Technische Universität Berlin

Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik

On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars: Long Version

Hartmut Ehrig, Claudia Ermel, Frank Hermann and Ulrike Prange

Bericht-Nr. 2009-11
ISSN 1436-9915

On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars: Long Version

Hartmut Ehrig, Claudia Ermel, Frank Hermann and Ulrike Prange

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

Bericht-Nr. 2009/11
ISSN 1436-9915

On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars: Long Version = Technical Report =

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

July 13, 2009

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.

1 Introduction

Model transformations based on triple graph grammars (TGGs) have been introduced by Schürr in [14]. 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 for bidirectional model transformations, so-called forward and backward transformation rules, can be derived automatically. Forward transformation rules take the source graph as input and produce a corresponding target graph (together with the correspondence graph linking it to the source graph). Since 1994, several extensions of the original TGG definitions have been published [15, 13, 10], and various kinds of applications have been presented [16, 11, 12].

Major properties expected to be fulfilled for model transformations based on forward transformation rules are termination, correctness and completeness.

in the following sense (see also [15]):

- *Correctness*: Whenever a forward transformation sequence starting with source graph $(G^S \leftarrow \emptyset \rightarrow \emptyset)$ derives triple graph $(G^S \leftarrow G^C \rightarrow G^T)$, then this triple graph must be derivable also by the triple rules in TGG.
- *Completeness*: Whenever a triple graph $(G^S \leftarrow G^C \rightarrow G^T)$ can be generated by the triple rules in TGG, then there is a forward transformation sequence leading from $(G^S \leftarrow \emptyset \rightarrow \emptyset)$ to $(G^S \leftarrow G^C \rightarrow G^T)$.

In a previous series of papers we focused on the formal definition of TGGs and the analysis of model transformation properties: in [3], 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 [6]). In [7], the formal results were extended to TGGs with negative application conditions (NACs), a key concept for many model transformations (see [15]. In contrast to the presented algorithm in [15] for controlling the model transformations we introduced NAC consistency based on source consistent forward sequences. 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 [8] 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 [4] and [7] 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 [9]. 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 [7]. 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.

2 Review of Triple Graph Grammars with NACs

Triple graph grammars [14] 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 [13], 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 [3] to typed, attributed graphs. In this section, we briefly review triple graph grammars with negative application conditions (NACs) [15, 7].

Definition 1 (Triple Graph and Triple Graph Morphism)**.** *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* $t_G : G^C \to G^T$*.* G *is empty, if all components are empty.*

A triple graph morphism $m = (m^S, m^C, m^T) : G \to H$ *between two triple graphs* $G = (G^S \xleftarrow{s_G} G^C \xrightarrow{t_G} G^T)$ *and* $H = (H^S \xleftarrow{s_H} H^C \xrightarrow{t_H} H^T)$ *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* $m^T \circ t_G = t_H \circ m^C$*. It is injective, if morphisms* m^S*,* m^C*, and* m^T *are injective. A typed triple graph* G *is typed over a triple graph* TG *by a triple graph morphism* $type_G : G \to TG$*.*

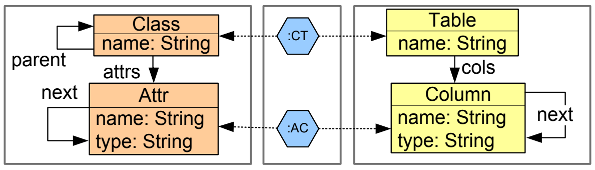


Figure 1: Triple type graph for *CD2RDBM*

Example 1. *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 [15] and [3], 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 dotted arrows. Note that the case study is equipped with attribution, which is based on the concept of E-graphs [5].*

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 and hence, also for the category $\mathbf{AGraphs}_{ATG}$ of attributed graphs.

Triple rules are used to build up source and target graphs as well as their correspondence graphs, i.e. they are non-deleting. Structure filtering which deletes parts of triple graphs, is performed by projection operations only, i.e. structure deletion is not done by rule applications.

Definition 2 (Triple Rule tr and Triple Transformation Step)**.** *A triple rule tr consists of triple graphs L and R, called left-hand and right-hand sides, and 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.*

$$\begin{tikzcd}
L = (L^S \arrow[d, "tr^S"'] & L^C \arrow[l, "s_L"'] \arrow[r, "t_L"] \arrow[d, "tr^C"'] & L^T) \arrow[d, "tr^T"'] \\
R = (R^S & R^C \arrow[l, "s_R"] \arrow[r, "t_R"'] & R^T)
\end{tikzcd}
\qquad
\begin{tikzcd}
L^S \arrow[dr] \arrow[dd, dotted, "m^S"'] & & L^C \arrow[ll] \arrow[rr] \arrow[dr] \arrow[dd, dotted, "m^C"'] & & L^T \arrow[dr] \arrow[dd, "m^T"'] & \\
& R^S \arrow[dd, "n^S"] & & R^C \arrow[ll] \arrow[rr] \arrow[dd, "n^C"] & & R^T \arrow[dd, "n^T"] \\
G = (G^S \arrow[dr, dotted, "t^S"] \arrow[dr, Rightarrow, "tr"'] & & G^C \arrow[ll, dotted] \arrow[rr, dotted] \arrow[dr, dotted, "t^C"] & & G^T) \arrow[dr, "t^T"] & \\
& H = (H^S & & H^C \arrow[ll, "s_H"] \arrow[rr, "t_H"] & & H^T)
\end{tikzcd}$$

*Given a triple rule $tr : L \to R$, a triple graph G and an injective triple graph morphism $m : L \to G$, called triple match m, a triple graph transformation step (*TGT*-step) $G \xRightarrow{tr,m} H$ from G to a triple graph H is given by three pushouts (H^S, t^S, n^S), (H^C, t^C, n^C) and (H^T, t^T, n^T) in category* **Graph** *with induced morphisms $s_H : H^C \to H^S$ and $t_H : H^C \to H^T$. Morphism $n = (n^S, n^C, n^T)$ is called comatch.*

Moreover, we obtain a triple graph morphism $t : G \to H$ with $t = (t^S, t^C, t^T)$ called transformation morphism.

