ is correct and complete.*

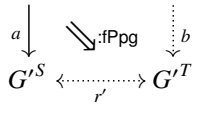
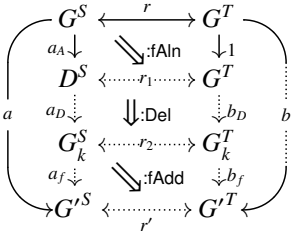
Proof. Correctness: By Fact 1 the provided constructions of operations fPpg and bPpg based on forward translation rules and backward translation rules, respectively, have functional behavior, i.e., for each input the computation results in a unique output.

$(a1)$: $\forall\, c \in C$:
$$\begin{array}{ccc} G^S & \xleftarrow{c} & G^T \\ {\scriptstyle 1}\downarrow & \searrow^{:\mathsf{fPpg}} & \downarrow{\scriptstyle 1} \\ G^S & \xleftarrow[c]{} & G^T \end{array}$$

$(a2)$: $\forall\, G'^S \in VL_S$:
$$\begin{array}{ccc} G^S & \xleftarrow{r} & G^T \\ {\scriptstyle a}\downarrow & \searrow^{:\mathsf{fPpg}} & \downarrow{\scriptstyle b} \\ G'^S & \xleftarrow[r':C]{} & G'^T \end{array}$$

$(b1)$: $\forall\, c \in C$:
$$\begin{array}{ccc} G^S & \xleftarrow{c} & G^T \\ {\scriptstyle 1}\downarrow & \swarrow_{:\mathsf{bPpg}} & \downarrow{\scriptstyle 1} \\ G^S & \xleftarrow[c]{} & G^T \end{array}$$

$(b2)$: $\forall\, G'^T \in VL_T$:
$$\begin{array}{ccc} G^S & \xleftarrow{r} & G^T \\ {\scriptstyle a}\downarrow & \swarrow_{:\mathsf{bPpg}} & \downarrow{\scriptstyle b} \\ G'^S & \xleftarrow[r':C]{} & G'^T \end{array}$$

Signature	Definition of Components
$\forall\, G'^S \in VL_S$: 	 $a = (a_1, a_2)$ $= (G^S \xleftarrow{a_1} D^S \xrightarrow{a_2} G'^S)$ $a_A = (a_1, 1)$ $a_D = (a'_1, 1)$ $a_f = (a_1 \circ a'_1, a_2)$ $b = b_f \circ b_D$

Law ($a1$): According to Def. 4, operation fPpg is composed by the following three steps. At first, fAln performs a pullback along the given identity modification leading to the resulting corresponding relation $r' = r$. Since $r \in C = VL(TGG)$, there is a triple sequence $s_D = \langle \varnothing \overset{tr^*}{\Longrightarrow} G \rangle$ with $r = G$. Therefore, operation Del results in $(a, r', b) = (1, r, 1)$ using that the sets of operational rules are deterministic according to Def. 4 implying in particular functional behavior. The triple sequence s_D induces a corresponding source consistent forward sequence $s_A = \langle (G^S \leftarrow \varnothing \rightarrow \varnothing) \overset{tr^*}{\Longrightarrow} G \rangle$ via the composition result (Thm. 1 in [13]). Due to the functional behavior of the set of forward translation rules this ensures that fAdd yields the same triple graph, i.e., the resulting correspondence is given by $r' = r$ and the resulting update is given by $b = id$. with $G \in VL(TGG)$.

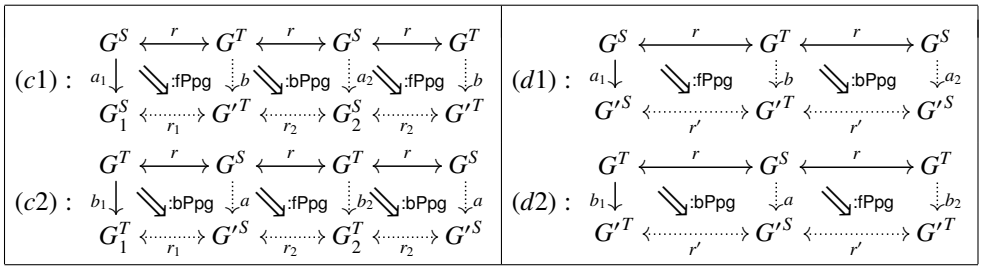
Law ($a2$): By Lemma 2, the computed sequences via operation Del and fAdd induce the corresponding composed source consistent forward sequence $s_A = (s_{A,1}; s_{A,2}) = \langle (G'^S \leftarrow \varnothing \rightarrow \varnothing) \overset{tr^*_F}{\Longrightarrow} G' \rangle$. Source consistency of s_A ensures that $r' = G' \in C = VL(TGG)$ due to correctness of model transformation sequences (Thm. 2 in [13]).

The laws ($b1$) and ($b2$) follow from laws ($a1$) and ($a2$) by the symmetric character of TGGs as well as by the definitions for forward and backward translation rules.

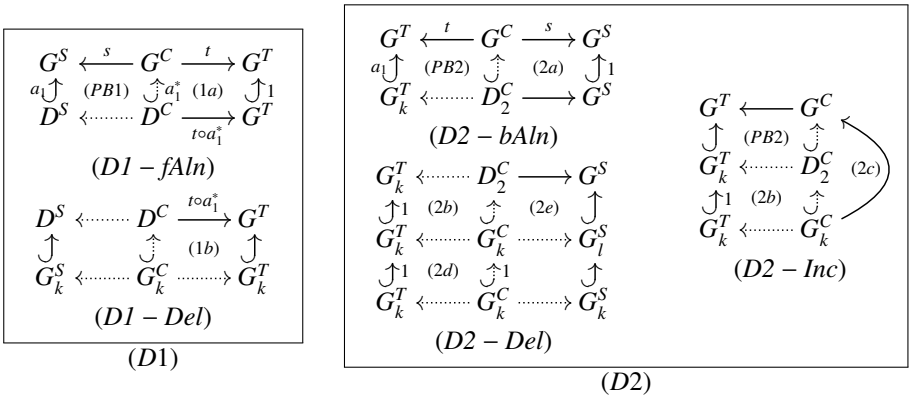
Completeness: By Def. 19, we have to show that the synchronization operations can be executed for all consistent source resp. target models that are the result of a given update. According to Fact 1, the propagation operations are total functions. By Def. 3, the operation fPpg is defined for all input models and provides a consistent result for consistent source models $G'^S \in VL_S$, i.e. no error occurs. Therefore, fPpg can be successfully applied for given source models $G'^S \in VL_S$. Analogously, operation bPpg ensures completeness for target models $G'^T \in VL_T$ due to the symmetry of the definitions.

Theorem 1 (Correctness and Weak Invertability (See Thm. 1 in Sec. 6)). *Let Synch(TGG) be a derived TGG synchronization framework, such that the sets of operational rules of TGG are deterministic. Then Synch(TGG) is correct and complete. If, additionally, TGG is pure and at most one set of operational rules was extended by a conservative policy, then Synch(TGG) is weakly invertible and if, moreover, TGG is tight and no policy was applied, then Synch(TGG) is also invertible.*

Proof. Correctness and Completeness are shown by Lemma 3.

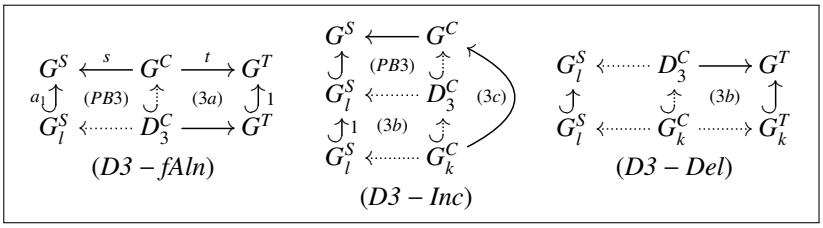


Weak Invertibility: Law (c1): At first, we show that the derived maximal markings (correspondences after applying fAln and Del) of the last two diagrams are the same.



