On-the-Fly Construction, Correctness and Completeness of Model Transformations Based on Triple Graph Grammars

Hartmut Ehrig, Claudia Ermel, Frank Hermann, and Ulrike Prange

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany
{ehrig,lieske,frank,uprange}@cs.tu-berlin.de

Abstract. Triple graph grammars (TGGs) are a formal and intuitive concept for the specification of model transformations. Their main advantage is an automatic derivation of operational rules for bidirectional model transformations, which simplifies specification and enhances usability as well as consistency.

In this paper we continue previous work on the formal definition of model transformations based on triple graph rules with negative application conditions (NACs). The new notion of partial source consistency enables us to construct consistent model transformations on-the-fly instead of analyzing consistency of completed model transformations.

We show the crucial properties termination, correctness and completeness (including NAC-consistency) for the model transformations resulting from our construction. Moreover, we define parallel independence for model transformation steps which allows us to perform partial-order reduction in order to improve efficiency. The results are applicable to several relevant model transformations and in particular to our example transformation from class diagrams to database models.

Keywords: Model transformation, triple graph grammars, correctness.

1 Introduction

Model transformations based on triple graph grammars (TGGs) have been introduced by Schürr in [1]. TGGs are grammars that generate languages of graph triples, consisting of a source graph G^S and a target graph G^T, together with a correspondence graph G^C "between" them. From a TGG, operational rules can be derived which define various model integration tasks, such as consistency checking, consistency recovery and bidirectional model transformation. Since 1994, several extensions of the original TGG definitions have been published [2,3,4], and various kinds of applications have been presented [5,6,7].

For source-to-target model transformation, so-called *forward* transformation, we derive rules which take the source graph as input and produce a corresponding target graph. Major properties expected to be fulfilled for model transformations are termination, correctness and completeness.

A. Schürr and B. Selic (Eds.): MODELS 2009, LNCS 5795, pp. 241–255, 2009.

In a previous series of papers we focused on the formal definition of TGGs and the analysis of model transformation properties: in [8], we showed how to analyze bi-directional model transformations based on TGGs with respect to information preservation, which is based on a decomposition and composition result for triple graph grammar sequences. Moreover, completeness and correctness of model transformations have been studied on this basis in [9]. In [10], the formal results were extended to TGGs with negative application conditions (NACs), a key concept for many model transformations (see [2]). In contrast to the presented algorithm in [2] we use the concept of source consistency, where the the transformation is controlled by a parsing of the source model, and we introduced NAC consistency as an extension. In this way we could extend several important results to the case of TGGs with NACs. Model transformations based on triple rules with NACs were also analyzed in [11] for a restricted class of triple rules with distinct kernel elements. For this restricted class of triple graph grammars local confluence and termination can be analyzed and thus, model transformations can be checked for functional behavior.

As shown in [12] and [10] the notion of *source consistency* ensures correctness and completeness of model transformations based on triple graph grammars with and without NACs. However, source consistency does not directly guide the construction of the model transformation, because it has to be checked for the complete forward sequence. This means that possible forward sequences have to be constructed until one is found to be source consistent. Additionally, termination of this search is not guaranteed in general.

It is the main contribution of this paper to introduce a construction technique for correct and complete model transformation sequences *on-the-fly*, i.e. correctness and completeness properties of a model transformation need not to be analyzed after completion, but are ensured by construction. In our construction, we check source consistency while creating the forward sequences and define suitable conditions for termination. Thus, re-computations of model transformations may be avoided. Moreover, we present a characterization of parallel independence of forward transformation steps and use this notion for an optimization of efficiency based on partial order reduction [13]. Summing up, the paper provides the basis for efficient implementations of model transformation tools that ensure termination, correctness and completeness.

The paper is structured as follows: Sec. 2 reviews the definition of triple graph grammars with NACs from [10]. In Sec. 3 we introduce an *on-the-fly* construction of source consistent forward transformation sequences, generalizing the notion of source consistency to *partial* source consistency. The on-the-fly construction is analyzed in Sec. 4 regarding correctness and completeness of the model transformations, and termination of the construction. Moreover, parallel independence of forward transformation steps is defined and used to find switch equivalent model transformation sequences by performing an optimization based on partial order reduction. Sec. 5 discusses related work, and Sec. 6 concludes the paper. Our technical report [14] contains full definitions for Sec. 2 and full proofs for the presented results in Secs. 3 and 4.

2 Review of Triple Graph Grammars with NACs

Triple graph grammars [1] are a well known approach for bidirectional model transformations. Models are defined as pairs of source and target graphs, which are connected via a correspondence graph together with its embeddings into these graphs. In [3], Königs and Schürr formalize the basic concepts of triple graph grammars in a set-theoretical way, which is generalized and extended by Ehrig et al. in [8] to typed, attributed graphs. In this section, we briefly review triple graph grammars with negative application conditions (NACs) [2,10].

A triple graph $G = (G^S \xleftarrow{s_G} G^C \xrightarrow{t_G} G^T)$ consists of three graphs G^S, G^C, and G^T, called source, correspondence, and target graphs, together with two graph morphisms $s_G : G^C \to G^S$ and $t_G : G^C \to G^T$. A triple graph morphism $m = (m^S, m^C, m^T) : G \to H$ consists of three graph morphisms $m^S : G^S \to H^S$, $m^C : G^C \to H^C$ and $m^T : G^T \to H^T$ such that $m^S \circ s_G = s_H \circ m^C$ and $m^T \circ t_G = t_H \circ m^C$. A typed triple graph G is typed over a triple graph TG by a triple graph morphism $type_G : G \to TG$.