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.

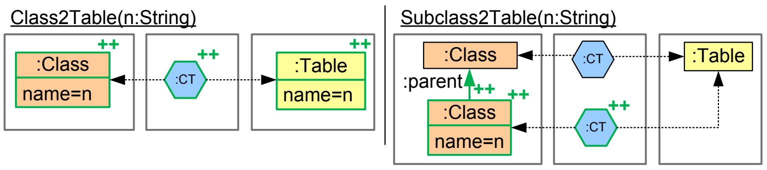


Figure 2: Rules for transforming classes to tables

Example 2 (Triple Rules)**.** *Examples for triple rules are given in Fig. 2 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.*

According to [7] 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 3 (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.

Definition 4 (Derived Triple Rules). *From each triple rule $tr = (L \to R)$ with NACs we derive the following source, target and forward rules:*

$$\begin{tikzcd}
(L^S \arrow[d, "tr^S"'] & \varnothing \arrow[l] \arrow[r] \arrow[d] & \varnothing) \arrow[d] \\
(R^S & \varnothing \arrow[l] \arrow[r] & \varnothing)
\end{tikzcd}$$

source rule tr_S

$$\begin{tikzcd}
(\varnothing \arrow[d] & \varnothing \arrow[l] \arrow[r] \arrow[d] & L_T) \arrow[d, "tr^T"'] \\
(\varnothing & \varnothing \arrow[l] \arrow[r] & R_T)
\end{tikzcd}$$

target rule tr_T

$$\begin{tikzcd}
(R^S \arrow[d, "id"'] & L^C \arrow[l, "tr^S \circ s_L"'] \arrow[r, "t_L"] \arrow[d, "tr^C"'] & L^T) \arrow[d, "tr^T"] \\
(R^S & R^C \arrow[l, "s_R"'] \arrow[r, "t_R"] & R^T)
\end{tikzcd}$$

forward rule tr_F

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. In order to formalize the domain and codomain of correct model transformation sequences we define the sets VL_S of source and VL_T of target models by a restriction of the integrated models to the source and target components, respectively.

Definition 5 (Triple, Source and Target Language). *A set of triple rules TR defines the* triple language *$VL = \{G \mid \varnothing \Rightarrow^* G \text{ via } TR\}$ of triple graphs.* Source language *VL_S and* target language *are derived by projection to the triple components, i.e. $VL_S = proj_S(VL)$ and $VL_T = proj_T(VL)$, where $proj_X$ is a projection defined by restriction to one of the triple components, i.e. $X \in \{S, T\}$.*

Note that a source rule tr_S may be applicable to triple graphs G even if the corresponding triple rule tr is not applicable, because the left hand side of tr_S is smaller in general. For this reason, 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 in Def. 5 and we have $VL_S \subseteq VL_{S0} = \{G_S \mid \varnothing \Rightarrow^* (G_S \leftarrow \varnothing \to \varnothing) \text{ via } TR_S\}$. Analogously, we have $VL_T \subseteq VL_{T0} = \{G_T \mid \varnothing \Rightarrow^* (G_T \leftarrow \varnothing \to \varnothing) \text{ via } TR_T\}$.

Example 3 (Triple Rules with NACs). *Examples for triple rules with NACs and derived rules are given in Fig. 3, where NACs are indicated by red frames with label "NAC". The triple rules specify the synchronous construction of attributes in the source component and their corresponding columns in the target*

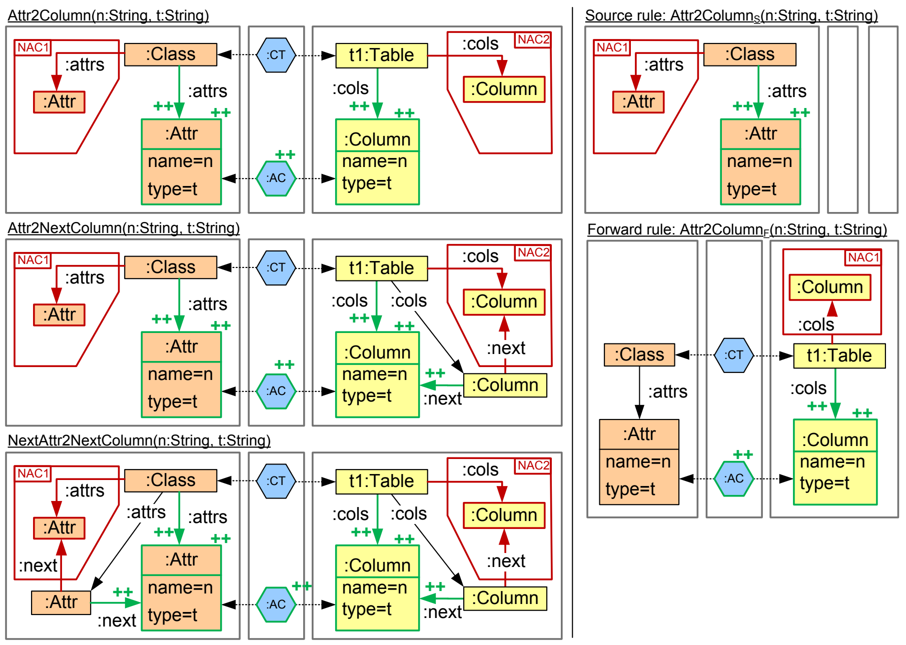

Attr2Column(n:String, t:String)
NAC1
:attrs
:Attr
:Class
:attrs
++
:Attr
name=n
type=t
:CT
:AC
t1:Table
:cols
:Column
name=n
type=t
NAC2
:cols
:Column
Attr2NextColumn(n:String, t:String)
NextAttr2NextColumn(n:String, t:String)
:next
Source rule: Attr2Column$_S$(n:String, t:String)
Forward rule: Attr2Column$_F$(n:String, t:String)

Figure 3: Rules for attributes and columns and derived source and forward rules

component. Attributes and columns build up list structures, which is ensured by the NACs. The first attribute of a class is created by rules "Attr2Column" and "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. 3, where tr_S contains the source NAC (NAC1) and tr_F the target NAC (NAC2) of tr.

