Electronic Communications of the EASST
Volume 30 (2010)

International Colloquium on Graph and Model Transformation On the occasion of the 65th birthday of Hartmut Ehrig
(GraMoT 2010)

Specification and Verification of Model Transformations

Frank Hermann, Mathias Hülsbusch, Barbara König

20 pages

Guest Editors: Claudia Ermel, Hartmut Ehrig, Fernando Orejas, Gabriele Taentzer
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Specification and Verification of Model Transformations [*]

Frank Hermann[1], **Mathias Hülsbusch**[2], **Barbara König**[2]

[1] Institut für Softwaretechnik und Theoretische Informatik,
Technische Universität Berlin, Germany,
`frank(at)cs.tu-berlin.de`

[2] Abteilung für Informatik und Angewandte Kognitionswissenschaft,
Universität Duisburg-Essen, Germany,
`{mathias.huelsbusch,barbara_koenig}(at)uni-due.de`

Abstract: Model transformations are a key concept within model driven development and there is an enormous need for suitable formal analysis techniques for model transformations, in particular with respect to behavioural equivalence of source models and their corresponding target models.

For this reason, we discuss the general challenges that arise for the specification and verification of model transformations and present suitable formal techniques that are based on graph transformation. In this context, triple graph grammars show many benefits for the specification process, e.g. modelers can work on an intuitive level of abstraction and there are formal results for syntactical correctness, completeness and efficient execution. In order to verify model transformations with respect to behavioural equivalence we apply well-studied techniques based on the double pushout approach with borrowed context, for which the model transformations specified by triple graph transformation rules are flattened to plain (in-situ) graph transformation rules.

The potential and adequateness of the presented techniques are demonstrated by an intuitive example, for which we show the correctness of the model transformation with respect to bisimilarity of source and target models.

Keywords: Model transformation, behavioural equivalence, verification

1 Introduction

In the setting of model driven architecture (MDA), a system is implemented by first specifying an abstract model, which is subsequently refined to executable code. This is done by model transformations, which transform a source model into a more concrete target model. The Object Mangaement Group has also introduced a standard for model transformations: QVT (Query/View/Transformation). A special case is refactoring, where only the internal structure of the model or system is changed and potentially optimized. Since refactorings are expected not to modify the functional behaviour of the system, the notion of behaviour preservation is crucial: how can we specify and verify that a model keeps its original behaviour after several refactoring

[*] Research supported by DFG project Behaviour-GT.

steps? The same question is often relevant for model transformations, in order to show that the implementation matches the original specification.

In this paper we will summarize some results on the specification and verification of model transformations. As underlying modelling framework we will use graph transformation, which is well-suited to handle the graph-like structures usually arising in MDA and UML. However, many existing model transformations in practice are directly encoded as e.g. XSLT-transformations. As a first contribution of this paper we will discuss the main challenges of model transformations and present the main benefits of using graph transformation with respect to technical results and with respect to usability.

The main two aims of the paper are the following. First, we introduce recent results on triple graph grammars, a formalism that allows to specify model transformations by constructing source and target models simultaneously and recording correspondences. Second, we will describe a technique for verifying that model transformations translate source models into bisimilar target models, using a variation of the borrowed context technique. For this, we will derive in-situ transformation rules from triple graph grammars. Both parts of the paper are based on the same example: a model transformation translating network-like models with different types of (bidirectional and unidirectional) links.

The structure of the paper is as follows. Section 2 describes the various challenges arising in the area of model transformation. Sections 3 and 4 subsequently introduce triple graph grammars for the specification of model transformations and describe several results obtained for triple graph grammars (e.g., syntactical correctness and completeness). In Section 5 we describe how to verify the example transformation using the borrowed context technique. Finally, we will compare with related work in Section 6 and conclude (Section 7).

2 Challenges for Model Transformations

Model transformations appear in several contexts, e.g. in the various facets of model driven architecture encompassing model refinement and interoperability of system components. The involved languages can be closely related or they can be more heterogeneous, e.g. in the special case of model refactoring the source language and the target language are the same. From a general point of view, a model transformation $MT : VL_S \Rrightarrow VL_T$ between visual languages transforms models from the source language VL_S to models of the target language VL_T. Main challenges were described in [SK08] for model transformation approaches based on triple graph grammars. Here, we extend this list and also the scope and describe general challenges for model transformations.

There are two dimensions, which contain major challenges for model transformations being on the one hand functional aspects and on the other hand non-functional aspects. The first dimension of functional aspects concerns the reliability of the produced results. Depending on the concrete application of a model transformation $MT : VL_S \Rrightarrow VL_T$, the following properties may have to be ensured.

1. *Syntactical Correctness:* For each model $M_S \in VL_S$ that is transformed by MT the resulting model M_T has to be syntactically correct, i.e. $M_T \in VL_T$.

