Correctness of Model Synchronization Based on Triple Graph Grammars

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 [22,19,1]). Triple graph grammars (TGGs) have been successfully applied in several case studies for bidirectional model transformation, model integration and synchronization [17,21,9,8], and in the implementation of QVT [12]. Inspired by Schürr et al. [20,21], we started to develop a formal theory of TGGs [7,14], 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. [9,8], and the replica synchronization framework [3]. The main ideas and results are the following:

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 [3] and realized by model transformations based on TGGs [7]. 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 [23].

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 is given in a technical report [15].

2 Running Example

Throughout the paper, we use a simple running example, which is based on previous work [2]. 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 to the marketing department (depicted as persons without pencils) and the target model

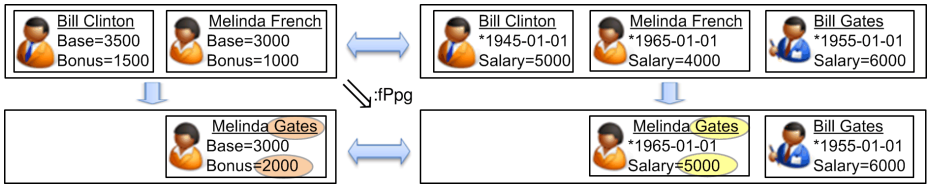


Fig. 1. Forward propagation

additionally contains the person "Bill Gates" belonging to the technical department (depicted as a person with pencil). The first 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 [21,7].

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 [7,6]. 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 preserve the typing. For $TG = (TG^S \leftarrow TG^C \to TG^T)$, 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.

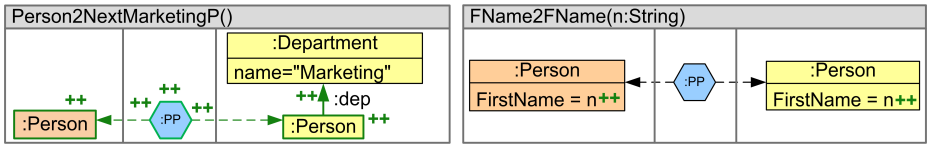


Fig. 2. Some triple rules of the TGG

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 $m : L \to G$. The result of this application is the triple graph H, where L is replaced by R in G. Technically, the result of the transformation is defined by a pushout diagram, 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 [14]. 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 $VL(TGG) \subseteq VL(TG)$.

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

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 [3] and triple spaces [4]. In our context, a model update $\delta : G \to G'$ is specified as a *graph modification* $\delta : G \overset{i_1}{\hookleftarrow} I \overset{i_2}{\hookrightarrow} G'$. 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.

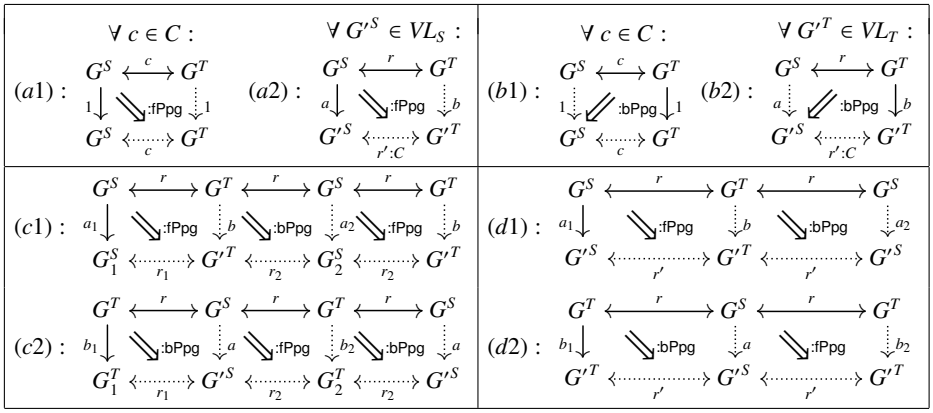


Fig. 4. Laws for correct and (weak) invertible synchronization frameworks

Definition 1 (TGG Model Framework). *Let* $TGG = (TG, \varnothing, TR)$ *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.*

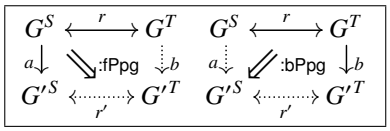


Fig. 3. Synchronization operations

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. 3, 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 [5,17]. 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. 4. 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. 3, 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. 3. 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. 4 are satisfied.*

Given 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. 4 are satisfied; and it is* invertible *if additionally axioms* $(d1)$ *and* $(d2)$ *in Fig. 4 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 [7].