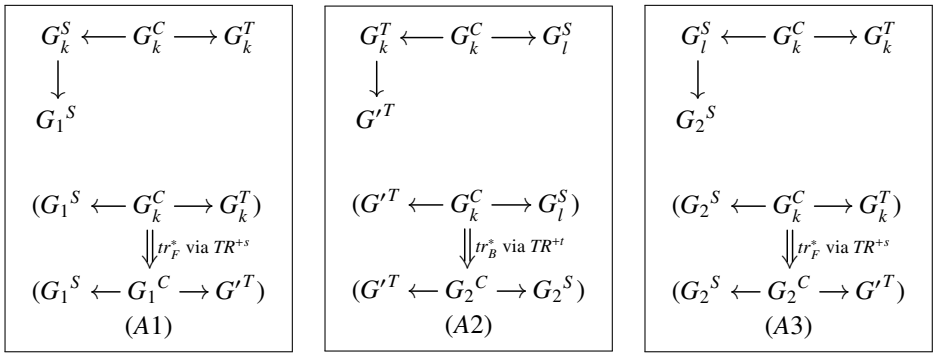
Diagrams $(D1)$ and $(D2)$ above concern the first two tiles of law $(c1)$ (fPpg and bPpg, respectively). Moreover, the diagrams consider the auxiliary operations fAln, bAln, and Del only. In diagram $(D1)$, the maximal consistent triple graph G_k is computed via applying first fAln and then Del. In particular, no further source identic consistency creating rule is applicable $(*)$ - a property that we will use for the last diagram.

Since diagram $(1a + 1b)$ commutes and $(PB2)$ is a pullback we derive a compatible inclusion $G_k^C \rightarrow D_2^C$ in diagram $(D2 - Inc)$, i.e., $(2b)$ and $(2c)$ commute. By equivalence of triple and consistency creating sequences according to Fact 10, we derive the corresponding consistency creating sequence. Since G_k^T in $(D2 - Del)$ is already completely created by the triple sequence respectively marked by the consistency creating sequence, there is no further consistency creating rule applicable apart from target identic ones. The TGG is pure by precondition. This implies that the only available target identic rules are empty rules on the target and correspondence component. By precondition, no backtracking is necessary for the sets of operational rules. Therefore, the completion of the consistency creating sequence by applying consistency creating rules as long as possible yields a triple graph $G_l = (G_k^T \leftarrow G_k^C \rightarrow G_l^S) \in VL(TGG)$ and the extended consistency creating sequence is terminated.



(D3)

Diagram (D3) concerns the last tile of law (c1). Similarly to Diagrams (D1) – (D2), we derive that there is an inclusion $G_k^C \to D_3^C$ in (D3 − Inc) with commutative (3b) and (3c), because (PB3) is a pullback and (2a + 2e) commutes. Again, by Fact 10 we derive the corresponding consistency creating sequence for triple graph $G_l \in VL(TGG)$. Since G_l^T in (D3 − Del) is already completely created by the triple sequence respectively marked by the consistency creating sequence, there is no further consistency creating rule applicable apart from source identic ones. The TGG is pure by precondition. This implies that the only available source identic rules are empty rules on the source and correspondence component. If one of them is still applicable, then it would have been applicable already for G_k^T in digram (D1), because it is empty on the source and correspondence component. This would be a contradiction to the terminated consistency creating sequence in diagram (D1) (see property (∗) for (D1)). Therefore, we derive again the triple graph $G_l = (G_k^T \leftarrow G_k^C \to G_l^S)$ as result of the terminated consistency creating sequence.



As visualized in diagram (A1), operation fAdd yields the sequence $s_{F,1} = \langle G_A = (G_1{}^S \leftarrow G_k^C \to G_k^T) \overset{tr_F^*}{\Longrightarrow} (G_1{}^S \leftarrow G_1{}^C \to G'^T)\rangle$ via TR_F^{+s} with $G' \in VL(TGG)$. By Lem. 2 there is a corresponding forward sequence $s_{fA} = \langle (G_1{}^S \leftarrow \emptyset \to \emptyset) \overset{tr^*_{F,fA}}{\Longrightarrow} (G_1{}^S \leftarrow G_k^C \to G_k^T)$, such that $s_A; s_{F,1}$ is source consistent.

By the composition and decomposition result for TGGs (Thm. 1 in [13]) we derive the corresponding backward sequences $s_{bA} = \langle (\emptyset \leftarrow \emptyset \to G'^T) \overset{tr^*_{B,bA}}{\Longrightarrow} (G_k^S \leftarrow G_k^C \to G'^T)\rangle$ via TR_B and $s_{B,1} = \langle (G_k^S \leftarrow G_k^C \to G'^T) \overset{tr_B^*}{\Longrightarrow} (G_1{}^S \leftarrow G_1{}^C \to G'^T)\rangle$ via TR_B^{+s}, such that $s_{bA}; s_{B,1}$ is target consistent.

Since the sets of operational rules of the TGG are deterministic (Def. 16), the pair $(TR_{BT}^{1t}, TR_{BT}^{+t})$ is sequentially independent and by Fact 2 we can shift the target identic steps to the end and since the TGG is pure we have $TR^{1t} = TR_S^{1t}$ leading to sequences

$s_1 = \langle (G_k^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,1}}{\Longrightarrow} ({G_m}^S \leftarrow {G_1}^C \rightarrow {G'}^T) \rangle$ via TR_B^{+t}, and $s_2 = \langle (G_m^S \leftarrow {G_1}^C \rightarrow {G'}^T) \overset{tr^*_{B,2}}{\Longrightarrow} ({G_1}^S \leftarrow {G_1}^C \rightarrow {G'}^T) = G' \rangle$ via TR_B^{1t}. Now, the original forward sequence used rules in TR_{FT}^{+s}, which implies that the rules in sequence s_1 are source and target creating, i.e. we have sequence $s_1 = \langle (G_k^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,1}}{\Longrightarrow} ({G_m}^S \leftarrow {G_1}^C \rightarrow {G'}^T) \rangle$ via $TR_B^{+t} \cap TR_B^{+s}$ $(**)$.

As visualized in diagram $(A2)$, operation bAdd yields the sequence $s_{A,2} = \langle G_B = (G_l^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,3}}{\Longrightarrow} ({G_2}^S \leftarrow {G_2}^C \rightarrow {G'}^T) \rangle$ via TR_B^{+t}. By Lem. 2 there is a corresponding backward sequence $s_{bA} = \langle (\varnothing \leftarrow \varnothing \rightarrow {G'}^T) \overset{tr^*_{B,bA}}{\Longrightarrow} (G_l^S \leftarrow G_k^C \rightarrow {G'}^T)$, such that $(s_{bA}; s_{A,2})$ is target consistent.

Moreover, operation Del yields the two triple sequences $s_{D,1} = \langle \varnothing \overset{tr^*}{\Longrightarrow} G_k \rangle$ via TR and $s_{D,2} = \langle G_k \overset{tr^*}{\Longrightarrow} (G_l^S \leftarrow G_k^C \rightarrow G_k^T) \rangle$ via $TR^{1t} \cap TR^{+s}$ as shown for Diagram $(D2)$. Applying stepwise the composition and decomposition result (Thm. 1 in [13]) we derive that sequence $s_{D,2}$ corresponds to the last part of sequence s_{bA}. Let $s_{bA,2} = \langle (G_k^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,bA}}{\Longrightarrow} (G_l^S \leftarrow G_k^C \rightarrow {G'}^T)$ be this last part and $s_{bA,1}$ the remaining first part. Since the TGG is pure by precondition we have that $TR^{1t} = TR_S^{1t}$ and using that $s_{D,2}$ is a sequence via $TR^{1t} \cap TR^{+s}$ we derive that $s_{bA,2}$ is a sequence via TR_S^{+s}.