2. *Semantical Correctness:* The semantics of each model $M_S \in VL_S$ that is transformed by MT has to be preserved or reflected, respectively.

3. *Completeness:* The model transformation MT can be performed on each model $M_S \in VL_S$. Additionally, it can be required that MT reaches all models $M_T \in VL_T$.

4. *Functional Behaviour:* For each source model M_S the model transformation MT will always terminate and lead to the same resulting target model M_T.

The second dimension of non-functional aspects of model transformations concerns usability and applicability. Therefore, from the application point of view some of the following challenges are also main requirements.

1. *Efficiency:* Model transformations should have polynomial space and time complexity. Furthermore, there may be further time constraints that need to be respected, depending on the application domain and the intended way of use.

2. *Intuitive Specification:* The specification of model transformations can be performed based on patterns that describe how model fragments in a source model correspond to model fragments in a target model. If the source (resp. target) language is a visual language then the components of the model transformation can be visualized using the concrete syntax of the visual language.

3. *Maintainability:* Extensions and modifications of a model transformation should be easy. Side effects of local changes should be handled and analyzed automatically.

4. *Expressiveness:* Specifications of model transformations should be expressive enough. For instance, special control conditions have to be available in order to handle more complex models, which, e.g., contain substructures with a partial ordering or hierarchies.

5. *Bidirectional model transformations:* The specification of a model transformation should provide the basis for both, a model transformation from the source to the target language and a model transformation in the inverse direction.

In the following section we present suitable techniques for the specification of model transformations based on graph transformation. These techniques provide validated and verified capabilities for a wide range of the challenges listed above.

3 Specification of Model Transformations by Triple Graph Grammars

A promising and well studied approach for the specification of model transformations is based on triple graph transformation [Sch94]. This section presents its main concepts and Sec. 4 shows its advantages from the formal and from the application point of view. The most important advantage of triple graph transformation is the combination of both, its intuitive way of specifying model transformations and its formal basis, for which correctness and completeness results are available.

Triple graphs combine three graphs - one for the source model, one for the target model and one in between, together with connecting graph morphisms for the specification of the correspondences between the elements in the source and the target model. This extension of plain graphs improves the definition of model transformations. Source models are parsed and their corresponding target models are completed without the need of deleting the source model in between. The correspondences between both models are used to guide the transformation process.

Definition 1 (Category **TripleGraphs**) Three graphs G^S, G^C, and G^T, called source, connection, and target graph, together with two graph morphisms $s_G : G^C \to G^S$ and $t_G : G^C \to G^T$ form a triple graph $G = (G^S \xleftarrow{s_G} G^C \xrightarrow{t_G} G^T)$. G is called *empty*, if G^S, G^C, and G^T are empty graphs.

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. Triple graphs and triple graph morphisms form the category **TripleGraphs**. Given a triple graph TG, called type graph, the category **TripleGraphs**$_{TG}$ of typed triple graphs is given by the slice category **TripleGraphs**$\backslash TG$.

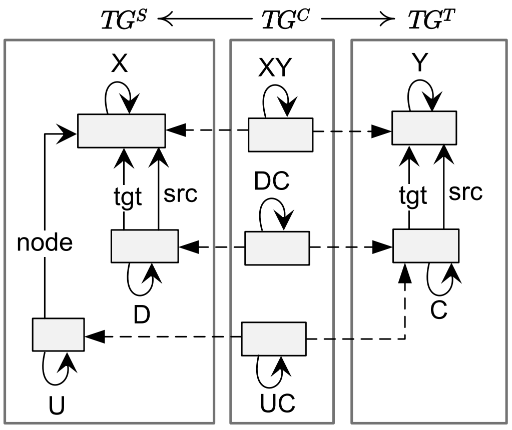

Figure 1: Triple Type Graph $TG = (TG^S \leftarrow TG^C \rightarrow TG^T)$

In the examples of this paper we consider a model transformation MT : BidiDiLang $\Rrightarrow$ UniDiLang between communication structure models. The language BidiDiLang contains models with bidirectional and unidirectional links and these models are transformed to models with unidirectional connections only in the language UniDiLang. Each pair of corresponding source and target models is given by a triple graph typed over the triple type graph TG in Fig. 1, which extensively uses the concept of labeled nodes, i.e. loops of different edge types. This allows us in Sec. 5.2 to merge different node types in a compact type graph in order to verify the semantical correctness of the model transformation. For this purpose, we transform the triple rules into suitable in-situ rules and analyze them with respect to the rules of the mixed semantics, i.e. a semantics for models that simultaneously contain source and target elements.