Thm. 1 based on [3, 7] shows that TGT-sequences can be decomposed to source and forward sequences and composed out of them. All together this correspondence is bijective. The result uses the following notion of match consistency.

Definition 6 (Match and Source Consistency)**.** *Let tr^*_S and tr^*_F be sequences of source rules $tr_{i,S}$ and forward rules $tr_{i,F}$, which are derived from the same triple rules tr_i for $i = 1, \dots, n$. Let further $G_{00} \xRightarrow{tr^*_S} G_{n0} \xRightarrow{tr^*_F} G_{nn}$ be a* TGT*-sequence with $(m_{i,S}, n_{i,S})$ being match and comatch of $tr_{i,S}$ (respectively $(m_{i,F}, n_{i,F})$ for $tr_{i,F}$) then match consistency of $G_{00} \xRightarrow{tr^*_S} G_{n0} \xRightarrow{tr^*_F} G_{nn}$ means that the S-component of the match $m_{i,F}$ is uniquely determined by the comatch $n_{i,S}$ $(i = 1, \dots, n)$.*

A TGT*-sequence $G_{n0} \xRightarrow{tr^*_F} G_{nn}$ is source consistent, if there is a match consistent sequence $\varnothing \xRightarrow{tr^*_S} G_{n0} \xRightarrow{tr^*_F} G_{nn}$.*

Note that by source consistency the application of the forward rules is controlled by the source sequence, which generates the given source model. Moreover, $\varnothing \xRightarrow{tr^*_S} G_{n0}$ is uniquely determined by source consistency of $G_{n0} \xRightarrow{tr^*_S} G_{nn}$.

Theorem 1 (De-/Composition of TGT-Sequences with NACs)**.**

1. **Decomposition***: For each TGT-sequence*
$$G_0 \xRightarrow{tr_1} G_1 \Rightarrow \ldots \xRightarrow{tr_n} G_n \tag{1}$$
with NACs there is a corresponding match consistent TGT-sequence
$$G_0 = G_{00} \xRightarrow{tr_{1,S}} G_{10} \ldots \xRightarrow{tr_{n,S}} G_{n0} \xRightarrow{tr_{1,F}} G_{n1} \ldots \xRightarrow{tr_{n,F}} G_{nn} = G_n \tag{2}$$
with NACs.

2. **Composition:** *For each match consistent transformation sequence* (2) *with NACs there is a canonical transformation sequence* (1) *with NACs.*

3. **Bijective Correspondence:** *Composition and decomposition are inverse to each other.*

Remark 1 (Injective matches)**.** *According to Def. 2 the matches of the triple rules are required to be injective. If we allow non-injective matches, then we must allow n and q in definition 3 to be non-injective as well.*

Model transformations with NACs from models of the source language VL_{S0} to models of the target language VL_{T0} can be defined on the basis of forward rules as shown in [3, 7]. In this paper we focus our attention to model transformations based on forward rules, where the forward sequence $G_0 \xRightarrow{tr_F^*} G_n$ is required to be source consistent. This means that it is controlled by the corresponding source sequence $\varnothing \xRightarrow{tr_S^*} G_0$.

Definition 7 (Model Transformation based on Forward Rules)**.** *A model transformation sequence $(G_S,\ G_0 \xRightarrow{tr_F^*} G_n, G_T)$ consists of a source graph G_S, a target graph G_T, and a NAC- as well as source consistent forward* TGT*-sequence $G_0 \xRightarrow{tr_F^*} G_n$ 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 $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$ with $G_S \in VL_{S0}$ and $G_T \in VL_{T0}$.

3 On-the-Fly Construction of Model Transformations

In order to construct a model transformation sequence $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$ according to Def. 7 from a given G_S there have been two alternatives up to now [3, 7]: Either we construct a parsing sequence $\varnothing \xRightarrow{tr_S^*} G_0$ first and then try to extend it to a match consistent sequence $\varnothing \xRightarrow{tr_S^*} G_0 \xRightarrow{tr_F^*} G_n$, or we construct directly a forward sequence $G_0 \xRightarrow{tr_F^*} G_n$ 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 extends source consistency in Def. 6 to the case where $G_{00} \xRightarrow{tr_S^*} G_{n0}$ and $G_0 \xRightarrow{tr_F^*} G_n$ are subsequences of the corresponding sequences of a match consistent sequence $\varnothing = G_{00} \xRightarrow{tr_S^*} G_0 \xRightarrow{tr_F^*} G_n$ with inclusion $g_n : G_{n0} \hookrightarrow G_0$.

Partial source consistency of a forward sequence, which is necessary for a complete model transformation, requires that there has to be 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, which is given by inverting the source sequence. This allows us to incrementally extend partially source consistent sequences and we can derive complete source consistent sequences, which ensure that all elements of the source model are translated exactly once.

Definition 8 (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*

$$\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0} \xhookrightarrow{g_n} G_0 \xRightarrow{tr_F^*} G_n$$

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 $g_i = g_n \circ t_{n,S} \dots t_{i-1,S}$.*

$$\begin{tikzcd}
L_{i,S} \arrow[r, hook, "tr_{i,S}"] \arrow[d, "m_{i,S}"'] \arrow[dr, phantom, "(1)"] & R_{i,S} \arrow[rr, hook] \arrow[d, "n_{i,S}"] \arrow[drr, phantom, "(2)"] & & L_{i,F} \arrow[r, hook, "tr_{i,F}"] \arrow[d, "m_{i,F}"'] \arrow[dr, phantom, "(3)"] & R_{i,F} \arrow[d, "n_{i,F}"] \\
G_{i-1,0} \arrow[r, hook, "t_{i,S}"'] & G_{i0} \arrow[r, hook, "g_i"'] & G_0 \arrow[r, hook] & G_{i-1} \arrow[r, hook, "t_{i,F}"'] & G_i
\end{tikzcd}$$

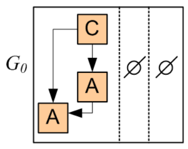
A NAC-consistent forward sequence $G_0 \xRightarrow{tr_F^*} G_n$ is partially source consistent, if there is a source sequence $\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0}$ with inclusion $G_{n0} \xhookrightarrow{g_n} G_0$ such that $G_{00} \xRightarrow{tr_S^*} G_{n0} \xhookrightarrow{g_n} G_0 \xRightarrow{tr_F^*}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

Remark 2.