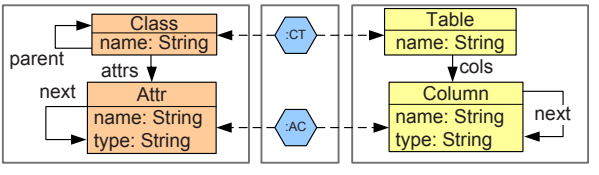


Fig. 1. Triple Type Graph for *CD2RDBM*

Example 1. Fig. 1 shows the type graph TG of the triple graph grammar GG for our example model transformation from class diagrams to database models. The source component of TG defines the structure of class diagrams while in its target component the structure of relational database models is specified. Classes correspond to tables and attributes to columns. Throughout the example, originating from [2] and [8], elements are arranged left, center, and right according to the component types source, correspondence and target. Morphisms starting at a correspondence part are given by dashed arrows. Note that the case study is equipped with attribution, which is based on the concept of E-graphs [15].

Triple rules synchronously build up source and target graphs as well as their correspondence graphs, i.e. they are non-deleting. A triple rule tr is an injective triple graph morphism $tr = (tr^S, tr^C, tr^T) : L \to R$ and w.l.o.g. we assume tr to be an inclusion. Given a triple rule $tr : L \to R$, an injective $m : L \to G$, a triple graph transformation step (TGT-step) $G \overset{tr,m,n}{\Longrightarrow} H$ from G to a triple graph H is given by a pushout of triple graphs with comatch $n : R \to H$ and

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

transformation inclusion $t : G \hookrightarrow H$. A sequence of triple graph transformation steps is called triple (graph) transformation sequence, short: TGT-sequence. Furthermore, a triple graph grammar $TGG = (S, TG, TR)$ consists of a triple start graph S, triple type graph TG and a set TR of triple rules.

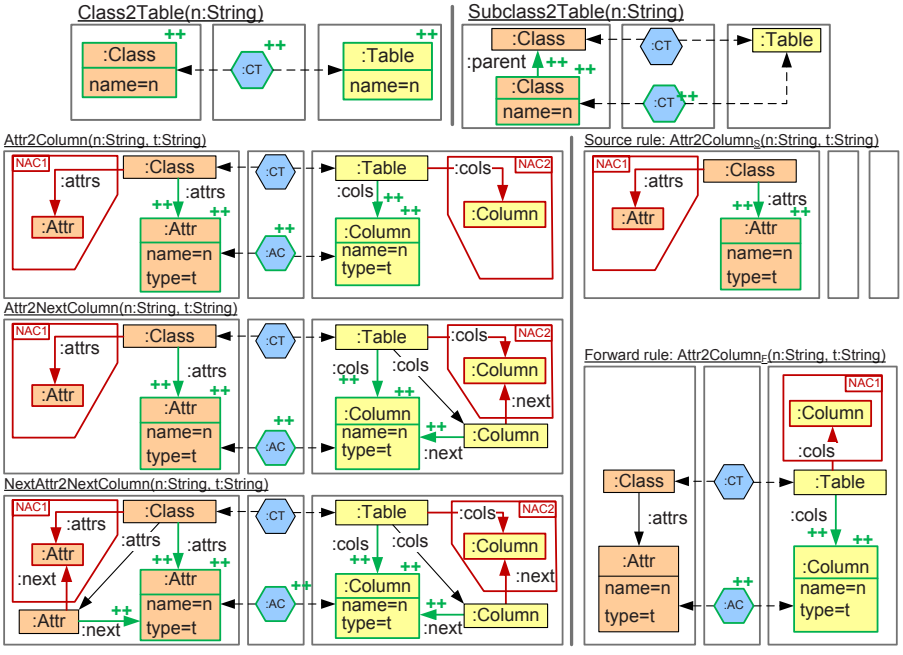


Fig. 2. Rules for the Model Tranformation *Class2Table*

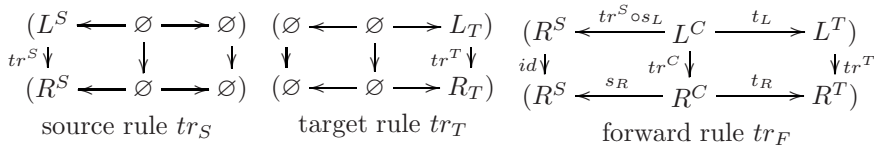
Example 2 (Triple Rules). The top line of Fig. 2 shows two triple rules in short notation. Left and right hand side of a rule are depicted in one triple graph. Elements, which are created by the rule, are labeled with green "++" and marked by green line coloring. Rule "*Class2Table*" synchronously creates a class in a class diagram with its corresponding table in the relational database. Accordingly, subclasses are connected to the tables of its super classes. The further rules contain NACs which we introduce next.

The extension of the results of this paper to the case with attributes is straight forward, because all results can be shown in the framework of weak adhesive HLR categories [15]. According to [10] we present negative application conditions for triple rules. In most case studies of model transformations source-target NACs are sufficient and we regard them as the standard case.