In order to improve the intuition of triple graphs typed over *TG* we present the graphs visually. The source, correspondence and target components of the triple graphs are separated by rectangles. The fill colours additionally improve this separation. Elements in the source model are light red while they are blue in the correspondence and yellow in the target model. Furthermore, the node types “X”, “D” (directed link), “U” (undirected link), “Y”, “C” (connection) as well as “XY”, “DC” and “UC” for the correspondence component will internally be represented by loops at the different nodes and we simplify the presentation by putting the label inside the node rectangle resp. hexagon.

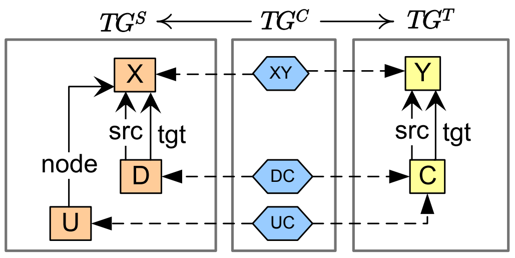

Figure 2: Visualization of the Triple Type Graph *TG*

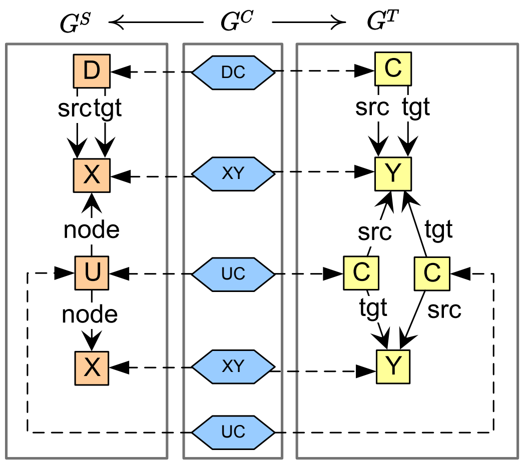

Figure 3: Triple Graph G with source model G^S and target model G^T

Example 1 (Triple graph) *The triple graph in Fig. 3 is typed over TG and shows an integrated model consisting of a source model G^S (left) and a target model G^T (right), which are connected via the correspondence nodes in the correspondence graph G^C. The source model specifies a node with label “X” having a message “m”, a self-referring directed link “D” and an outgoing undirected link “U”. Similarly the target model contains two nodes, but labelled with “Y”, and instead of one undirected link between both nodes there are two connections “C” defining possibilities for communication in both directions. The corresponding elements of both models are* *related by graph morphisms (indicated by dashed lines) from the correspondence graph (light blue) to the source and target componencts, respectively.*

A triple graph grammar generates a language of triple graphs, i.e. a language of integrated models consisting of models of the source and the target language and a correspondence structure in between. The triple rules of a triple graph grammar specify the synchronous creation of elements in the source component and its corresponding elements in the target component. Therefore, triple rules are non-deleting. The triple rules of a triple graph grammar are the basis for deriving the operational rules of the model transformation from models of one language into the other.

Definition 2 (Triple Graph Transformation and Triple Graph Grammar) A triple rule $tr = L \xrightarrow{tr} R$ is an injective triple graph morphisms tr from a triple graph L (left hand side) to a triple graph R (right hand side). A triple graph grammar $TGG = (TG, S, TR)$ consists of a triple graph TG (type graph), a triple graph S (start graph) and triple rules TR - both typed over TG.

Given a triple rule $tr = (tr^S, tr^C, tr^T) : L \to R$, a triple graph G and an injective triple graph morphism $m = (m^S, m^C, m^T) : L \to G$, 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 a pushout in **TripleGraphs**. The triple graph language L of TGG is defined by $L = \{G \mid \exists$ triple graph transformation $S \Rightarrow^* G\}$.

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

Triple Rule

$$\begin{tikzcd}
L \arrow[r, "tr"] \arrow[d, "m"'] \arrow[dr, phantom, "(PO)"] & R \arrow[d, "n"] \\
G \arrow[r, "t"'] & H
\end{tikzcd}$$

Transformation Step

Model transformations based on triple graph transformation are performed by taking the source model and extending it to an integrated model, where all its corresponding elements in the correspondence and target component are completed. Thereafter, this integrated model is restricted to its target component, which is the result of the model transformation. For this reason, triple graph transformation rules are non-deleting. This implies that the first step in the DPO graph transformation approach [EEPT06] can be omitted, because the creation of elements is performed in the second step.