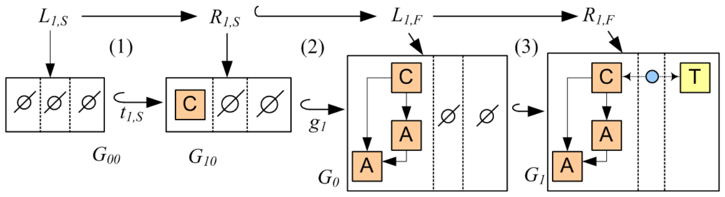
1. *If $g_n = id_{G_0}$, partial match consistency coincides with match consistency.*
2. *For $n = 0$ the partially match consistent sequence is given by $g_0 : G_{00} \hookrightarrow G_0$.*

Remark 3. *Note that we can also consider a more general version of Def. 8, where G_{00} is not required to be empty. In this case Thm. 2 is modified accordingly and the model G_{00} is fixed for all steps of the construction of a partially match consistent sequence. This notion would provide the basis for incremental model transformations. We can assume that G_{00} is an integrated model that contains a source model and its corresponding target model equipped with the correspondence part. G_0 as extension of G_{00} contains further elements in the source component that have to be transformed into target elements in order to propagate the updates from the source model to the target model.*

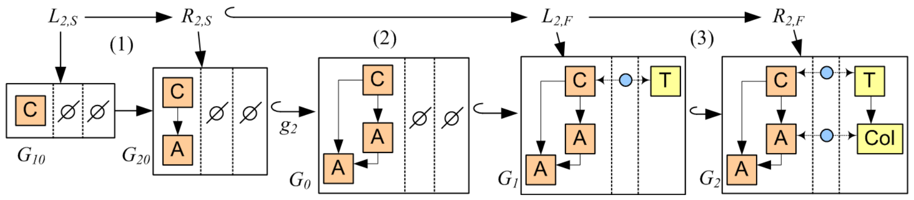
Example 4 (Partial Match and Source Consistency). *Let us consider a candidate sequence starting with triple graph G_0 (depicted to the right) which represents a class diagram consisting of one class with two linked attributes. By triple rules, G_0 is mapped to a corresponding table with two linked columns.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



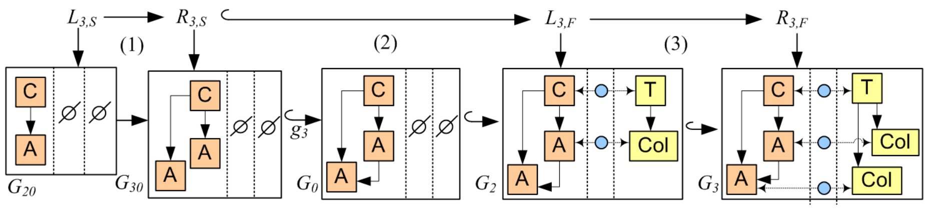
In the first step ($i = 1$), we apply rule $tr_{1,S} = Class2Table_S$ to the empty start graph G_{00} yielding the source graph G_{10} which contains one class. Obviously, G_{10} is included in G_0. Hence, diagram (2) commutes for step 1. The corresponding forward rule $tr_{1,F} = Class2Table_F$ is applied to G_0 and maps the class node to a table node, resulting in G_1.



For step $i = 2$, we apply source rule $tr_{2,S} = Attr2Column_S$ 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, and diagram (2) for step 2 commutes. 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 ($i = 3$), 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.



Since for all considered steps, diagram (2) of Def. 8 commute, we conclude that the 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 a partial match consistent sequence.

The forward sequence $G_0 \xRightarrow{tr_{1,F}} G_1 \xRightarrow{tr_{2,F}} G_2 \xRightarrow{tr_{3,F}} G_3$ *is partially source consistent, because we have the partial match consistent sequence* $\emptyset = G_{00} \xRightarrow{tr_{1,S}} G_{10} \xRightarrow{tr_{2,S}} G_{20} \xRightarrow{tr_{3,S}} G_{30} \xhookrightarrow{g_n} G_0 \xRightarrow{tr_{1,F}} G_1 \xRightarrow{tr_{2,F}} G_2 \xRightarrow{tr_{3,F}} G_3$.

Note that this 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, and 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 9 (Forward Consistent Match)**.** *Given a partially match consistent sequence* $\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n-1,0} \xhookrightarrow{g_n} G_0 \xRightarrow{tr_F^*} G_{n-1}$ *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* (1) *below is a pullback and the matches* $m_{n,F}$ *and* $m_{n,S}$ *satisfy the corresponding target and source NACs, respectively.*

$$\begin{tikzcd}
L_{n,S} \arrow[r, hook] \arrow[d, "m_{n,S}"'] & R_{n,S} \arrow[r, hook] \arrow[d, phantom, "(1)"] & L_{n,F} \arrow[d, "m_{n,F}"] \\
G_{n-1,0} \arrow[r, hook, "g_{n-1}"'] & G_0 \arrow[r, hook] & G_{n-1}
\end{tikzcd}$$