Definition 1 (Negative Application Conditions). *Given a triple rule* $tr = (L \to R)$*, a general negative application condition (NAC)* (N, n) *consists of a*

triple graph N and an injective triple graph morphism $n : L \to N$. A NAC with $n = (n^S, id_{L_C}, id_{L_T})$ is called source NAC *and a NAC with $n = (id_{L_S}, id_{L_C}, n^T)$ is called* target NAC*. This means that source-target NACs, i.e. either source or target NACs, prohibit the existence of certain structures either in the source or in the target part only.*

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



Operational rules for model transformations are automatically derived from the set of triple rules TR. From each rule tr of TR we derive a forward rule tr_F for forward transformation sequences and a source rule tr_S for the construction resp. parsing of a model of the source language. Analogously, we derive a target rule tr_T for models of the target language and backward rules tr_B, which are not presented explicitly. Furthermore, tr_S contains all source NACs of tr and tr_F as well as tr_T contain all target NACs of tr. TR_S, TR_T and TR_F denote the sets of all source, target resp. forward rules derived from TR.

A set of triple rules TR with NACs and start graph $\varnothing$ generates a visual language VL of integrated models, i.e. models with elements in the source, target and correspondence component. Source language VL_S and target language VL_T are derived by projection to the triple components, i.e. $VL_S = proj_S(VL)$ and $VL_T = proj_T(VL)$. The set VL_{S0} of models that can be generated resp. parsed by the set of all source rules TR_S is possibly larger than VL_S and we have $VL_S \subseteq VL_{S0} = \{G_S \,|\, \varnothing \Rightarrow^* (G_S \leftarrow \varnothing \rightarrow \varnothing)$ *via* $TR_S\}$. Analogously, we have $VL_T \subseteq VL_{T0} = \{G_T \,|\, \varnothing \Rightarrow^* (G_T \leftarrow \varnothing \rightarrow \varnothing)$ *via* $TR_T\}$.

Example 3 (Triple Rules with NACs). Examples for triple rules with NACs and derived rules are given in Fig. 2. NACs are indicated by red frames with labels "NAC" and they control the construction of attribute lists in the source part and corresponding column lists in the target part. The first attribute of a class is either created by the rule "Attr2Column" or by "Attr2NextColumn" while rule "NextAttr2NextColumn" extends an existing list of attributes. Lists of columns are initialized by rule "Attr2Column" only, because there is no inheritance structure in data base tables, and they are extended by the other two rules. The source rule tr_S and forward rule tr_F of $tr =$"Attr2Column" are shown in the right part of Fig. 2, where tr_S contains the source NAC (NAC1) and tr_F the target NAC (NAC2) of tr. Forward transformations using the derived rules according to Section 3 process the attribute lists in the natural order, i.e. starting with the root element of a list.

As introduced in [8,10] we are now able to define model transformations based on source consistent forward transformations $G_0 \Rightarrow^* G_n$ via $(tr_{1,F}, \ldots, tr_{n,F})$,

short $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$. Source consistency of $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ means that there is a source sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0$ such that the sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is match consistent, i.e. the S-component of each match $m_{i,F}$ of $tr_{i,F}(i = 1..n)$ is uniquely determined by the comatch $n_{i,S}$ of $tr_{i,S}$, where $tr_{i,S}$ and $tr_{i,F}$ are source and forward rules of the same triple rules tr_i. Altogether the forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is controlled by the corresponding source sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0$, which is unique in the case of match consistency.

Definition 2 (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 NAC- as well as source consistent forward* TGT*-sequence* $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ *with* $G_S = proj_S(G_0)$ *and* $G_T = proj_T(G_n)$.

A model transformation $MT : VL_{S0} \Rrightarrow VL_{T0}$ *is defined by all model transformation sequences* $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ *with* $G_S \in VL_{S0}$ *and* $G_T \in VL_{T0}$.

Finally, let us note that we have shown in [8,10] that each TGT-sequence $G_0 \overset{tr^*}{\Longrightarrow} G_n$ with NACs can be decomposed into a match consistent TGT-sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ with NACs and vice versa, which is the basis for correctness and completeness of model transformations in Sec. 4.

3 On-the-Fly Construction of Model Transformations

In order to construct a model transformation sequence $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ according to Def. 2 from a given G_S there have been two alternatives up to now [8,10]: Either we construct a parsing sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0$ first and then try to extend it to a match consistent sequence $\varnothing \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$, or we construct directly a forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ and check afterwards, whether it is source consistent. This means that many candidates of forward transformation sequences may have to be constructed before a source consistent one is found.

We present an on-the-fly check of source consistency using the new notion of partial source consistency. The construction proceeds stepwise and constructs partial source consistent forward sequences. For each step the possible matches of model transformation rules are filtered, such that sequences that will not lead to a source consistent one are rejected as soon as possible. Simultaneously, the corresponding source sequences of the forward sequences are constructed on-the-fly leading to complete source sequences for the complete forward sequences. Intuitively, this can be seen as an on-the-fly parsing of the source model.

Partial source consistency of a forward sequence, which is necessary for a complete model transformation, requires a corresponding source sequence such that both sequences are partially match consistent. This means that the matches of the forward sequence are controlled by an automatic parsing of the source model, given by inverting the source sequence. We incrementally extend partially source consistent sequences and can derive complete source consistent sequences ensuring that all elements of the source model are translated exactly once.