Example 2 (Triple Graph Grammar) *The triple graph grammar* $TGG = (TG, \emptyset, TR)$ *for the model transformation* MT : BidiDiLang $\Rrightarrow$ UniDiLang *contains the triple type graph in Fig. 2, the empty start graph and the rules TR in Fig. 4. Each rule specifies a pattern that describes how particular fragments of the communication structure models shall be related. We present the rules in compact notation, i.e. the left and the right hand side of a rule are shown in one triple graph and the additional elements that occur in the right hand side only are marked by green line colour and a double plus sign.*

The rule "nodeX2nodeY" synchronously creates an "X" node in the source model and its corresponding "Y" node in the target model. Thus, in this case the left hand side of this rule is the empty triple graph, because all elements are created. The rule "directed2connection" creates directed links "D" between two "X" nodes in the source component and their corresponding *connection "C" between the related "Y" nodes in the target component. Finally, the rule "undirected2connection" creates an undirected link "U" in the source component and relates it with two connections "C" for the communication in both directions between the "Y" nodes that are already related to the "X" nodes in the source component.*

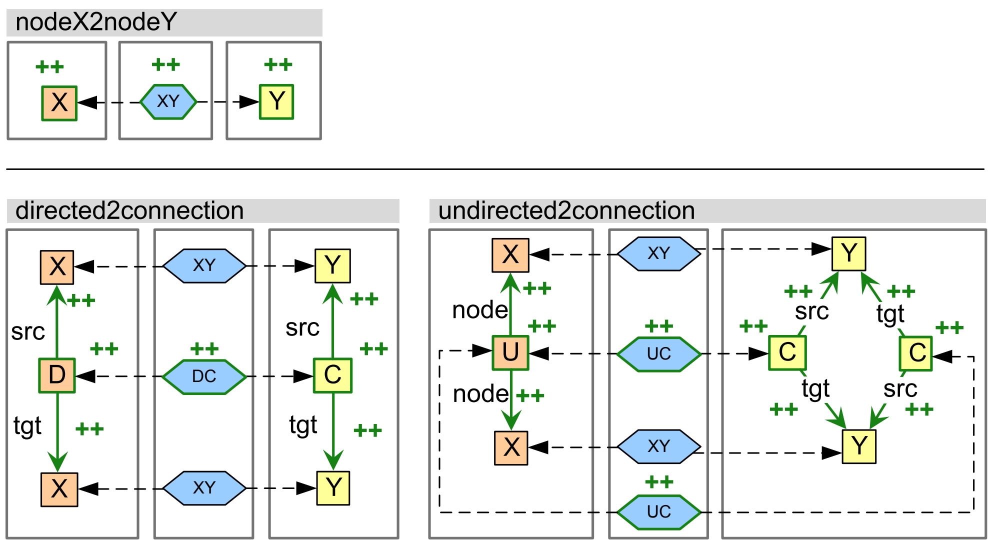

Figure 4: Triple Rules of the Triple Graph Grammar *TGG*

Based on the triple rules of a triple graph grammar, the operational source and forward rules for model transformations from models of the source language to models of the target language are derived automatically [Sch94, KW07, EEE+07]. The source rules will be used to parse the given source model of a forward model transformation, which guides the forward transformation, in which the forward rules are applied. Since triple rules have a symmetric character, the backward rules for backward model transformations from models of the target to models of the source language can be derived as well.

Definition 3 (Derived Source and Forward Rule) Given a triple rule $tr = (tr^S, tr^C, tr^T) : L \to R$ the source rule $tr_S : L_S \to R_S$ is derived by extending the graph morphism $tr^S : L^S \to R^S$ with empty graphs and empty morphisms for the remaining correspondence and target components, i.e. $L^C_S = L^T_S = R^C_S = R^T_S = \emptyset$. The forward rule $tr_F = (tr^S_F, tr^C_F, tr^T_F)$ is derived by taking tr and redefining the following components: $L^S_F = R^S$, $tr^S_F = id$, and $s_{L_F} = tr^S \circ s_L$.

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

Triple Rule tr

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

Source Rule tr_S

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

Forward Rule tr_F

Example 3 (Derived Rules) *The derived forward rules and one derived source rule of the triple*