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*

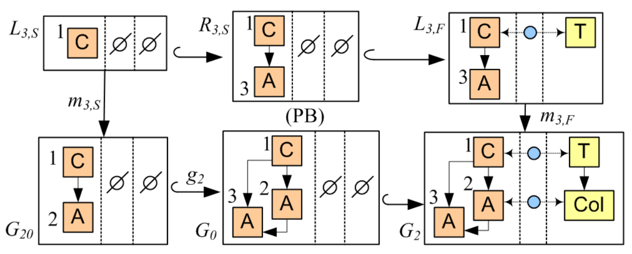
$$(2) : m_{n,F}(L_{n,F}) \cap G_{n-1,0} = m_{n,F}(L_{n,F}).$$

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, depicted below, 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. 9 is a pullback. Analogously, the matches from forward rules in steps 1 and 2 are also forward consistent.*



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 2 (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 $s = \langle\, G_0 \xRightarrow{tr_F^*} G_{n-1} \,\rangle$ with $\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n-1,0}$ 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 $G_0 \xRightarrow{tr_F^*} G_{n-1} \xRightarrow{tr_{n,F}} G_n$ with $G_{00} \xRightarrow{tr_S^*} G_{n-1,0} \xRightarrow{tr_{n,S}} G_{n0}$ 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 $G_0 \xRightarrow{tr_F^*} G_n$ is source consistent leading to a model transformation sequence $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$ with G_S and G_T being the source and target models of G_0 and G_n.*

Proof. We have to show that this procedure is well-defined, i.e. that in Step 2, a forward consistent match leads to an extended partially source consistent sequence $G_0 \xRightarrow{tr_F^*} G_n$.

Given the situation as in Step 2 above, $(1) + (2)$ is a pullback because $m_{n,F}$ is forward consistent. The construction of pushout (1) leads to the source transformation $G_{n-1,0} \xRightarrow{tr_{n,S}} G_{n0}$, embedding $g_n : G_{n0} \hookrightarrow G_0$ and $g_n^S \circ n_{n,S}^S = m_{n,F}^S$ due to $R_{n,S}^S = L_{n,F}^S$, $G_0^S = G_{n-1}^S$ and the S-components of (1) and $(1) + (2)$ being a a pushout and a pullback over monomorphisms, respectively, such that the induced morphism $g_n^S : G_{n0}^S \to G_0^S$ is a monomorphism and w.l.o.g. an inclusion. Hence $g_n : G_{n0} \hookrightarrow G_0$ is an inclusion.

Moreover, $G_{n-1,0} \xRightarrow{tr_{n,S}} G_{n0}$ and $G_{n-1} \xRightarrow{tr_{n,F}} G_n$ are NAC-consistent by assumption. Thus, $G_0 \xRightarrow{tr_F^*} G_n$ is partially source consistent.

$$\begin{tikzcd}
L_{n,S} \arrow[r, hook, "tr_{n,S}"] \arrow[d, "m_{n,S}"'] \arrow[dr, phantom, "(1)"] & R_{n,S} \arrow[rr, hook] \arrow[d, "n_{n,S}"] \arrow[dr, phantom, "(2)"] & & L_{n,F} \arrow[d, "m_{n,F}"] \\
G_{n-1,0} \arrow[r, hook, "t_{n,S}"] \arrow[rr, hook, bend right, "g_{n-1}"'] & G_{n0} \arrow[r, hook, "g_n"] & G_0 \arrow[r, hook] & G_{n-1}
\end{tikzcd}$$

$\square$

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.

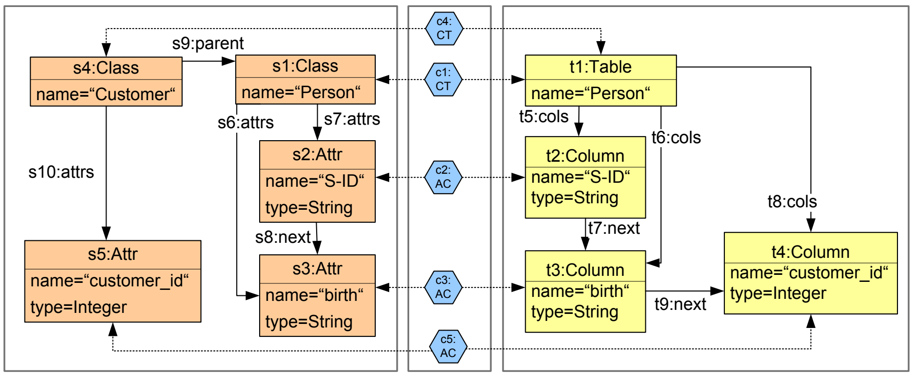


Figure 4: G_5 of Forward Sequence

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 $tr_{4,F}$. Hence, we cannot extend our sequence found so far to a source consistent sequence. The reason is that there exists no triple rule for inserting a* next *link between two already inserted attributes. The mistake we made was to use the wrong rule $Attr2Column_S$ for the insertion of the second attribute. If we had used rule $NextAttr2NextColumn_S$ 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, one has two choices: either, we have to try to apply a different rule in the previous step (and maybe have to go back even further), or we restrict the applicability of our triple rules, e.g. by adding negative application conditions. In the second case, when considering also the NACs in Fig. 3, we will always construct a source consistent sequence, because only one attribute-adding rule would be applicable in each step. An example for a source-consistent sequence, constructed by partially source consistent sequences according to Thm. 2, is the model transformation ($G_S = G_{0,S}, G_0 \xRightarrow{tr_F^*} G_5, G_T = G_{5,T}$), where G_5 (shown in Fig. 4) is generated by the forward sequence $G_0 \xRightarrow{Class2Table} G_1 \xRightarrow{Attr2Col} G_2 \xRightarrow{Subclass2Table} G_3 \xRightarrow{NextAttr2NextCol} G_4 \xRightarrow{Attr2NextCol} G_5$, and G_0 is generated by the corresponding source sequence $\varnothing \xRightarrow{tr_S^*} G_0$. All ele-*

Table 1: Steps of Source Consistent Model Transformation

	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

ments in Fig. 4 are labeled with numbers. Table 1 specifies the matches and the created objects for each transformation step.

4 Analysis of the Construction and Improvement of Efficiency

In this section, we analyze the on-the-fly construction in Thm. 2 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 3 (Correctness and Completeness).