We can compose sequences $s_{bA,2}$ and $s_{A,2}$ to $s_5 = (s_{bA,2}; s_{A,2})$. Recall that $(s_{bA}; s_{A,2}) = (s_{bA,1}; s_{bA,2}; s_{A,2})$ is target consistent. Since the TGG is deterministic, we have by Def. 16 that the pair $(TR_{BT}^{1t}, TR_{BT}^{+t})$ is sequentially independent, such that we can shift the steps of $s_{bA,2}$ beyond $s_{A,2}$ according to Fact 3 and derive the sequences $s_6 = \langle (G_k^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,1}}{\Longrightarrow} (G_\#{}^S \leftarrow {G_2}^C \rightarrow {G'}^T) \rangle$ and $s_7 = \langle (G_\#^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,2}}{\Longrightarrow} ({G_2}^S \leftarrow {G_2}^C \rightarrow {G'}^T) \rangle$. Using $(**)$ from before concerning diagram $(A1)$, we know that there is also the backward sequence $s_{1,B} = (G_k^S \leftarrow G_k^C \rightarrow {G'}^T) \overset{tr^*_{B,1}}{\Longrightarrow} ({G_m}^S \leftarrow {G_1}^C \rightarrow {G'}^T)$ via $TR_B^{+t} \cap TR_B^{+s}$. Since the TGG is deterministic, we know that backtracking is not required and we have functional behavior. Functional behavior ensures unique results for model transformation sequences, i.e., we have unique results for target consistent backward transformation sequences. Sequence $(s_{0,B}; s_{1,B})$ is target consistent according to property $(**)$. Analogously, we can compose $s_{0,B}$ and s_6 and derive derive the target consistent backward sequence $(s_{0,B}; s_6)$, because s_7 uses only target identic rules. If no policy is applied we can already conclude that $({G_m}^S \leftarrow {G_1}^C \rightarrow {G'}^T)$ of sequence $s_{1,B}$ coincides with $(G_\#{}^S \leftarrow {G_2}^C \rightarrow {G'}^T)$ of s_6.

By precondition, at most one conservative policy was applied. We have to show that the derived backward transformation sequences via the composition and decomposition result are not eliminated by the policy.

case 1: A conservative policy was applied for the set of forward translation rules. We can conclude that $s_{1,B}$ and s_6 coincide on the resulting triple graph, because there is no restriction on the matches for the backward steps. Thus, the corresponding forward sequence $s_{1,F}$ of $s_{1,B}$ (from Diagram $(A1)$) is a possible transformation sequence for Diagram (3), but additionally, some target identic steps can be applied according to s_7. Now, since the operational forward rules are identic on the source component this

means that the resulting triple graph of Diagram ($A3$) is given by ($G_2^S \leftarrow G_2^C \rightarrow G'^T$) as required.

case 2: A conservative policy was applied for the set of backward translation rules. We can conclude that s_6 might not be the directly corresponding sequence to $s_{1,B}$, but according to the definition of a conservative policy we know that if there is a match for an original operational rule then there is also one for the rule that is extended by attribute conditions. Therefore, the sequences coincide on the applied rules, but the matches may be different. Thus, we have that s_6 is performed via rules that are source and target creating, because it corresponds to $s_{1,B}$ concerning the applied rules. This means that we can derive from (s_6; s_7) the corresponding forward sequence via source creating rules as shown in diagram ($A3$). Since, in this case no policy is applied for the set of forward translation rules, this sequence is possible. Due to functional behavior, the resulting triple graph is ensured to be the one of (s_6; s_7) in Diagram ($A2$).
Law ($c2$): The result for law ($c2$) follows by the symmetry of the definitions.
Invertibility: Concerning law ($d1$) we can use the precondition that all operational rules are source and target creating and no policy was applied. This ensures for the diagrams above that graph $G_l^S = G_k^S$ in Diagram ($D2$) and the derived sequences $s_{1,B}$ and s_3 directly correspond by the composition and decomposition result for TGGs. The result for law ($d2$) follows by the symmetry of the definitions. □

A.6 Incremental computation of consistency creating sequences via operation Del

In order to reduce the effort for computing consistent parts of a triple graph, we can store an already executed consistency creating sequence and reuse this sequence for the next computation. For this purpose, all involved rules and matches have to be stored. By construction the corresponding subobject transformation system (STS) as presented in [1,16], we do not need to also store all intermediate graphs, but only one so-called super object that is obtained by gluing together the intermediate graphs along their common parts. All matches are then embeddings into the super object.

Moreover, the provided dependency analysis in [16] can be used to efficiently separate the invalid steps and their dependent ones from the remaining valid steps. Now, a stored consistency creating sequence can be reused by removing the invalid steps (caused by the deletion of elements in the right hand side of the rule or by violation of application conditions by the new graph) and their dependent steps. Thereafter, the obtained consistency creating sequence is continued, such that further still consistent substructures can be reused. The derived new maximal marking via consistency creating is then used for executing the operation fAdd.

B Details of the Case Study

In this section, we provide the details of our case study. At first we present the complete TGG in App. B.1. Thereafter, in App. B.3, we show the analysis results concerning functional behaviour of the TGG operations. Finally, in App. B.4, we show that the derived synchronization framework is not invertible in the general sense by a counter example. However, as shown in Sec. 6, the synchronization framework is weakly invertible.

B.1 Components of the TGG

The triple graph grammar $TGG = (TR, \emptyset, TR)$ of our case study is given by the triple type graph TG in Fig 10, the empty start graph and the triple rules in the subsequent figures.

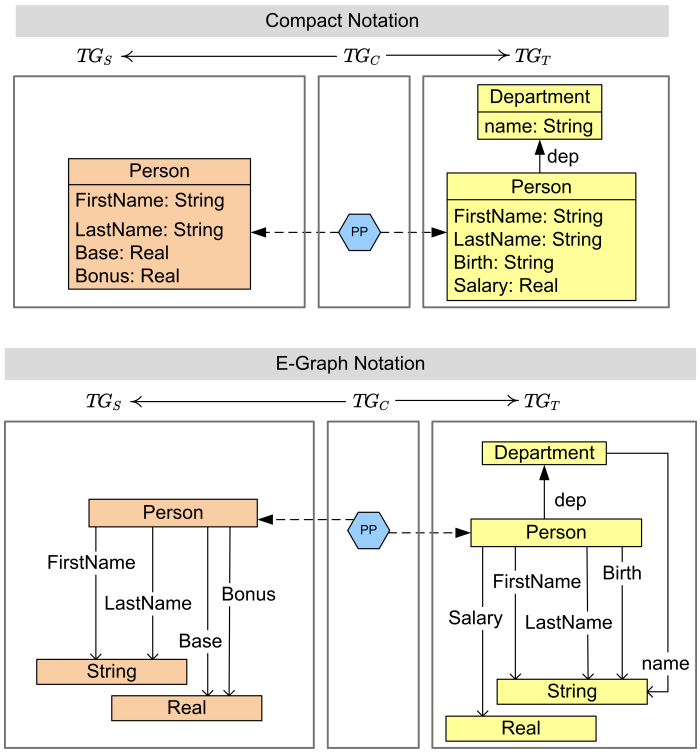


Fig. 10. Triple type graph TG

Example 6 (Type Graph). According to the triple type graph TG in Fig. 10, the models of the source domain contain persons including their detailed salary information (bonus

and base salary) and their names. Models of the target domain additionally contain the departments to which a person is assigned to, the birth date of a person, and a single value for the complete salary of a person, while the details about bonus and base salary are not provided.