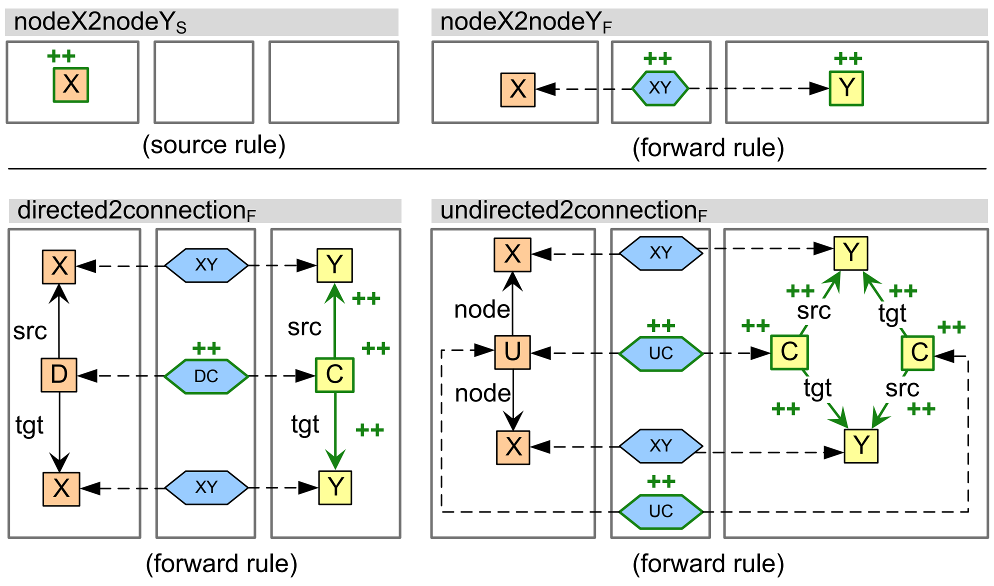

Figure 5: Some Derived Source and Forward Rules

graph grammar TGG in Fig. 4 are shown in Fig. 5. The source rule "nodeX2nodeY$_S$" creates a single "X" node and will be used to parse all nodes with label "X" in a given source model of a model transformation. Based on the found matches the corresponding forward rule "nodeX2nodeY$_F$" will be applied and it will insert a "Y" node in the target component for each detected "X" node. Similarly, the other two forward rules specify the completion of the correspondence and target structure for communication links in the source component. Directed "D" links are transformed to directed "C" connections between the already translated and corresponding "Y" nodes. For undirected "U" links we create two connections in both directions to complete the integrated model fragment.

As introduced in [EEE+07, EHS09] model transformations can be defined based on source consistent forward transformations $G_0 \Rightarrow^* G_n$ via $(tr_{1,F}, \dots, tr_{n,F})$, short $G_0 \xRightarrow{tr_F^*} G_n$. Source consistency intuitively means that the source model in G_0 can be parsed and all its elements are translated exactly once into corresponding fragments in the resulting target model. More precisely, source consistency of $G_0 \xRightarrow{tr_F^*} G_n$ requires that there is a source sequence $\varnothing \xRightarrow{tr_S^*} G_0$ such that the sequence $\varnothing \xRightarrow{tr_S^*} G_0 \xRightarrow{tr_F^*} G_n$ 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 $G_0 \xRightarrow{tr_F^*} G_n$ is controlled by the corresponding source sequence $\varnothing \xRightarrow{tr_S^*} G_0$, which is unique in the case of match consistency.

Definition 4 (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 source consistent forward TGT-sequence $G_0 \xRightarrow{tr_F^*} G_n$ with $G_S = proj_S(G_0)$ and $G_T = proj_T(G_n)$, where "$proj_X$" is the projection to the X-component of a triple graph for $X \in \{S,C,T\}$. 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 $G_T \in VL_{T0}$.

Considering the source model in Fig. 3 we can construct the following source consistent forward transformation: with $G_S = G^S$: $(G^S \leftarrow \emptyset \rightarrow \emptyset) = G_0 \xRightarrow{nodeX2nodeY_F, m_1} G_1 \xRightarrow{nodeX2nodeY_F, m_2} G_2 \xRightarrow{directed2connection_F, m_3} G_3 \xRightarrow{undirected2connection_F, m_4} G_4 = (G^S \leftarrow G^C \rightarrow G^T)$ and we derive the integrated model $G = G_4$ and the target model $G_T = G^T$ as shown in Fig. 3.

4 Results for Model Transformations Based on Triple Graph Grammars

There are already many important results for model transformations based on triple graph transformation and in this section we compare the available results with respect to the listed challenges in Sec. 2.

Model transformations based on source consistent forward sequences are syntactically correct and complete with respect to the triple patterns [EEHP09], i.e. with respect to the language $VL = \{G \mid \emptyset \Rightarrow^* G \text{ in } TGG\}$ containing the integrated models generated by the triple rules. More precisely, each model transformation translates a source model into a target model, such that the integrated model that contains both models can be created by applications of the triple rules to the empty start graph. This means that both models can be synchronously created according to the triple patterns. Vice versa, a model transformation can be performed on each source model that is part of an integrated model in the generated triple language VL.

For the more formal view on these results we explicitly define the language of translatable source models VL_S and of reachable target models VL_T by $VL_S = \{G_S \mid (G_S \leftarrow G_C \rightarrow G_T) \in VL\}$ and $VL_T = \{G_T \mid (G_S \leftarrow G_C \rightarrow G_T) \in VL\}$. As shown in [EHS09] and extended in [HEOG10, HEGO10] model transformations based on TGGs using the control condition source consistency are syntactically correct and complete.