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

triple rule tr

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

source rule tr_S

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

forward rule tr_F

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

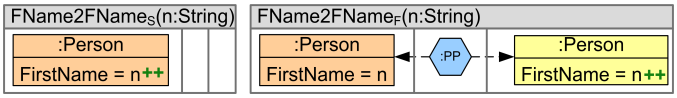


Fig. 5. Derived source and forward rules

The derived operational rules provide the basis for the definition of model transformations based on source-consistent forward transformation sequences [7,11]. *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 [7,14]. 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 [7,11,14]. *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* [14] 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 sucessfully executed if the source model is completely marked with **T**. We indicate these markers in the examples by checkmarks in the visual notation and by bold font face in the graph representation. Similarly, from the triple rules, we can also create *marking rules* [15], 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.

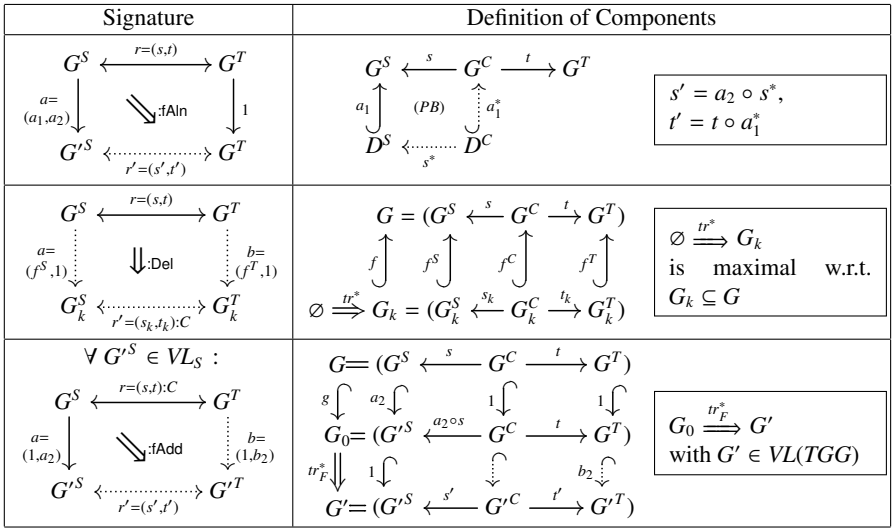


Fig. 6. Auxiliary operations fAln, Del and fAdd

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 marking 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 rules [15], 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 [15].

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 $G_k^S \subseteq G^S$ and $G_k^T \subseteq G^T$ 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_k^T are deleted, i.e., the new target model is given by G_k^T. Finally, in the last step (operation fAdd), the elements in G'^S that extend G_k^S are transformed to corresponding structures in G'^T, i.e., G_k^T 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 $TGG = (TG, \varnothing, TR)$ 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 marking 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 marking rules corresponds to a maximal triple rule sequence as shown in the right middle part of Fig. 6 and discussed in more detail in [15]. 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 [7], the sequence implies consistency of G' as stated above. The constructions for these auxiliary operations are provided in full detail in [15].