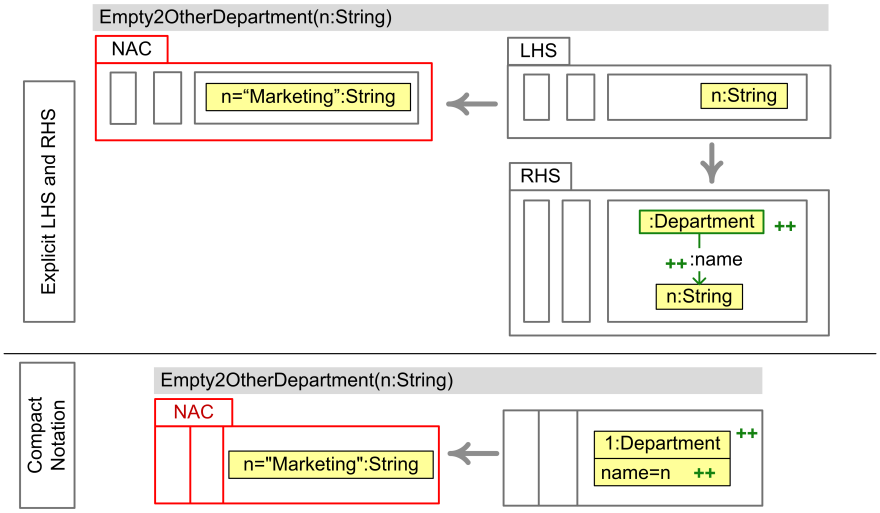


Fig. 11. Triple rules - part 1

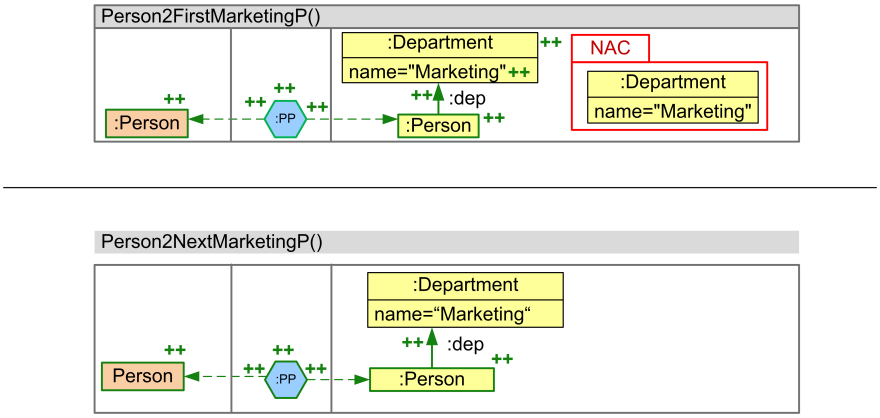


Fig. 12. Triple rules - part 2

Example 7 (Triple Rules (Parts 1 and 2)). The triple rules of the TGG are depicted in short notation in Figs. 11 to 14 and in we first explain Figs. 11 and 12. The first rule (Empty2OtherDepartment) is depicted additionally with explicit left and right hand side. It creates a new department in the target model, but does not change the source model. The negative application condition (NAC) ensures that this rule cannot be applied for creating a department with name "Marketing". For this purpose, rule 2 (Person2FirstMarketing) is used, where the NAC ensures that the given target model does not contain already a department with name "Marketing". This rule additionally creates a person of the new department in the target model and a corresponding person in the source model. Rule "Person2NextMarketingP" is applied in order to extend both models with further persons in the marketing department. Note that the attributes of the created persons are not set. This is possible in our formal framework of attributed graph transformation based on the notion of E-graphs [7]. The main advantage is that we can propagate changes of attribute values without the need for deleting and recreating the owning structural nodes. This is important from the efficiency and application point of view.

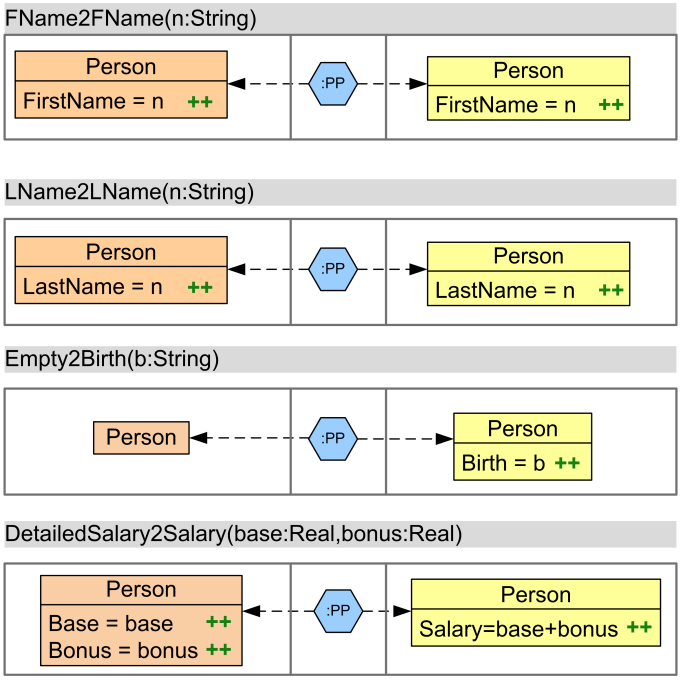


Fig. 13. Triple rules - part 3

Example 8 (Triple Rules (Parts 3 and 4)). The further triple rules of the TGG are depicted in Figs. 13 to 14. The four rules in Fig. 13 concern the creation of attribute values

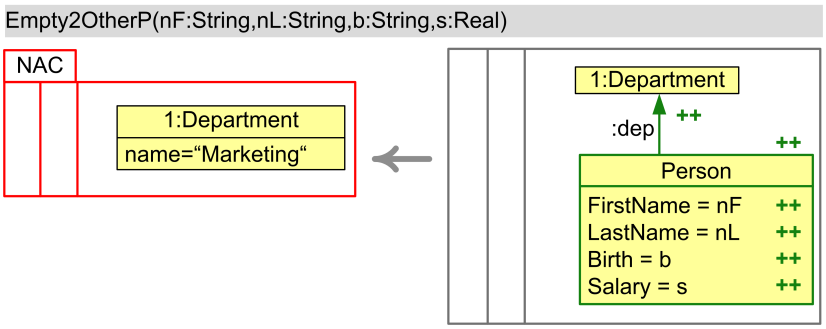


Fig. 14. Triple rules - part 4

only. Rules "FName2FName" and "LName2LName" create new corresponding values for first and last names, respectively. The next rule "Empty2Birth" assigns the birth date of a person in the target component and does not change the source component. Finally, rule "DetailedSalary2Salary" assigns the detailed salary values (bonus and base) in the source component and the sum of them in the target component. The last rule "Empty2OtherPerson" of the TGG is presented in Fig. 14 and creates a new person of a department that is not in the marketing department. Therefore, there are no correspondence to the source model and the rule directly creates the person including all attribute values.

B.2 Derived and Modified Sets of Operational Rules

Based on the specified TGG defining the language of consistent integrated models $VL(TGG)$ we automatically derive the operational rules for consistency creating, forward translation and backward translation according to Def. 9. Moreover, we define a conservative policy for one backward translation rule (Def. 15) in order to ensure functional behavior. In the following, we present the resulting sets of operational rules including the conservative policy.