Definition 3 (Partial Match and Source Consistency). *Let TR be a set of triple rules with source and target NACs and let TR_F be the derived set of forward rules with target NACs. A NAC-consistent sequence*

$$\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$$

defined by pushout diagrams (1) *and* (3) *for $i = 1 \dots n$ with $G_0^C = \varnothing$, $G_0^T = \varnothing$ and inclusion $g_n : G_{n0} \hookrightarrow G_0$ is called* partially match consistent*, if diagram* (2) *commutes for all i, which means that the source component of the forward match $m_{i,F}$ is determined by the comatch $n_{i,S}$ of the corresponding step of the source sequence with $g_i = g_n \circ t_{n,S} \dots t_{i-1,S}$.*

$$\begin{array}{ccccccc}
L_{i,S} & \xrightarrow{tr_{i,S}} & R_{i,S} & \longrightarrow & L_{i,F} & \xrightarrow{tr_{i,F}} & R_{i,F} \\
{\scriptstyle m_{i,S}}\downarrow & (1) & \downarrow{\scriptstyle n_{i,S}} & (2) & {\scriptstyle m_{i,F}}\downarrow & (3) & \downarrow{\scriptstyle n_{i,F}} \\
G_{i-1,0} & \xrightarrow[t_{i,S}]{} & G_{i,0} & \xrightarrow[g_i]{} & G_0 & \longrightarrow & G_{i-1} \xrightarrow[t_{i,F}]{} G_i
\end{array}$$

A NAC-consistent forward sequence $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is partially source consistent, if there is a source sequence $\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0}$ with inclusion $G_{n0} \overset{g_n}{\hookrightarrow} G_0$ such that $G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is partially match consistent.

Remark 1

1. If $g_n = id_{G_0}$, partial match consistency coincides with match consistency.
2. For $n = 0$ the partially match consistent sequence is given by $g_0 : G_{00} \hookrightarrow G_0$.

Example 4 (Partial Match and Source Consistency). Let us consider a sequence starting with triple graph G_0 (depicted in the center of Fig. 3) which represents a class diagram consisting of one class with two linked attributes. G_0 will be mapped to a corresponding table with two linked columns. Note that for this example, we assume the triple rules shown in Fig. 2, but first without NACs.

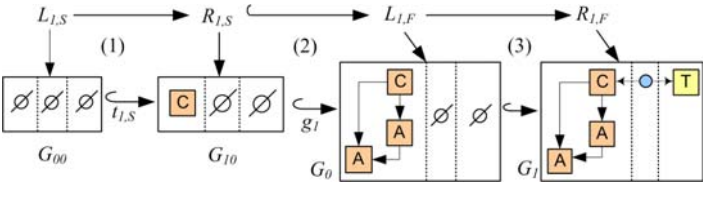


Fig. 3. Step 1 of the partially match-consistent sequence

In the first step ($i = 1$), shown in Fig. 3, we apply rule $tr_{1,S} = Class2Table_S$ to the empty start graph G_{00} yielding the source graph G_{10} which contains one class. Obviously, G_{10} is included in G_0. Hence, diagram (2) commutes for step 1. The corresponding forward rule $tr_{1,F} = Class2Table_F$ is applied to G_0 and maps the class node to a table node, resulting in G_1. For step $i = 2$ (not depicted), we apply the source rule $tr_{2,S} = Attr2Columns_S$ to graph G_{10} which

adds an attribute and links it to the class. The result graph is G_{20}. Again, G_{20} is included in G_0, which is included in G_1. The corresponding forward rule $tr_{2,F} = Attr2Column_F$ is applied to G_1, resulting in G_2, where the upper attribute of the class now is mapped to a column of the table.

In the third step ($i = 3$), shown in Fig. 4, we apply the same source rule once more, i.e. $tr_{3,S} = Attr2Column_S$, and add a second attribute to G_{20}, resulting in source graph G_{30}. This graph is included in G_0, which in turn is included in G_2. Diagram (2) commutes for step 3. The application of the corresponding forward rule $tr_{3,F} = Attr2Column_F$ at the co-match of $tr_{3,S}$ yields G_3, where now also the second attribute is mapped to a column of the table.

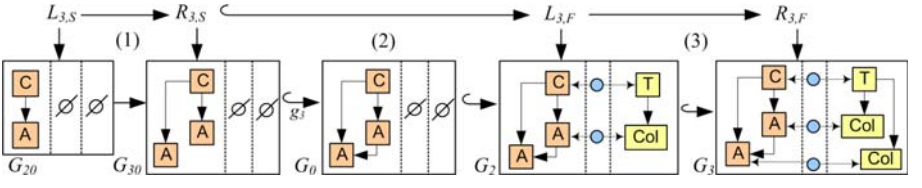


Fig. 4. Step 3 of the partially match-consistent sequence

Since for each considered step, diagram (2) of Def. 3 commutes, we conclude that sequence $\emptyset = G_{00} \overset{tr_{1,S}}{\Longrightarrow} G_{10} \overset{tr_{2,S}}{\Longrightarrow} G_{20} \overset{tr_{3,S}}{\Longrightarrow} G_{30} \overset{g_n}{\hookrightarrow} G_0 \overset{tr_{1,F}}{\Longrightarrow} G_1 \overset{tr_{2,F}}{\Longrightarrow} G_2 \overset{tr_{3,F}}{\Longrightarrow} G_3$ is partially match consistent. Hence, the forward sequence $G_0 \overset{tr_{1,F}}{\Longrightarrow} G_1 \overset{tr_{2,F}}{\Longrightarrow} G_2 \overset{tr_{3,F}}{\Longrightarrow} G_3$ is partially source consistent. Note that the forward sequence, although being partially source consistent, cannot be extended to a complete source consistent sequence. The reason is that after the third step, we do not find a new partially source consistent match for some $tr_{4,F}$. We will analyze in Ex. 6 what went wrong and how NACs in triple rules can help to improve the construction of valid source consistent sequences.

In order to provide an improved construction of source consistent forward sequences we characterize valid matches by introducing the following notion of forward consistent matches. The formal condition of a forward consistent match is given by a pullback diagram where both matches satisfy the corresponding NACs. Intuitively, it specifies that the effective elements of the forward rule are matched for the first time in the forward sequence (see Interpretation 1 below).