Theorem 1 (Syntactical Correctness) *Each model transformation sequence given by $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$, which is based on a source consistent forward transformation sequence $G_0 \xRightarrow{tr_F^*} G_n$ with $G_0 = (G_S \leftarrow \emptyset \rightarrow \emptyset)$ and $G_n = (G_S \leftarrow G_C \rightarrow G_T)$ is syntactically correct, i.e. $G_n \in VL$.*

Theorem 2 (Completeness) *For each $G_S \in VL_S$ there exists a model $G_T \in VL_T$ with a model transformation sequence $(G_S, G_0 \xRightarrow{tr_F^*} G_n, G_T)$ where $G_0 \xRightarrow{tr_F^*} G_n$ is source consistent with $G_0 = (G_S \leftarrow \emptyset \rightarrow \emptyset)$ and $G_n = (G_S \leftarrow G_C \rightarrow G_T)$.*

Functional behaviour of model transformations ensures unique results for any given source model. A powerful as well as efficient technique for analyzing functional behaviour of model transformations based on TGGs is presented in [HEOG10, HEGO10] based on the generation of translation attributes and using the critical pair analysis engine of the tool AGG [AGG09]. The presented example in this paper already shows functional behaviour for the forward transformation. However, for the backward direction the behaviour is not functional. Consider, e.g., two "Y" nodes that are connected by two connections "C" in opposite direction. They can be transformed to one unidirectional link or to two directed links.

Concerning the non-functional properties of model transformations in the second list of challenges in Sec. 2 triple graph transformations show a very promising basis providing already most of the requested properties while the existing results above are preserved. In order to define expressive model transformations, the concept of negative application conditions (NACs) is commonly used and allows the modeler to specify complex model transformations [EHS09]. We are currently working on the extension of model transformations based on TGGs to the more general nested applications [HP09], which provide the expressive power of first order logic on graphs. Furthermore, as shown in [EEE+07], information preserving bidirectional model transformations can be characterized by source consistent forward transformations based on triple graph grammars. Moreover, the efficiency of executing source consistent model transformations is improved in [EEHP09] by defining an on-the-fly construction, for which termination is ensured if the source rules are creating, i.e. each triple rule creates at least one element in the source component. As a second optimization, suitable conditions for parallel independence were defined in [EEHP09] in order to perform partial order reductions. The efficiency is further improved in [HEGO10] using translation attributes and a sufficient condition for avoiding backtracking completely. Finally, model transformations based on triple graph transformations are flexible in the sense that new rules can be added without changing the existing rules whenever new structures are introduced into the visual language.

Coming back to the first list of challenges in Sec. 2 we prove in Sec. 5.2 the semantical correctness of the model transformation presented in this paper and we show how this approach can be generalized to other model transformations as well.

Summing up, triple graph grammars are an adequate and promising basis for model transformations and the existing results show its intuitive, expressive, formally well-founded and efficient character.

5 Verification of Model Transformations

5.1 The Borrowed Context Technique

In the following we will describe how to verify model transformations, by treating the case study introduced above. The approach we are using here has been described in more detail in [HKR+10b, HKR+10a] for a different case study. We will here omit the technical details and refer the interested reader to [HKR+10a].

Before we can even state what behaviour preservation actually means in our setting, it is necessary to introduce an operational semantics, given by graph transformation rules, for both the source and the target model. This operational semantics will equip source as well as target models with labelled transition systems, where transitions correspond to the application of graph transformation rules and are of the form $G_1 \overset{\alpha}{\Rightarrow} G_2$. Note that α is the transition label, which is obtained from the applied production p via a given *map*-function, i.e., $\alpha = map(p)$. The *map*-function, assigning a global label to every rule, is necessary since we compare different operational rules. Now, behaviour preservation in our setting means that the source model and the corresponding target model are bisimilar (with respect to the labelled transitions).

We will use the borrowed context technique [EK06, RKE08], which refines a labelled transition system (or even unlabelled reaction rules) in such a way that the resulting bisimilarity is a congruence (see also [LM00]). By a congruence we mean a relation over graphs that is preserved by contextualization, i.e., by gluing with a given environment graph over a specified interface. This is a mild generalization of standard graph rewriting in that we consider "open" graphs, equipped with a suitable interface.

Note that in this section we will not work directly with triple graph grammars, however in the conclusion we will discuss some preliminary ideas on the verification of model transformation based directly on triple graph grammars. Instead here we use in-situ transformation rules, where the in-situ rules are derived from the triple rules of Sec. 3, in order to be able to exploit the existing congruence results. The derivation of equivalent in-situ transformation rules has been done manually, it is however quite straightforward in this case. We are not using the usual forward transformation rules since they would be larger and quite unwieldy for our purposes.