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. 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*).*

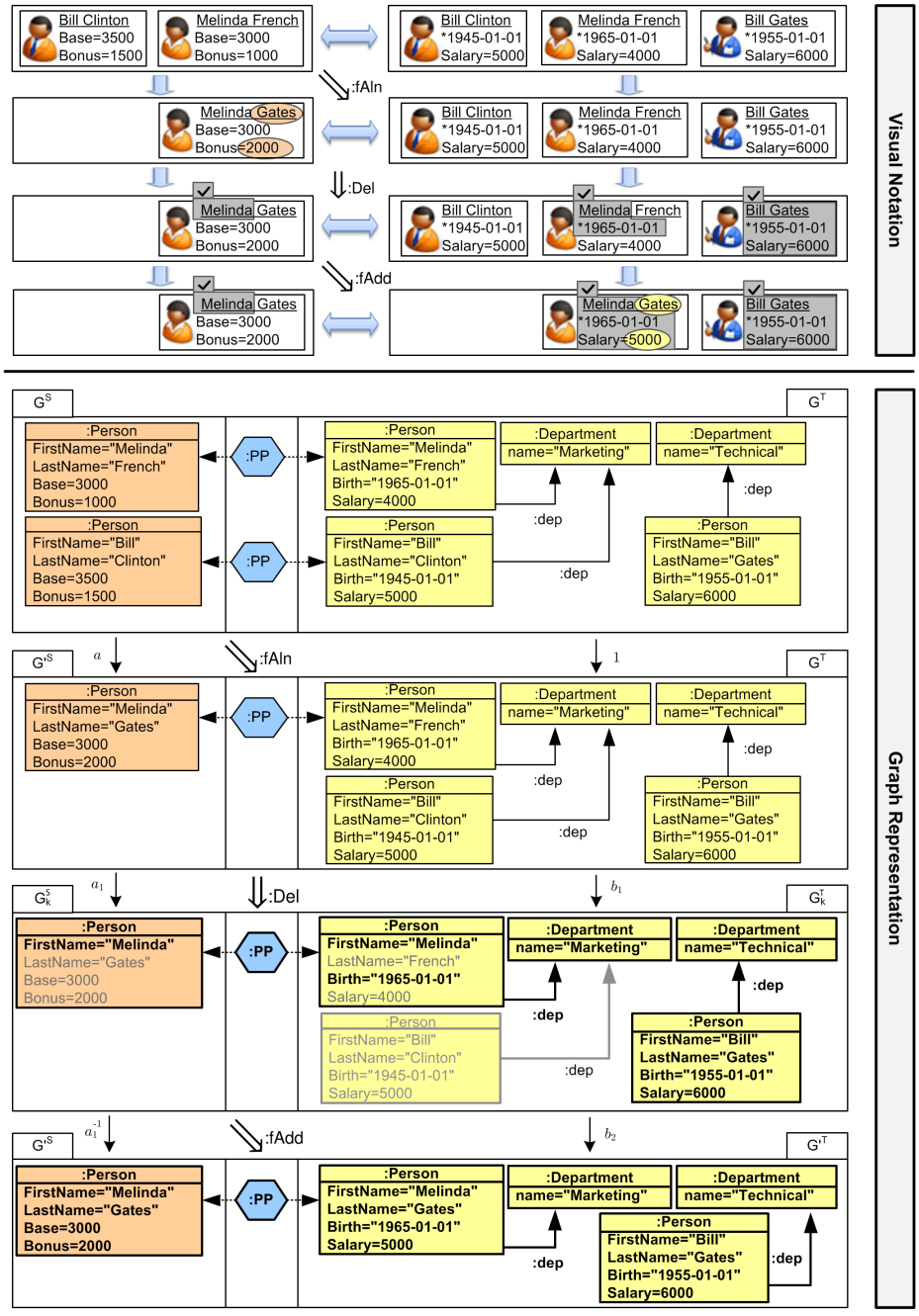
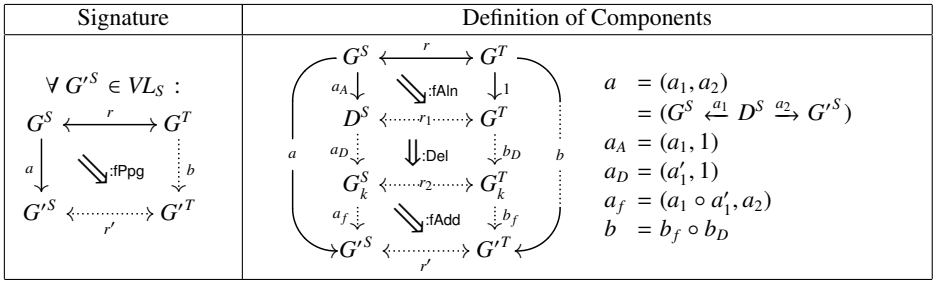