- Correctness: *If the on-the-fly construction terminates with $g_n = id_{G_0}$, then the resulting model transformation $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$ 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 $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$, which can be obtained by the on-the-fly construction.*

Remark 4. *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.*

Proof.

- Correctness: If the procedure terminates with $g_n = id_{G_0}$ and a source consistent forward transformation $G_0 \xRightarrow{tr_F^*} G_n$ with a corresponding source transformation $\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0} = G_0$ then $\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0} = G_0 \xRightarrow{tr_F^*} G_n$ is match consistent and by Thm. 1 there is a TGT-sequence $\varnothing = G_{00} \xRightarrow{tr^*} G_n$ with $G_0^S = G_n^S = G_S$ and $G_n^T = G_T$ and by Def. 5 $G_S \in VL_S$ and $G_T \in VL_T$.
- Completeness: $G_S \in VL_S$ implies that there is a TGT-transformation $\varnothing = G_{00} \xRightarrow{tr^*} G_n$ with $G_n^S = G_S$ and $tr^* = (tr_i)_{i=1...n}$, which can be

decomposed by Thm. 1 into a match consistent sequence $G_{00} \xRightarrow{tr_S^*} G_{n0} = G_0 \xRightarrow{tr_F^*} G_n$ with matches $m_{i,S}$ and $m_{i,F}$.

The on-the-fly construction starts with $\varnothing = G_{00}$ and $g_0 : G_{00} \hookrightarrow G_0$. In Step 2, for $i = 1, \ldots n$ we have a partially match consistent sequence $\varnothing = G_{00} \xRightarrow{tr_S^{1\ldots i}} G_{i0} \xhookrightarrow{g_i} G_0 \xRightarrow{tr_F^{1\ldots i}} G_i$. Choose $tr_{i+1,F}$ as the next rule in the forward sequence with match $m_{i+1,F}$. For the source match $m_{i+1,S}$, (1) is a pushout and since the original sequence is source consistent $m_{i+1,F}$ is uniquely determined by $n_{i+1,S}$, which means that there is an inclusion $g_{i+1}^S : G_{i+1,0}^S \hookrightarrow G_0^S = G_i^S$ such that $m_{i+1,F}^S = g_{i+1}^S \circ n_{i+1}^S$ and $g_i^S = g_{i+1}^S \circ t_{i+1,S}^S$. With (1) being both pushout and pullback, and g_{i+1} and $G_0 \hookrightarrow G_i$ being monomorphisms we have that $(1) + (2)$ is a pullback, leading to the fact that $m_{i+1,F}$ is forward consistent. This procedure terminates after n steps with $g_n = id_{G_0}$ leading to the target model $G_T = G_n^T$.

$$\begin{tikzcd}
L_{i+1,S} \arrow[r, hook, "tr_{i+1,S}"] \arrow[d, "m_{i+1,S}"'] \arrow[dr, phantom, "(1)"] & R_{i+1,S} \arrow[rr, hook] \arrow[d, "n_{i+1,S}"] & \arrow[d, phantom, "(2)"] & L_{i+1,F} \arrow[d, "m_{i+1,F}"] \\
G_{i0} \arrow[r, hook, "t_{i+1,S}"'] & G_{i+1,0} \arrow[r, hook, "g_{i+1}"] & G_0 \arrow[r, hook] & G_i
\end{tikzcd}$$

□

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 4 (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. $R_{i,S} \setminus L_{i,S} \neq \varnothing$.*

Proof. In the case of creating source rules, the sequence of inclusions $G_{00}^S \xhookrightarrow{t_{1,S}^S} G_{10}^S \xhookrightarrow{t_{2,S}^S} G_{20}^S \ldots$ is strictly increasing, which means that we have, after a finite number of steps, that either $G_{n0} = G_0$ and the procedure terminates, or there are no more forward rules with forward consistent matches and the procedure aborts. □

Example 7 (Termination)**.** *The on-the-fly construction of triple graph G_5 in Ex. 6 terminates because all used source rules in the source sequence are creating, as can be easily seen in Table 1 in the left column* Source Sequence Elements*.*

Confluence. For functional behaviour of the model transformations, also confluence should be considered, which insures that the results of different transformations of a source graph lead to the same target graph. In general, it would be interesting to find sufficient conditions for local confluence and confluence.

Example 8. *Note that the triple rules from our running example are not confluent. We have source-consistent sequences for the same G_0 which lead to different corresponding tables. If a class has more than one attribute, the order of columns in the resulting table depends on the order in which we transform the attributes. So, the columns of the table could be linked to each other in any order and would always be a valid result of a source-consistent forward transformation.*

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 10 (Parallel Independence of Partially Match Consistent Extensions)**.** *Two partially match consistent sequences*
$\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0} \xRightarrow{tr_{1,S}} G_{n+1,0} \xhookrightarrow{g_{n+1}} G_0 \xRightarrow{tr_F^*} G_n \xRightarrow{tr_{1,F}} G_{n+1}$ *and*
$\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0} \xRightarrow{tr_{2,S}} G'_{n+1,0} \xhookrightarrow{g'_{n+1}} G_0 \xRightarrow{tr_F^*} G_n \xRightarrow{tr_{2,F}} G'_{n+1}$
are parallel independent *if* $G_{n0} \xRightarrow{tr_{1,S}} G_{n+1,0}$ *and* $G_{n0} \xRightarrow{tr_{2,S}} G'_{n+1,0}$ *as well as* $G_n \xRightarrow{tr_{1,F}} G_{n+1}$ *and* $G_n \xRightarrow{tr_{2,F}} G'_{n+1}$ *are parallel independent leading to the diagram* (1_S) *and* (1_F)*, and diagram* (2) *is a pullback.*