Example 9 (Derived Sets of Consistency Creating Rules). Figures 15-16 show the set of the consistency creating rules derived from the TGG in Sec. B.1 according to Def. 9. Intuitively, for each element $x \in R$ (node, edge, or attribute) of a triple rule $tr = (L \rightarrow R)$ a separate translation attribute (`tr` or `tr_`x) is added for the consistency creating rule tr_{CC}. If an element $x \in R$ is preserved by the triple rule tr ($x \in R \backslash L$), then the consistency creating rule preserves it as well and the translation attribute has value **T**. Otherwise, if $x \in R$ is is created by tr ($x \in L$), then it becomes a preserved element in the consistency creating rule tr_{CC} and the corresponding translation attribute is changed from **F** to **T**. In visual notation, this means that all plus signs are replaced by additional translation attributes whose values are changed from **F** to **T**.

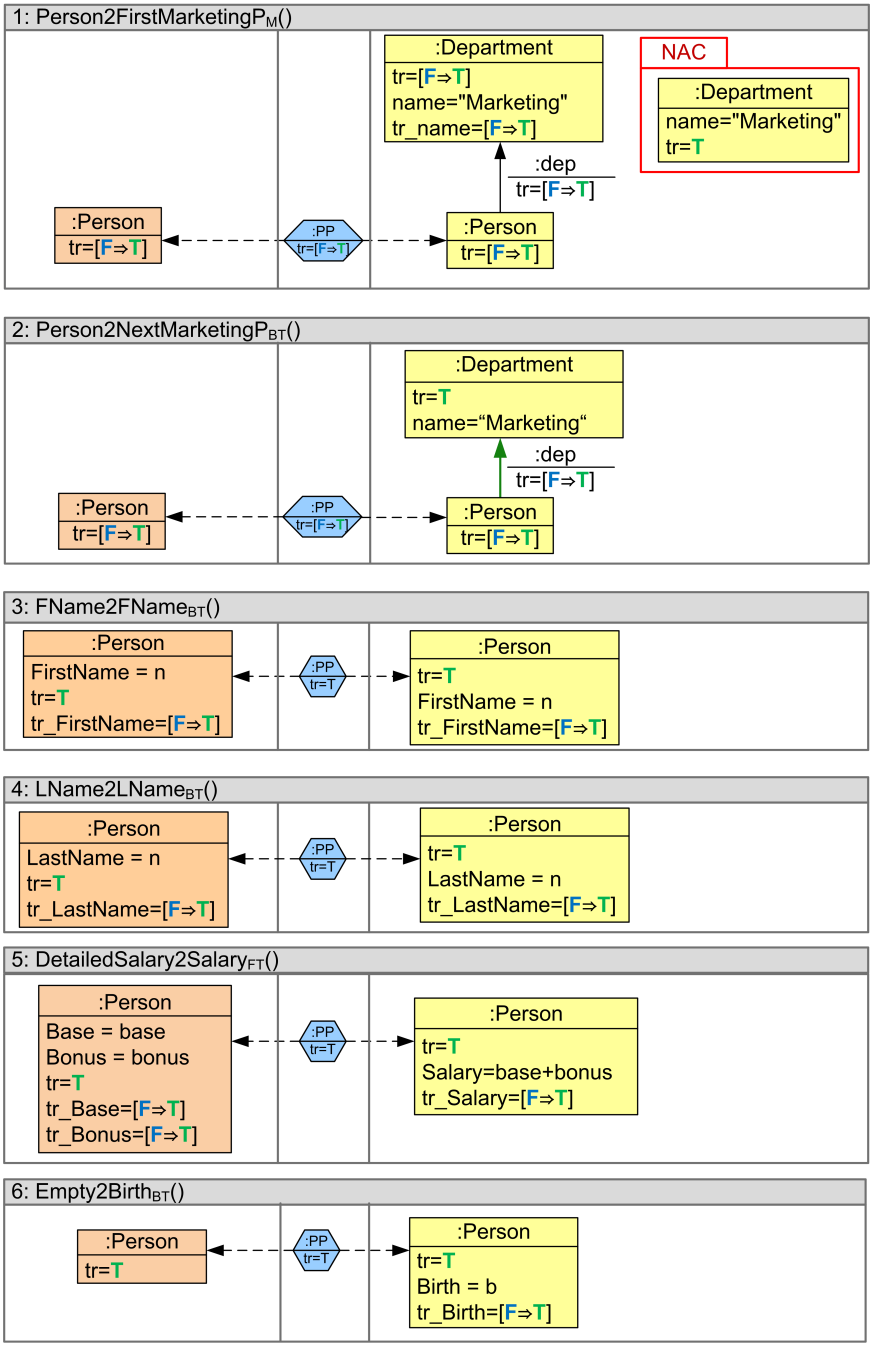


Fig. 15. Derived Operational Triple rules: TR_{CC} (part 1)

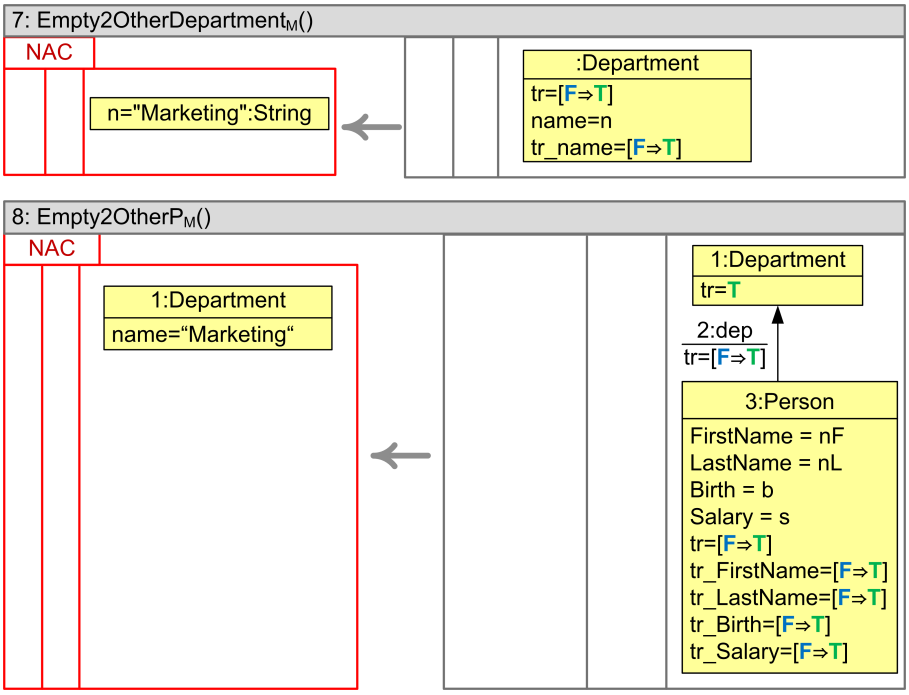


Fig. 16. Derived Operational Triple rules: TR_{CC} (part 2)