Fig. 7. Forward propagation in detail: visual notation (top) and graph representation (bottom)




```
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 marking 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

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 $(G_k^S \leftrightarrow G_k^T)$. Finally, fAdd composes the embedding $G_k^S \to G'^S$ with correspondence $(G_k^S \leftrightarrow G_k^T)$ leading to $(G'^S \leftrightarrow G_k^T)$, 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 $r' = (\varnothing, \varnothing)$ 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 state our main result concerning correctness, completeness, and invertibility. The proofs and full technical details are provided in the technical report [15]. 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 [6], which is used in the automated analysis engine of the tool AGG [23]. A critical pair specifies a conflict be-

tween two rules in minimal context. Solving a conflict means to find compatible merging transformation steps, which is formalized by the notion of strict confluence [6]. 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. By Fact 2 in [15], we provided 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.*

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 [14] 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.*

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 2 in [15], 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 [15], 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 [23]. 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 [23] 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 [14]. 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 [17,21,9,8], and in the implementation of QVT [12]. Moreover, several formal results are available concerning correctness, completeness, termination [7,10], functional behavior [16,10], and optimization with respect to the efficiency of their execution [14,18,10]. The presented constructions for performing model transformations and model synchronizations are inspired by Schürr et al. [20,21] and Giese et al. [8,9], 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 [22] and Diskin described a more general delta-based view within the *tile algebra* framework [3]. The constructions in the present paper are inspired by tile algebra and follow the general framework presented by Diskin et al. [4], 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 [9]. 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. [8] and Greenyer et al. [13] proposed to extend the preservation of substructures by allowing for the reuse of any *partial* substructure of a rule causing, however, nondeterministic 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. [9,8] 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 [15].

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) [7,14], we have presented in this paper a formal TGG framework for model synchronization inspired by [9,8,20,21]. 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 (Fact 1) based on general results for TGGs in [14].

In future work, the tool Henshin based on AGG [23] will be extended to implement the synchronization algorithm for forward propagation in Fig. 8. Moreover, the relationship with lenses [22] and delta based bidirectional transformations [4] will be studied in more detail, especially in view of composition of lenses leading to composition of synchronization operations. Furthermore, we will study synchronization based on non-deterministic forward and backward propagation operations in more detail.

References

1. 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)
2. 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)
3. 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)

4. 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)
5. 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)
6. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theor. Comp. Science, Springer (2006)
7. 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)
8. Giese, H., Hildebrandt, S.: Efficient Model Synchronization of Large-Scale Models . Tech. Rep. 28, Hasso Plattner Institute at the University of Potsdam (2009)
9. Giese, H., Wagner, R.: From model transformation to incremental bidirectional model synchronization. Software and Systems Modeling 8(1), 21–43 (2009)
10. 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)
11. Golas, U., Ehrig, H., Hermann, F.: Formal Specification of Model Transformations by Triple Graph Grammars with Application Conditions. EC-EASST 39 (2011)
12. 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)
13. 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)
14. 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)
15. 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)
16. 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)
17. 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)
18. 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)
19. 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)
20. Schürr, A.: Specification of Graph Translators with Triple Graph Grammars. In: Proc. WG'94. LNCS, vol. 903, pp. 151–163. Springer Verlag, Heidelberg (1994)
21. Schürr, A., Klar, F.: 15 Years of Triple Graph Grammars. In: Proc. ICGT'08. LNCS, vol. 5214, pp. 411–425 (2008)
22. Stevens, P.: Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions . Software and Systems Modeling 9, 7–20 (2010)
23. TFS-Group, TU Berlin: AGG (2011), `http://tfs.cs.tu-berlin.de/agg`