$$\begin{tikzcd}
G_{n0} \arrow[r, Rightarrow, "tr_{1,S}"] \arrow[d, Rightarrow, "tr_{2,S}"'] \arrow[dr, phantom, "(1_S)"] & G_{n+1,0} \arrow[d, Rightarrow, "tr_{2,S}"] & G_n \arrow[r, Rightarrow, "tr_{1,F}"] \arrow[d, Rightarrow, "tr_{2,F}"'] \arrow[dr, phantom, "(1_F)"] & G_{n+1} \arrow[d, Rightarrow, "tr_{2,F}"] & G_{n0} \arrow[r, hook, "t_{1,S}"] \arrow[d, hook, "t_{2,S}"'] \arrow[dr, phantom, "(2)"] & G_{n+1,0} \arrow[d, hook, "g_{n+1}"] \\
G'_{n+1,0} \arrow[r, Rightarrow, "tr_{1,S}"'] & G_{n+2,0} & G'_{n+1} \arrow[r, Rightarrow, "tr_{1,F}"'] & G_{n+2} & G'_{n+1,0} \arrow[r, hook, "g'_{n+1}"'] & G_0
\end{tikzcd}$$

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

$$\begin{tikzcd}
 & & G_{n+1,0} \arrow[dr, Rightarrow, "tr_{2,S}"] & & & & G_{n+1} \arrow[dr, Rightarrow, "tr_{2,F}"] & \\
\varnothing = G_{00} \arrow[r, Rightarrow, "tr_S^*"] & G_{n0} \arrow[ur, Rightarrow, "tr_{1,S}"] \arrow[dr, Rightarrow, "tr_{2,S}"'] & & G_{n+2,0} \arrow[r, hook] & G_0 \arrow[r, Rightarrow, "tr_F^*"] & G_n \arrow[ur, Rightarrow, "tr_{1,F}"] \arrow[dr, Rightarrow, "tr_{2,F}"'] & & G_{n+2} \\
 & & G'_{n+1,0} \arrow[ur, Rightarrow, "tr_{1,S}"'] & & & & G'_{n+1} \arrow[ur, Rightarrow, "tr_{1,F}"'] &
\end{tikzcd}$$

Proof. We show the partial match consistency of the sequence $\varnothing = G_{00} \xRightarrow{tr_S^*} G_{n0} \xRightarrow{tr_{1,S}} G_{n+1,0} \xRightarrow{tr_{2,S}} G_{n+2,0} \xhookrightarrow{g_{n+2}} G_0 \xRightarrow{tr_F^*} G_n \xRightarrow{tr_{1,F}} G_{n+1} \xRightarrow{tr_{2,F}} G_{n+2}$, the other one follows dually. It suffices to show that the match $m_{2,F}$ in pushout (3_F) is forward consistent, which means that (4) is a pullback.

$$\begin{tikzcd}
L_{2,S} \arrow[r, "tr_{2,S}"] \arrow[d, "m_{2,S}"'] \arrow[dr, phantom, "(3_S)"] & R_{2,S} \arrow[d, "n_{2,S}"] & L_{2,F} \arrow[r, "tr_{2,F}"] \arrow[d, "m_{2,F}"'] \arrow[dr, phantom, "(3_F)"] & R_{2,F} \arrow[d, "n_{2,F}"] & L_{2,S} \arrow[r, "tr_{2,S}"] \arrow[d, "m_{2,S}"] & R_{2,S} \arrow[r, hook] \arrow[d, phantom, "(4)"] & L_{2,F} \arrow[d, "m_{2,F}"'] \\
G_{n+1,0} \arrow[r, hook, "t_{2,S}"'] & G_{n+2,0} & G_{n+1} \arrow[r, hook, "t_{2,F}"'] & G_{n+2} & G_{n+1,0} \arrow[r, hook, "g_{n+1}"'] & G_0 \arrow[r, hook] & G_{n+1}
\end{tikzcd}$$

By parallel independence we have the following pushouts from (1_S) and (1_F) with $(3_S) = (6_S) + (7_S)$ and $(3_F) = (6_F) + (7_F)$.

$$\begin{tikzcd}
 & L_{1,S} \arrow[r, "tr_{1,S}"] \arrow[d, "m_{1,S}"'] \arrow[dr, phantom, "(5_S)"] & R_{1,S} \arrow[d, "n_{1,S}"] \\
L_{2,S} \arrow[r, "m'_{2,S}"] \arrow[d, "tr_{2,S}"'] \arrow[dr, phantom, "(6_S)"] & G_{n0} \arrow[r, "t_{1,S}"] \arrow[d, "t'_{2,S}"'] \arrow[dr, phantom, "(7_S)"] & G_{n+1,0} \arrow[d] \\
R_{2,S} \arrow[r, "n'_{2,S}"'] & G'_{n+1,0} \arrow[r] & G_{n+2,0}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & L_{1,F} \arrow[r, "tr_{1,F}"] \arrow[d, "m_{1,F}"'] \arrow[dr, phantom, "(5_F)"] & R_{1,F} \arrow[d, "n_{1,F}"] \\
L_{2,F} \arrow[r, "m'_{2,F}"] \arrow[d, "tr_{2,F}"'] \arrow[dr, phantom, "(6_F)"] & G_{n} \arrow[r, "t_{1,F}"] \arrow[d, "t'_{2,F}"'] \arrow[dr, phantom, "(7_F)"] & G_{n+1} \arrow[d] \\
R_{2,F} \arrow[r, "n'_{2,F}"'] & G'_{n+1} \arrow[r] & G_{n+2}
\end{tikzcd}$$

This implies that $m_{2,S} = t_{1,S} \circ m'_{2,S}$ and $m^S_{2F} = t^S_{1F} \circ m'^S_{2F} = m'^S_{2F}$ because $t^S_{1,F} = id$. Moreover, match consistency of the second sequence implies that $m^S_{2,F} = g'^S_{n+1} \circ n'^S_{2,S}$. (6_S) is a pushout and also a pullback, and thus the square $(6_S) + (2)$ as a composition of pullbacks is also a pullback, and hence also (4) is a pullback, because $m_{2,S} = t_{1,S} \circ m'_{2,S}$ and the S-component of (8) is a pullback with horizontal identities and $m^S_{2,F} = g'^S_{n+1} \circ n'^S_{2,S}$.