Definition 4 (Forward Consistent Match). *Given a partially match consistent sequence* $\emptyset = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n-1,0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_{n-1}$ *then a match* $m_{n,F} : L_{n,F} \to G_{n-1}$ *for* $tr_{n,F} : L_{n,F} \to R_{n,F}$ *is called* forward consistent *if there is a source match* $m_{n,S}$ *such that diagram* (1) *is a pullback and the matches* $m_{n,F}$ *and* $m_{n,S}$ *satisfy the corresponding target and source NACs, respectively.*

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

Interpretation 1. The pullback property of (1) means that the intersection of the match $m_{n,F}(L_{n,F})$ and the source graph $G_{n-1,0}$ constructed so far is equal to $m_{n,F}(L_{n,S})$, the match restricted to $L_{n,S}$, i.e. we have

$$(2) : m_{n,F}(L_{n,F}) \cap G_{n-1,0} = m_{n,F}(L_{n,F}).$$

This condition can be checked easily and $m_{n,S} : L_{n,S} \to G_{n-1,0}$ is uniquely defined by restriction of $m_{n,F} : L_{n,F} \to G_{n-1}$. Furthermore, as a direct consequence of (2) we have

$$(3) : m_{n,F}(L_{n,F} \setminus L_{n,S}) \cap G_{n-1,0} = \varnothing.$$

On the one hand, the source elements of $L_{n,F} \setminus L_{n,S}$ - called effective elements - are the elements to be transformed by the next step of the forward transformation sequence. On the other hand, $G_{n-1,0}$ contains all elements that were matched by the preceding forward steps, because matches of the forward sequence coincide on the source part with comatches of the source sequence. Hence, condition (3) means that the effective elements were not matched before, i.e. they do not belong to $G_{n-1,0}$.

Example 5 (Forward Consistent Match). In the partial match consistent sequence from Ex. 4, all forward rule matches are forward consistent. Consider for example the situation in step 3, shown in Fig. 5, where all mappings have been indicated explicitly by equal numbers. We can see that $L_{3,F} \cap G_{20} = L_{3,S}$, which implies that Diagram (1) from Def. 4 is a pullback. Analogously, the matches from forward rules in steps 1 and 2 are also forward consistent.

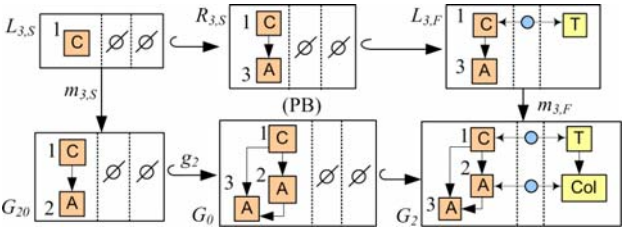


Fig. 5. Forward consistent match from step 3

In the following improved construction of model transformations, we check the matches to be forward consistent. This allows us to filter the available matches to those which can lead to correct model transformations while those matches that cannot lead to correct model transformations are rejected.

Theorem 1 (On-the-Fly Construction of Model Transformations). *Given a triple graph G_0 with $G_0^C = G_0^T = \varnothing$, execute the following steps:*

1. *Start with $G_{00} = \varnothing$ and $g_0 : G_{00} \hookrightarrow G_0$.*
2. *For $n > 0$ and an already computed partially source consistent sequence $s = \langle G_0 \overset{tr_F^*}{\Longrightarrow} G_{n-1} \rangle$ with $\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n-1,0}$ and embedding g_{n-1} :*

$G_{n-1,0} \hookrightarrow G_0$ *find a (not yet considered) forward consistent match for some* $tr_{n,F}$ *leading to a partially source consistent sequence* $G_0 \overset{tr_F^*}{\Longrightarrow} G_{n-1} \overset{tr_{n,F}}{\Longrightarrow} G_n$ *with* $G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n-1,0} \overset{tr_{n,S}}{\Longrightarrow} G_{n0}$ *and embedding* $g_n : G_{n0} \hookrightarrow G_0$*. If there is no such match,* s *cannot be extended to a source consistent sequence. Repeat until* $g_n = id_{G_0}$ *or no new forward consistent matches can be found.*

3. *If the procedure terminates with* $g_n = id_{G_0}$*, then* $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ *is source consistent leading to a model transformation sequence* $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ *with* G_S *and* G_T *being the source and target models of* G_0 *and* G_n*.*

The on-the-fly construction does not restrict the choice of a suitable n, $tr_{n,F}$, and match in Step 2. Hence, different search algorithms are possible, e.g.

- *Depth First:* If we increase n after every iteration, and only decrease n by 1 if no more new forward consistent matches can be found, a depth-first search is performed.
- *Breadth First:* If we increase n only after all forward consistent matches for n are considered, the construction performs a breadth-first search.

Depending on the type of the model transformation, other search strategies may be reasonable. In Sec. 4, we show how to make the construction more efficient by analyzing independent transformations.