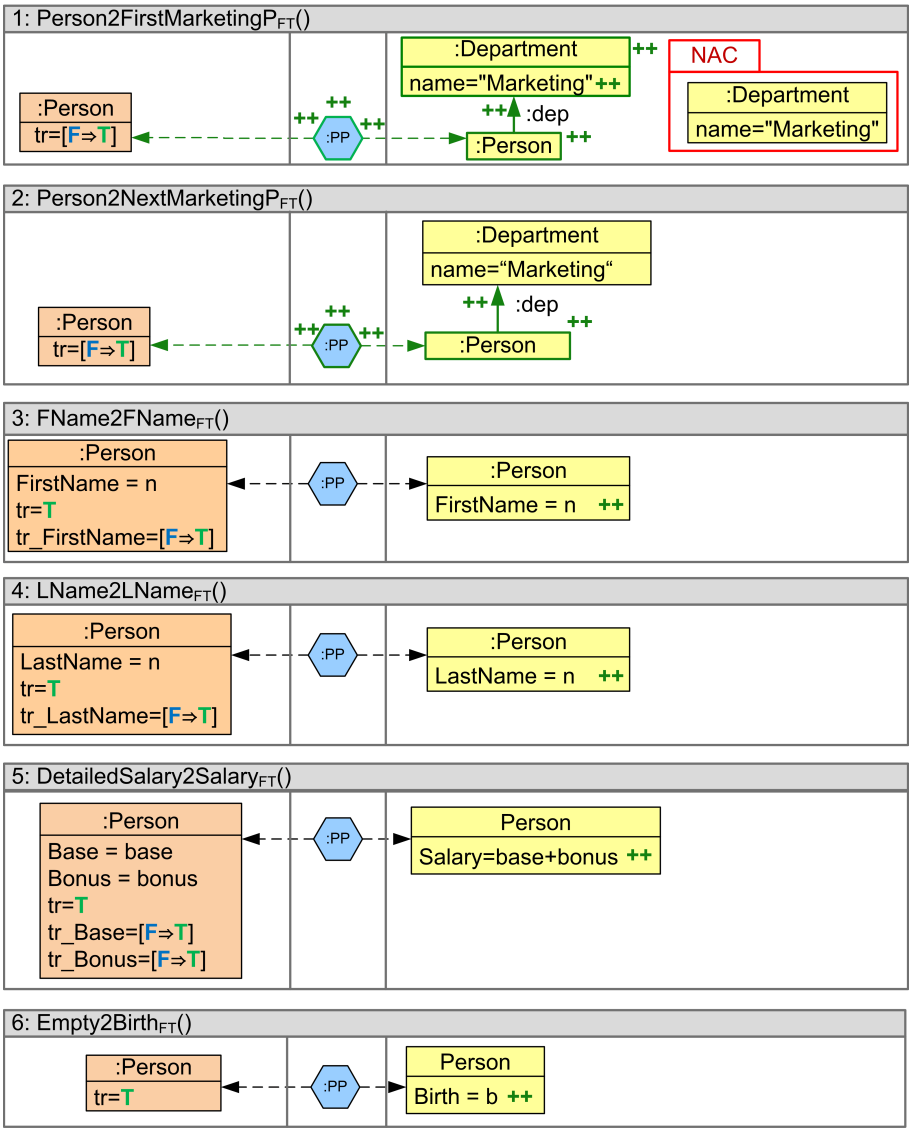


Fig. 17. Derived Operational Triple rules: TR_{FT} (part 1)

Example 10 (Derived Sets of Forward Translation Rules). Figures 17-18 show the set of the forward translation rules derived from the TGG in Sec. B.1 according to Def. 9. Intuitively, for each element x in the source component R^S (node, edge, or attribute) of a triple rule $tr = (L \rightarrow R)$ a separate translation attribute (`tr` or `tr_`x) is added for the forward translation rule tr_{FT}. If an element $x \in R^S$ is preserved by the triple rule tr, then the forward translation rule preserves it as well and the translation attribute has

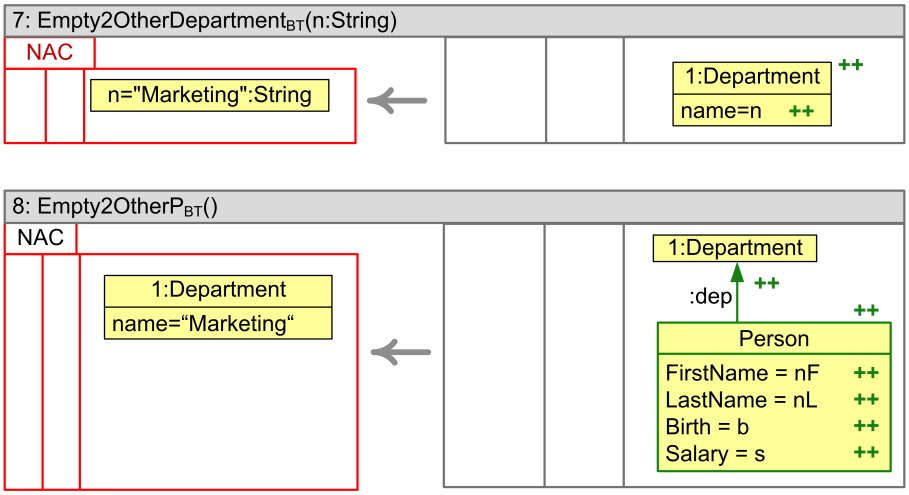


Fig. 18. Derived Operational Triple rules: TR_{FT} (part 2)

value **T**. Otherwise, if $x \in R^S$ is is created by tr, then it becomes a preserved element in the forward translation rule tr_{FT} and the corresponding translation attribute is changed from **F** to **T**. In visual notation, this means that all plus signs in the source component are replaced by additional translation attributes whose values are changed from **F** to **T**.

Note that the rules 6-8 are identities on the source component and not used for fPpg in order to ensure termination. This is possible as shown in Sec. B.3 according to Fact 4 using the automated analysis via the tool AGG for dependency analysis.

Example 11 (Derived Sets of Backward Translation Rules). Figures 19-20 show the set of the backward translation rules derived from the TGG in Sec. B.1 according to Def. 9 and using an additional conservative policy. They are derived dually to the case of forward translation rules. Intuitively, for each element x in the target component R^T (node, edge, or attribute) of a triple rule $tr = (L \to R)$ a separate translation attribute (`tr` or `tr_`x) is added for the backward translation rule tr_{BT}. If an element $x \in R^T$ is preserved by the triple rule tr, then the backward translation rule preserves it as well and the translation attribute has value **T**. Otherwise, if $x \in R^T$ is is created by tr, then it becomes a preserved element in the backward translation rule tr_{BT} and the corresponding translation attribute is changed from **F** to **T**. In visual notation, this means that all plus signs in the target component are replaced by additional translation attributes whose values are changed from **F** to **T**.

Rule 5 : "$DetailedSalary2Salary_{BT}$()" is extended by a policy in the form of an additional positive application condition. Since the left hand side of this rule specifies only the sum of the salary of a person, the values of the base and bonus components are not fixed via a match. The application condition PAC requires that both values are set to half times the amount of the salary sum. Now, this is possible for each number, such that we can conclude that the policy is conservative (Def. 15), which is important