$$\begin{tikzcd}
L_{2,S} \arrow[r, "tr_{2,S}"] \arrow[d, "m'_{2,S}"'] \arrow[dr, phantom, "(6_S)"] & R_{2,S} \arrow[r, hook] \arrow[d, "n'_{2,S}"] & L_{2,F} \arrow[dd, "m_{2,F}"] \\
G_{n,0} \arrow[r, "t'_{2,S}"] \arrow[d, "t_{1,S}"'] \arrow[dr, phantom, "(2)"] & G'_{n+1,0} \arrow[d, "g'_{n+1}"] & \\
G_{n+1,0} \arrow[r, "g_{n+1}"'] & G_0 \arrow[r, hook] & G_{n+1}
\end{tikzcd}$$

□

Example 9 (Parallel Independence). *Consider the sequence of rule applications in Table 1. Here, we may switch step 2 and step 3 without changing the result* G_5 *since the sequences* $\varnothing = G_{00} \xRightarrow{Class2Table_S} G_{10} \xRightarrow{Attribute2Column_S} G_{2,0} \xhookrightarrow{g_2} G_0 \xRightarrow{Class2Table_F} G_1 \xRightarrow{Attribute2Column_F} G_2$ *and* $\varnothing = G_{00} \xRightarrow{Class2Table_S} G'_{10} \xRightarrow{Subclass2Table_S} G'_{2,0} \xhookrightarrow{g'_2} G_0 \xRightarrow{Class2Table_F} G_1 \xRightarrow{Subclass2Table_F} G'_2$ *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 [9], 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 [15, 13, 10], and various kinds of applications have been presented [16, 11, 12]. For an extensive overview see [15]. A new extension of TGGs towards declarative, pattern-based model transformation is presented in [2], where triple rules are derived from triple graph constraints.

Furthermore, Kindler and Wagner [12] 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 [15], 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. 9).

As stated in Sec. 1 this paper extends especially various concepts and results of our previous papers [3, 8, 6, 7]. 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 [15]:

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. 3 for an algorithm based on our construction in Thm. 2.

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. 3 and Thm. 4 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 [15] only meets this condition because it avoids backtracking by aborting a translation in the case that 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 their approach. Note further that by Thm. 5 we are able to perform partial order reduction, which has shown to provide massive power for the reduction of complexity (see e.g. [9]).

4. *Expressiveness:* Finally, 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 [14, 3, 7] 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 [1].

Finally, we have 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. But 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 [5] including tool support by our tool AGG [17]. Furthermore, correctness criteria independent from the TGG as well as an extension to general application conditions shall be developed.

References

[1] C. Brandt, F. Hermann, and T. Engel. Security and Consistency of IT and Business Models at Credit Suisse realized by Graph Constraints, Transformation and Integration using Algebraic Graph Theory. In *Proc. Int. Conf. on Exploring Modeling Methods in Systems Analysis and Design 2009 (EMMSAD'09)*, volume 29 of *LNBIP*, pages 339–352. Springer, 2009.

[2] J. de Lara and E. Guerra. Pattern-based model-to-model transformation. In H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, editors, *Proc. of ICGT'08*, volume 5214 of *Lecture Notes in Computer Science*, pages 426–441. Springer, 2008.

[3] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, and G. Taentzer. Information Preserving Bidirectional Model Transformations. In M. Dwyer and A. Lopes, editors, *Proc. of FASE'07*, volume 4422 of *LNCS*, pages 72–86. Springer, 2007.

[4] H. Ehrig, K. Ehrig, and F. Hermann. From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. In *Proc. of GT-VMT'08. EC-EASST*, 10:1–14, 2008.

[5] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation.* EATCS Monographs. Springer, 2006.

[6] H. Ehrig, C. Ermel, and F. Hermann. On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars. In G. Karsai and G. Taentzer, editors, *Proc. of GraMoT'08.* ACM, 2008.

[7] H. Ehrig, F. Hermann, and C. Sartorius. Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions. In *Proc. of GT-VMT'09. EC-EASST*, 18, 2009. to appear.

[8] H. Ehrig and U. Prange. Formal Analysis of Model Transformations Based on Triple Graph Rules with Kernels. In H. Ehrig, R. Heckel, G. Rozenberg,

and G. Taentzer, editors, *Proc. ICGT'08*, volume 5214 of *LNCS*, pages 178–193. Springer, 2008.

[9] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems – An Approach to the State-Explosion Problem*, volume 1032 of *LNCS*. Springer, 1996.

[10] E. Guerra and J. de Lara. Attributed Typed Triple Graph Transformation with Inheritance in the Double Pushout Approach. Technical Report UC3M-TR-CS-2006-00, Universidad Carlos III, Madrid, 2006.

[11] E. Guerra and J. de Lara. Model View Management with Triple Graph Grammars. In A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, editors, *Proc. of ICGT'06*, volume 4178 of *LNCS*, pages 351–366. Springer, 2006.

[12] E. Kindler and R. Wagner. Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios. Technical Report TR-ri-07-284, Universität Paderborn, 2007.

[13] A. Königs and A. Schürr. Tool Integration with Triple Graph Grammars - A Survey. *ENTCS*, 148:113–150, 2006.

[14] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In G. Tinhofer, editor, *Proc. of WG'94*, volume 903 of *LNCS*, pages 151–163. Springer, 1994.

[15] A. Schürr and F. Klar. 15 Years of Triple Graph Grammars. In H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, editors, *Proc. of ICGT'08*, volume 5214 of *LNCS*, pages 411–425. Springer, 2008.

[16] G. Taentzer, K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovsky, U. Prange, D. Varro, and S. Varro-Gyapay. Model Transformation by Graph Transformation: A Comparative Study. In *Proc. WMTP'05*, 2005.

[17] TFS-group, TU Berlin. *AGG*, 2009. `http://tfs.cs.tu-berlin.de/agg`.