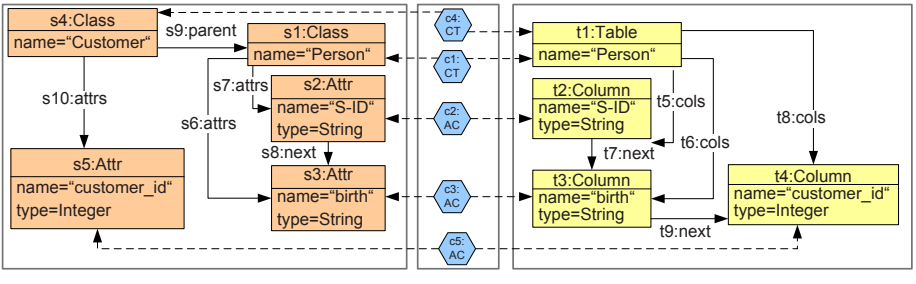


Fig. 6. G_5 of Forward Sequence

Example 6 (On-the-Fly Construction). Let us assume we have found already the partial match consistent sequence from Ex. 4 by depth-first search. All forward rule matches found so far are forward consistent. But after the third rule application step ($i = 3$), we do not find a new partial source consistent match for some $tr_{4,F}$. We cannot extend the sequence to a source consistent one, because there is no triple rule for inserting a *next* link between two existing attributes. The mistake we made was to use the wrong rule $Attr2Columns_S$ for the insertion of the second attribute. If we had used rule $NextAttr2NextColumns_S$ instead, we would have constructed a sequence which could be extended to a source consistent sequence. If a sequence cannot be

extended to a source-consistent one, we have two choices: either, we have to try to apply a different rule in a previous step, or we restrict the applicability of our triple rules, e.g. by adding negative application conditions. Here, we can use the NACs in Fig. 2, which ensure that only one attribute-adding rule is applicable in each step. An example for a source-consistent sequence, constructed by partially source consistent sequences according to Thm. 1, is the model transformation $(G_S = G_{0,S}, G_0 \stackrel{tr_F^*}{\Longrightarrow} G_5, G_T = G_{5,T})$, where G_5 (shown in Fig. 6) is generated by the forward sequence $G_0 \stackrel{Class2Table}{\Longrightarrow} G_1 \stackrel{Attr2Col}{\Longrightarrow} G_2 \stackrel{Subclass2Table}{\Longrightarrow} G_3 \stackrel{NextAttr2NextCol}{\Longrightarrow} G_4 \stackrel{Attr2NextCol}{\Longrightarrow} G_5$, and G_0 is generated by the corresponding source sequence $\varnothing \stackrel{tr_S^*}{\Longrightarrow} G_0$. All elements in Fig. 6 are labeled with numbers. The following table specifies the matches and the created objects for each transformation step. Note that we cannot accidentally apply the rule $Class2Table_F$ at subclasses, because in this case the transformation will not become source consistent - the edge of the type "parent" will be missing.

	Source Sequence Elements		Forward Sequence Elements	
Step	Matched	Created	Matched	Created
1		s1	s1	c1,t1
2	s1	s2,s7	s1,s2,s7,c1,t1	c2,t2,t5
3	s1	s4,s9	s1,c1,t1,s4,s9	c4
4	s1,s2,s7	s3,s8	s1-s3,s6-s8,c1,t1,t2,t5	c3,t3,t6,t7
5	s4	s5,s10	s4,s5,s10,c4,t1,t3,t6	c5,t4,t8,t9

4 Analysis and Improvement of the Construction

In this section, we analyze the on-the-fly construction in Thm. 1 regarding correctness, completeness, and termination of the model transformations and show how to improve efficiency by parallel independence, which allows partial order reduction.

The on-the-fly construction is correct, which means that if it terminates both the source and target models of the resulting model transformations are valid models of the source and target languages, respectively. Moreover, it is also complete, which means that for any source model the procedure can find a model transformation sequence leading to a corresponding target model.

Theorem 2 (Correctness and Completeness)

- Correctness: *If the on-the-fly construction terminates with $g_n = id_{G_0}$, then the resulting model transformation $(G_S, G_0 \stackrel{tr_F^*}{\Longrightarrow} G_n, G_T)$ is correct, i.e. $G_S \in VL_S$ and $G_T \in VL_T$.*
- Completeness: *For each $G_S \in VL_S$ there exists $G_T \in VL_T$ with a model transformation $(G_S, G_0 \stackrel{tr_F^*}{\Longrightarrow} G_n, G_T)$, which can be obtained by the on-the-fly construction.*

Remark 2. Dually, for each $G_T \in VL_T$ there exists $G_S \in VL_S$ where the corresponding model transformation can be obtained dually by partially target consistent sequences.

In general, the termination of the on-the-fly construction cannot be guaranteed. But for the case that all source rules create new elements also the termination of the on-the-fly construction is ensured.

Theorem 3 (Termination). *The on-the-fly construction of a triple graph G_0 with $G_0^C = G_0^T = \varnothing$ terminates if all source rules $tr_{i,S}$ are creating, i.e. $R_{i,S} \setminus L_{i,S} \neq \varnothing$.*

Example 7 (Termination). The on-the-fly construction of triple graph G_5 in Ex. 6 terminates because all of the used source rules in the source sequence are creating, which can be seen in the left column of the table.

In the following, we describe how to improve efficiency by analyzing parallel independence of extensions. Two partially match consistent sequences which differ only in the last rule application are parallel independent if the last rule applications are parallel independent both for the source and forward sequence, and, in addition, if the embeddings into the given graph G_0 are compatible.