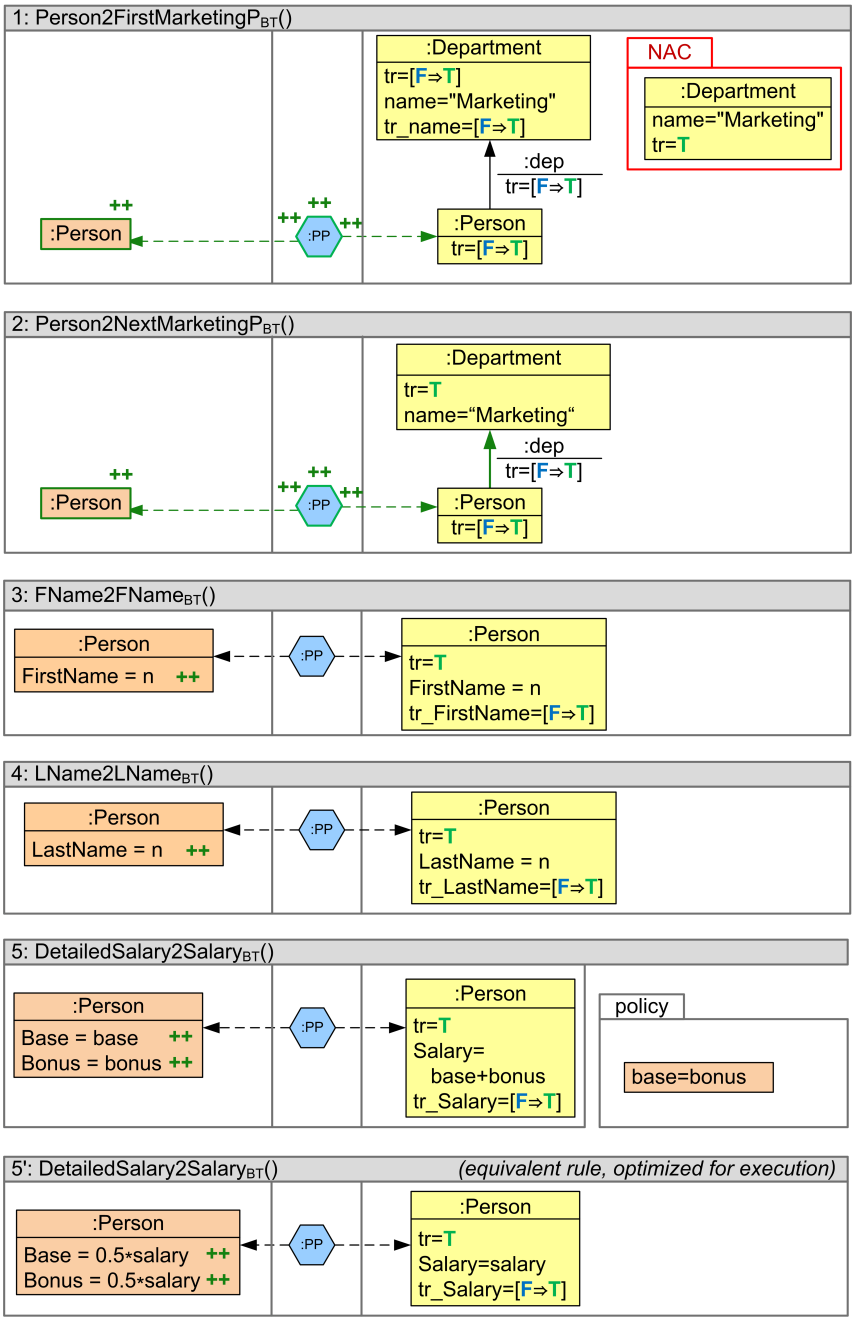


Fig. 19. Derived Operational Triple rules: TR_{BT} (part 1)

for ensuring completeness of the propagation operation bPpg (see Thm. 1). Note that all backward translation rules are used for bPpg in contrast to operation fPpg before.

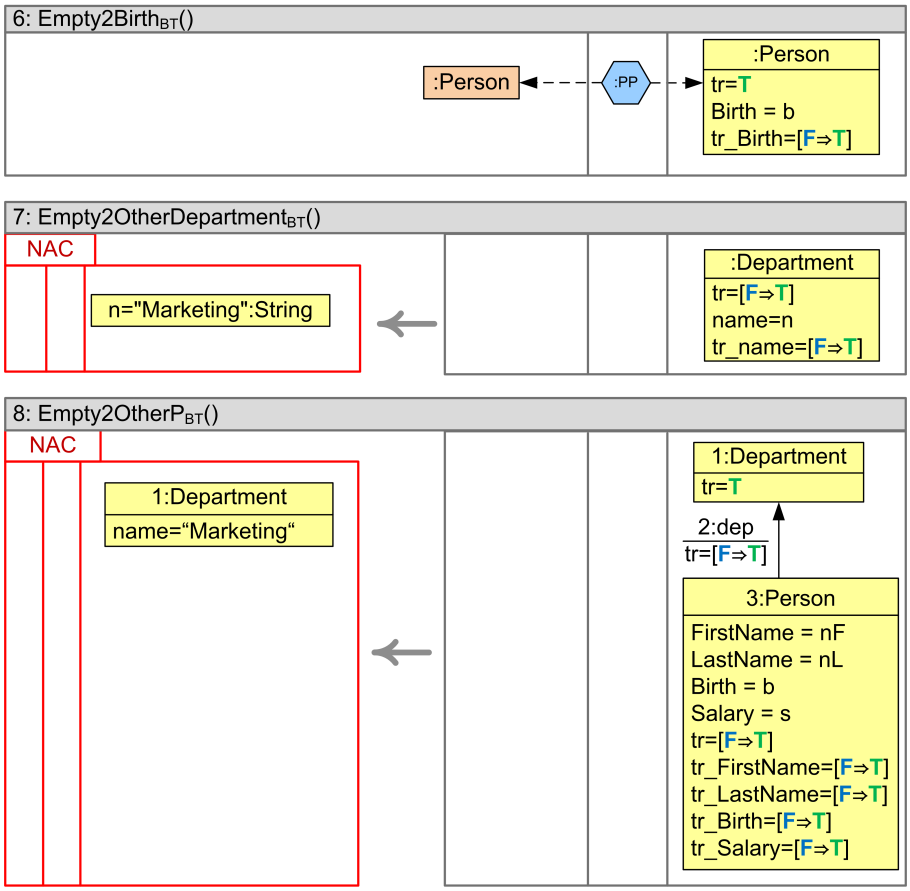


Fig. 20. Derived Operational Triple rules: TR_{BT} (part 2)

B.3 Analysis of the Sets of Operational Rules and Functional Behaviour

If all significant critical pairs of TR_{CC} are strictly confluent, then we do not need to perform backtracking for consistency creating and there is a unique maximal consistency marking for any input. This means especially that the result is unique, if there are no critical pairs for TR_{CC}. Termination is ensured, if each rule created at least one element, which is practically always given, because otherwise there would be a superfluous triple rule $tr : L \to R$ in TR with $L = R$.

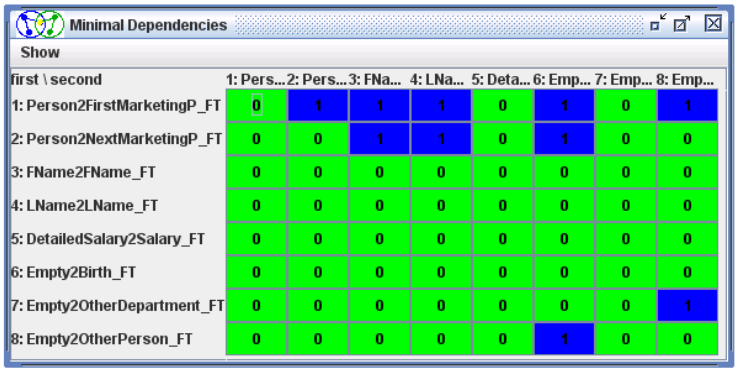


Fig. 21. Dependency analysis with AGG for TR_{FT} - blue fields contain dependencies

Fact 14 (Case Study: Deterministic TGG Synchronization Operations - Termination). *We verified that the synchronization operations of our case study terminate. For this purpose, we used the critical pair analysis engine of AGG for analyzing all dependencies.*

Concerning the set TR_{CC}, we have that each rule is marking changing, which ensures termination.

Concerning the set TR_{FT}, we derived the resulting table depicted in Fig. 21. The source identic rules are the rules with number 6 to 8. According to the table there is no dependency (blue entry) for any pair (p, q) with $p \geq 6$ and $q \leq 5$. Therefore, termination is ensured, because the source identic rules are not used for the forward translation.