The basic idea behind the borrowed context technique is to describe the possible interactions of a part of the model with the environment, i.e., with the remaining yet unspecified rest of the model. In addition to existing labels, we add the following information to a transition: what is the (minimal) context that a graph with interface needs to evolve? More concretely we have transitions of the form

$$(J \to G) \overset{\alpha,(J\to F\leftarrow K)}{\Longrightarrow} (K \to H)$$

where the components have the following meaning: $(J \to G)$ is the original graph with interface J (given by an injective morphism from J to G) which evolves into a graph H with interface K. The label is composed of two entities: the original label $\alpha = map(p)$ stemming from the operational rule p and furthermore two injective morphisms $(J \to F \leftarrow K)$ detailing what is borrowed from the environment. The graph F represents the additional graph structure, whereas J, K are the two interfaces (of G and H) which are mapped to F via graph morphisms.

We will now introduce the necessary definitions.

Definition 5 (context, cospan) A *graph with interface* is a graph morphism $J \to G$.

A *context* (also called *cospan*) consists of two injective graph morphisms $J \to F \leftarrow K$. The composition of two cospans is performed by taking the pushout.

$$\begin{tikzcd}
 & & K \arrow[dl] \arrow[dr] & & \\
J \arrow[r, tail] \arrow[drr, dotted, bend right] & F \arrow[dr] & PO & E \arrow[dl] & M \arrow[l, tail] \arrow[dll, dotted, bend left] \\
 & & D & &
\end{tikzcd}$$

Definition 6 (Rewriting with Borrowed Contexts) Given a graph with interface $J \to G$ and a production $p\colon L \leftarrow I \to R$, we say that $J \to G$ reduces to $K \to H$ with transition label $J \to F \leftarrow K$ if there are graphs D, G^+, C and additional morphisms such that the diagram below commutes and the squares are either pushouts (PO) or pullbacks (PB) with injective morphisms. In this case a *rewriting step with borrowed context* exists and is written as follows:

$$(J \to G) \overset{map(p),(J \to F \leftarrow K)}{\Longrightarrow} (K \to H)$$

(in words: $J \to G$ reduces to $K \to H$ with transition labels $map(p)$ and $J \to F \leftarrow K$).

$$\begin{tikzcd}
D \arrow[r] \arrow[d] \arrow[dr, phantom, "PO"] & L \arrow[d] \arrow[dr, phantom, "PO"] & I \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "PO"] & R \arrow[d] \\
G \arrow[r] & G^{+} & C \arrow[l] \arrow[r] & H \\
J \arrow[r] \arrow[u] \arrow[ur, phantom, "PO"] & F \arrow[u] \arrow[ur, phantom, "PB"] & K \arrow[l] \arrow[u] \arrow[ur, dotted]
\end{tikzcd}$$

After these preliminaries, we can now define the notion of bisimilation and bisimilarity with borrowed context labels. Note that under certain conditions and for closed systems this notion specializes to standard bisimilarity, which ignores the borrowed context label. This will be explained later in more detail.

Definition 7 (Bisimulation, Bisimilarity) Let $\mathscr{P}$ be a set of productions. Let $\mathscr{R}$ be a symmetric relation consisting of pairs of graphs with interfaces of the form $(J \to G, J \to G')$, also written $(J \to G)\mathscr{R}(J \to G')$.

The relation $\mathscr{R}$ is a *bisimulation* if whenever we have $(J \to G)\mathscr{R}(J \to G')$ and a transition $(J \to G) \overset{\alpha,(J \to F \leftarrow K)}{\Longrightarrow} (K \to H)$ can be derived from $\mathscr{P}$, then there exists a morphism $K \to H'$ and a transition $(J \to G') \overset{\alpha,(J \to F \leftarrow K)}{\Longrightarrow} (K \to H')$ such that $(K \to H)\mathscr{R}(K \to H')$.

We write $(J \to G) \sim (J \to G')$ whenever there exists a bisimulation $\mathscr{R}$ that relates the two morphisms. The relation $\sim$ is called *bisimilarity*.

We have shown that (strong) bisimilarity defined in transition systems with borrowed context labels is a congruence. This holds also if we enrich the labels with $\alpha = map(p)$ as described above. This extended congruence result was shown to be correct in [HKR+10a].

Theorem 3 (Bisimilarity is a Congruence [EK06, HKR+10a]) *Bisimilarity $\sim$ is a congruence, i.e., it is preserved by embedding into contexts as specified in Def. 5.*

5.2 Using the Borrowed Context Technique for the Verification of Model Transformations

For an in-situ model transformation within the same language, applications of the borrowed context technique are quite immediate: show for every transformation rule that the left-hand and right-hand sides L, R with interface I are bisimilar with respect to the operational rules. Then the source model must be bisimilar to the target model by the congruence result. This idea has been exploited in [RLK+08] for showing behaviour preservation of refactorings.