Definition 5 (Parallel Independence of Partially Match Consistent Extensions). *Two partially match consistent sequences*

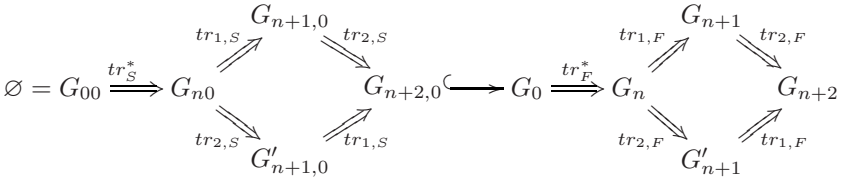
$$\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_{1,S}}{\Longrightarrow} G_{n+1,0} \overset{g_{n+1}}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n \overset{tr_{1,F}}{\Longrightarrow} G_{n+1} \text{ and}$$
$$\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_{2,S}}{\Longrightarrow} G'_{n+1,0} \overset{g'_{n+1}}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n \overset{tr_{2,F}}{\Longrightarrow} G'_{n+1}$$

are parallel independent *if $G_{n0} \overset{tr_{1,S}}{\Longrightarrow} G_{n+1,0}$ and $G_{n0} \overset{tr_{2,S}}{\Longrightarrow} G'_{n+1,0}$ as well as $G_n \overset{tr_{1,F}}{\Longrightarrow} G_{n+1}$ and $G_n \overset{tr_{2,F}}{\Longrightarrow} G'_{n+1}$ are parallel independent leading to the diagram (1_S) and (1_F), and diagram (2) is a pullback.*

$$\begin{array}{ccc} G_{n0} & \overset{tr_{1,S}}{\Longrightarrow} & G_{n+1,0} \\ {\scriptstyle tr_{2,S}}\downarrow & (1_S) & \downarrow{\scriptstyle tr_{2,S}} \\ G'_{n+1,0} & \underset{tr_{1,S}}{\Longrightarrow} & G_{n+2,0} \end{array} \qquad \begin{array}{ccc} G_n & \overset{tr_{1,F}}{\Longrightarrow} & G_{n+1} \\ {\scriptstyle tr_{2,F}}\downarrow & (1_F) & \downarrow{\scriptstyle tr_{2,F}} \\ G'_{n+1} & \underset{tr_{1,F}}{\Longrightarrow} & G_{n+2} \end{array} \qquad \begin{array}{ccc} G_{n0} & \overset{t_{1,S}}{\hookrightarrow} & G_{n+1,0} \\ {\scriptstyle t_{2,S}}\downarrow & (2) & \downarrow{\scriptstyle g_{n+1}} \\ G'_{n+1,0} & \underset{g'_{n+1}}{\hookrightarrow} & G_0 \end{array}$$

In the case of parallel independence of the extensions, both extensions can be extended both in the source and forward sequences leading to two longer partially match consistent sequences which are switch-equivalent.

Theorem 4 (Partial Match Consistency with Parallel Independence). *If $\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_{1,S}}{\Longrightarrow} G_{n+1,0} \overset{g_{n+1}}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n \overset{tr_{1,F}}{\Longrightarrow} G_{n+1}$ and $\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_{2,S}}{\Longrightarrow} G'_{n+1,0} \overset{g'_{n+1}}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n \overset{tr_{2,F}}{\Longrightarrow} G'_{n+1}$ are* parallel independent *then the following upper and lower sequences are partially match consistent and called* switch equivalent*.*



Example 8 (Parallel Independence). Consider the sequence of rule applications in Ex. 6. Here, we may switch step 2 and step 3 without changing the result G_5 since the sequences $\varnothing = G_{00} \overset{Class2Table_S}{\Longrightarrow} G_{10} \overset{Attribute2Column_S}{\Longrightarrow} G_{2,0} \overset{g_2}{\hookrightarrow} G_0 \overset{Class2Table_F}{\Longrightarrow} G_1 \overset{Attribute2Column_F}{\Longrightarrow} G_2$ and $\varnothing = G_{00} \overset{Class2Table_S}{\Longrightarrow} G'_{10} \overset{Subclass2Table_S}{\Longrightarrow} G'_{2,0} \overset{g'_2}{\hookrightarrow} G_0 \overset{Class2Table_F}{\Longrightarrow} G_1 \overset{Subclass2Table_F}{\Longrightarrow} G'_2$ are parallel independent.

We can analyze parallel independence on-the-fly for the forward steps which are applicable to the current intermediate triple graph. Based on the induced partial order of dependencies between the forward steps we can apply several techniques of partial order reduction in order to improve efficiency. This means that we can neglect remaining switch-equivalent sequences, if one of them has been constructed. This improves efficiency of corresponding depth-first and breadth-first algorithms. For an overview of various approaches concerning partial order reduction see [13], where also benchmarks show that these techniques can dramatically reduce complexity.

5 Related Work and Evaluation of Our Approach

Since 1994, several extensions of the original TGG definitions have been published [2,3,4], and various kinds of applications have been presented [5,6,7]. For an extensive overview see [2]. A new extension of TGGs towards declarative, pattern-based model transformation is presented in [16], where triple rules are derived from triple graph constraints.

Furthermore, Kindler and Wagner [7] discuss that several applications of model transformations based on TGGs require an efficient strategy for finding a correct transformation sequence because of the non-deterministic character of the matching of forward rules. A new strategy for controlling the construction of a model transformation was given in [2], where elements of the source model are distinguished for each step of the model transformation whether they were translated so far. In this paper we have formalized this separation by specifying which elements were matched so far and we call the new matched elements in an intermediate model transformation step effective elements (see Def. 4).