Finally, there are no target identic backward translation rules, because all triple rules are creating on the target component. Therefore, termination is ensured for the set of backward translation rules.

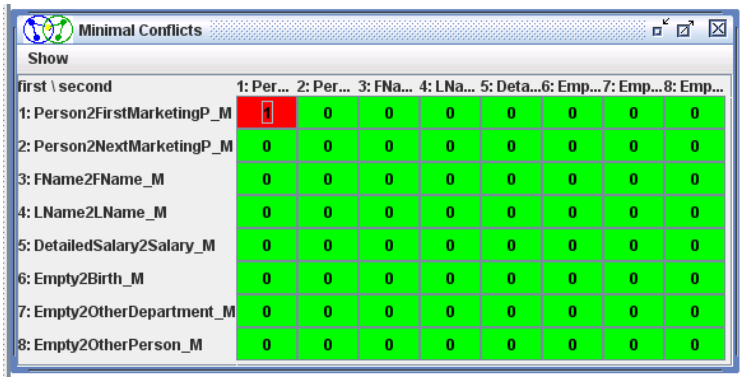


Fig. 22. Critical pair analysis with AGG for TR_{CC} - red fields contain conflicts

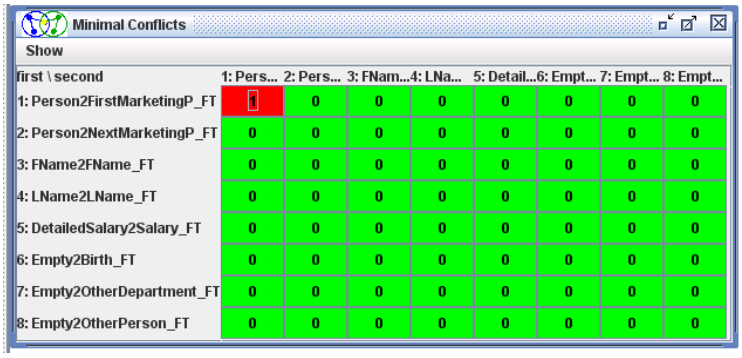


Fig. 23. Critical pair analysis with AGG for TR_{FT} - red fields contain conflicts

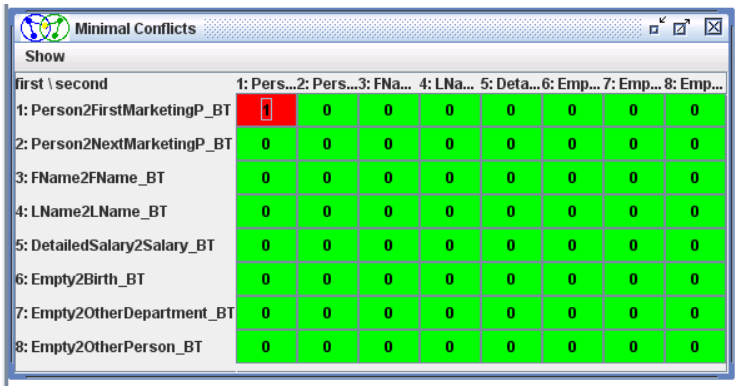


Fig. 24. Critical pair analysis with AGG for TR_{BT} - red fields contain conflicts

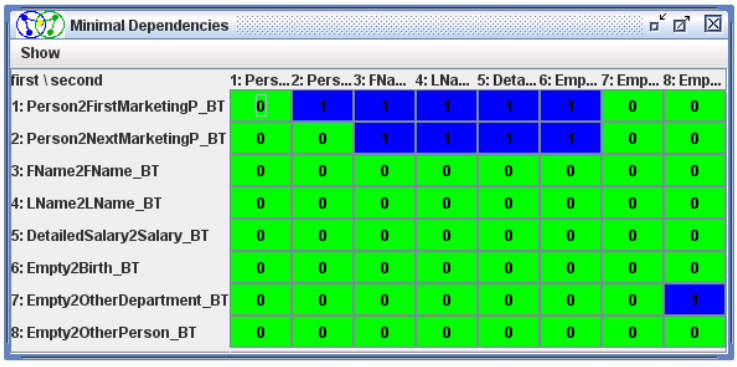


Fig. 25. Dependency analysis with AGG for TR_{BT} - blue fields contain dependencies

Fact 15 (Case Study: Deterministic TGG Synchronization Operations - Unique Results). *For our case study we verified that the synchronization operations are functions using the result of termination from before. For this purpose we used the critical pair analysis engine of AGG.*

Concerning the set TR_{CC}, we derived the resulting table depicted in Fig. 22. The only generated critical pair is (p_1, p_1) for $p_1 = Person2FirstMarketingP_{CC}$ and it is strictly confluent by applying rule $p_2 = Person2NextMarketingP_{CC}$ to the remaining structure and since p_2 does not contain any NAC we automatically have NAC-strict confluence.

Concerning the set TR_{FT}, we derived the resulting table depicted in Fig. 23 using that the language VL(TGG) ensures that there are no two departments with name "Marketing" (ensured by the NACs of the first two rules). the only significant critical pair is again strictly confluent via one transformation step using rule $p_2 = Person2NextMarketingP_{FT}$, where no NAC is involved.

The set TR_{BT} is not functional directly, because there is the choice how to split the salary into base and bonus. We can restrict the choice for the rule "DetailedSalary2Salary" to $base = bonus = 1/2 salary$ as a policy pol as shown by the additional positive application condition in Fig. 19. The policy is conservative, because no other rule depends on this rule as shown by the generated dependency table by AGG in Fig. 25 and moreover, if there is a match for the original rule, then there is a match for the restricted rule because the restricted values are real numbers and therefore always possible. We derive the table depicted in Fig. 24, where the only significant critical pair is again strictly confluent via one transformation step using rule $p_2 = Person2NextMarketingP_{BT}$, where no NAC is involved.

B.4 Invertibility

As presented in Sec. 6, the case study ensures weak invertibility. However, it does not ensure invertibility in the more general sense and we now provide an explicit counter example.

Fact 16 (Case Study: Weak Invertibility). *In order to apply Thm. 1 concerning weak invertibility we have to show that the TGG is pure and a at most one set of operational rules was restricted by a conservative policy. The used policy for the set of backward translation rules is conservative, which we have shown already in Fact. 15. No further policy is applied and the TGG is pure, because each rule is either creating on the source and target component, or it is creating either on the source or the target component and empty on the other components. Therefore, we can apply Thm. 1 and derive weak invertibility.*

Fact 17 (Case Study: Invertibility). *The derived synchronization framework for our case study is not invertible in the general notion (laws (d_1) and (d_2)) according to the following counter example, where we consider law (d_2). Consider a model update b as depicted in Fig. 26, where a new person is added leading to the resulting target model G'^T. The propagation via* bPpg *yields the source model G'^S, but the subsequence*

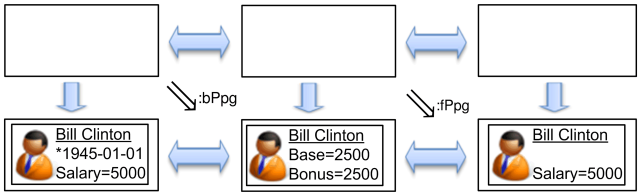


Fig. 26. Counter example for invertibility

propagation via operation fPpg *yields the target model* G''^{T}*, which does not contain any information about the birth date. Therefore,* $G'^{T} \neq G''^{T}$ *meaning that Synch*(*TGG*) *is not invertible. However, Synch*(*TGG*) *is weak invertible according to Fact 16 above. This means that the process is invertible if* G'^{T} *is generated by* fPpg*, which means that* G'^{T} *has no birth date for persons that are added by the update.*