As stated in Sec. 1 this paper extends concepts and results of our previous papers [8,11,9,10]. In the following we explain how our approach complies with the design principles of the "Grand Research Challenge of the Triple Graph Grammar Community", which was formulated by Schürr et al. in [2]:

1. *Correctness:* Model transformations shall be correct in the way that whenever the algorithm translates a source model G_S into a target model G_T then

there has to be a triple graph $G = (G_S \leftarrow G_C \rightarrow G_T) \in VL$. This property is shown in Thm. 2 for an algorithm based on our construction in Thm. 1.
2. *Completeness and Termination:* Completeness means that the algorithm translates each model $G_S \in VL_S$. This property subsumes Termination. Both properties are ensured for our construction by Thm. 2 and Thm. 3 if triple rules are creating on the source part.
3. *Efficiency:* Model transformations shall have polynomial space and time complexity with exponent k the maximal number of elements of a rule. Our construction does not guarantee this requirement in general. But note that the algorithm in [2] only meets this condition because it avoids backtracking by aborting a translation when the chosen sequence of model transformation steps does not lead to a target model, even if there may be a possible sequence. Therefore, completeness is not achieved in [2]. By Thm. 4 we are able to perform partial order reduction, which has shown to provide massive power for the reduction of complexity (see e.g. [13]).
4. *Expressiveness:* Features that are urgently needed for solving practical problems like NACs and attribute conditions shall be captured. Both, NACs and attributes are handled by our approach. It remains open, whether our restriction to source-target NACs rules out some interesting practical applications.

6 Conclusion and Future Work

In this paper we have given a new formal construction of model transformations based on triple graph grammars including crucial properties like NAC-consistency, correctness, completeness and a sufficient condition for termination. In contrast to previous formal constructions in [1,8,10] the new construction avoids a parsing of the source graph beforehand or afterwards, but allows to construct simultaneously NAC-consistent forward and source transformation sequences leading to an on-the-fly construction of model transformations. Moreover, we have shown correctness and completeness of this on-the-fly construction and termination for triple rules with non-identical source part. Currently, these constructions are being implemented by us based on Mathematica libraries [17].

Finally, we studied parallel independence of model transformation steps, which allows us to perform partial-order reduction in order to improve efficiency of the construction. We have not analyzed local confluence in this paper, which - together with termination - leads to functional behaviour of the model transformation. We are confident that our concept of parallel independence can be extended to study critical pairs and local confluence for model transformation sequences based on existing approaches for graph transformation systems [15] including tool support by AGG [18]. Furthermore, additional correctness criteria shall be developed for the case that source and target languages VL_S and VL_T are defined independently of the triple graph language VL generated by the TGG.

References

1. Schürr, A.: Specification of Graph Translators with Triple Graph Grammars. In: Mayr, E.W., Schmidt, G., Tinhofer, G. (eds.) WG 1994. LNCS, vol. 903, pp. 151–163. Springer, Heidelberg (1995)

2. Schürr, A., Klar, F.: 15 Years of Triple Graph Grammars. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS, vol. 5214, pp. 411–425. Springer, Heidelberg (2008)
3. Königs, A., Schürr, A.: Tool Integration with Triple Graph Grammars - A Survey. ENTCS 148, 113–150 (2006)
4. Guerra, E., de Lara, J.: Attributed Typed Triple Graph Transformation with Inheritance in the Double Pushout Approach. Technical Report UC3M-TR-CS-2006-00, Universidad Carlos III, Madrid (2006)
5. Taentzer, G., Ehrig, K., Guerra, E., de Lara, J., Lengyel, L., Levendovsky, T., Prange, U., Varro, D., Varro-Gyapay, S.: Model Transformation by Graph Transformation: A Comparative Study. In: Proc. WMTP 2005 (2005)
6. Guerra, E., de Lara, J.: Model View Management with Triple Graph Grammars. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) ICGT 2006. LNCS, vol. 4178, pp. 351–366. Springer, Heidelberg (2006)
7. Kindler, E., Wagner, R.: Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios. Technical Report TR-ri-07-284, Universität Paderborn (2007)
8. Ehrig, H., Ehrig, K., Ermel, C., Hermann, F., Taentzer, G.: Information Preserving Bidirectional Model Transformations. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 72–86. Springer, Heidelberg (2007)
9. Ehrig, H., Ermel, C., Hermann, F.: On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars. In: Karsai, G., Taentzer, G. (eds.) Proc. of GraMoT 2008. ACM, New York (2008)
10. Ehrig, H., Hermann, F., Sartorius, C.: Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions. Electronic Communications of the EASST 18 (to appear, 2009)
11. Ehrig, H., Prange, U.: Formal Analysis of Model Transformations Based on Triple Graph Rules with Kernels. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS, vol. 5214, pp. 178–193. Springer, Heidelberg (2008)
12. Ehrig, H., Ehrig, K., Hermann, F.: From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. Electronic Communications of the EASST 10, 1–14 (2008)
13. Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems. LNCS, vol. 1032. Springer, Heidelberg (1996)
14. Ehrig, H., Ermel, C., Hermann, F., Prange, U.: On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars: Long Version. Technical Report 2009-11, TU Berlin (2009), `http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/`
15. Ehrig, H., et al.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs. Springer, Heidelberg (2006)
16. de Lara, J., Guerra, E.: Pattern-based model-to-model transformation. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) ICGT 2008. LNCS, vol. 5214, pp. 426–441. Springer, Heidelberg (2008)
17. Brandt, C., Hermann, F., Engel, T.: Security and Consistency of IT and Business Models at Credit Suisse realized by Graph Constraints, Transformation and Integration using Algebraic Graph Theory. In: Proc. of EMMSAD 2009. LNBIP, vol. 29, pp. 339–352. Springer, Heidelberg (2009)
18. TFS-group, TU Berlin: AGG (2009), `http://tfs.cs.tu-berlin.de/agg`