To set up the entire machinery, we first need the operational semantics of the two languages under consideration (BiDiLang and UniDiLang). In Fig. 6 and Fig. 7 we describe the dynamic evolution of a system: in both cases messages can be created and deleted at arbitary moments in

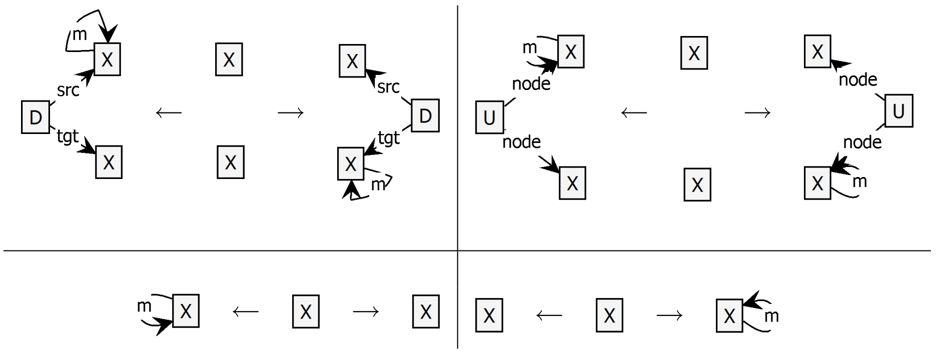

Figure 6: BiDiLang, rules of the operational semantics

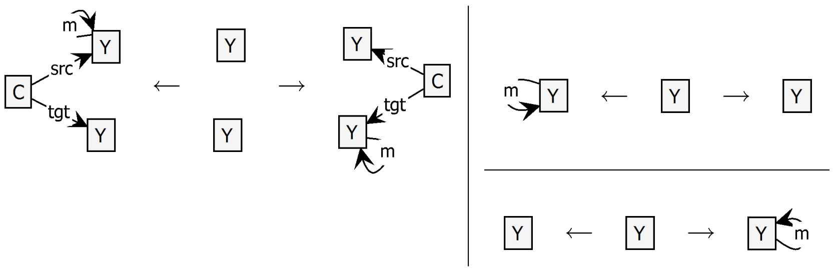

Figure 7: UniDiLang, rules of the operational semantics

time. Furthermore, in language BiDiLang the node labelled D describes a directed connection over which messages can be passed in only one direction, whereas the node labelled U describes an undirected connection allowing a movement in any direction (note that the two edges leaving the U-node have the same label and are hence undistinguishable). In the second language (UniDiLang) we have only one type of connection, working similarly to the directed connection in the first language.

Now, as announced above, in order to reuse the congruence result we are applying in-situ transformation rules (given in Fig. 8) which are similar to the triple graph grammar rules given in Sec. 3.

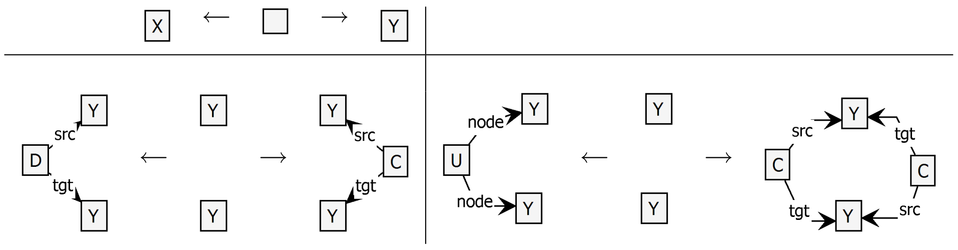

Figure 8: Rules for the in-situ model transformation

Note that these in-situ rules will lead to "mixed" (or hybrid) models which incorporate components of both the source and the target model. Hence we need a joint type graph (see Fig. 9) that contains node and edge types of both languages.

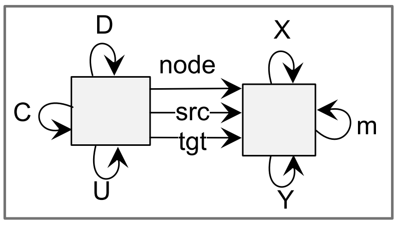

Figure 9: Combined Type Graph TG_{ST} for mixed Models

Furthermore, since we generate mixed models but still want to exploit the congruence result, it is necessary to have an operational semantics also for those models, which has to satisfy the following conditions: (i) the mixed rules are *not* applicable to a pure source or target model; (ii) it is possible to show bisimilarity of left-hand and right-hand sides of all transformation rules. Finally, observe that our final aim is to show bisimilarity of closed graphs, i.e., of graphs with empty interface of the form $\emptyset \to G$. If the operational rules of the source and target languages have connected left-hand sides then such a graph will either borrow nothing or borrow the whole left-hand side. It can be shown that if all left-hand sides are connected, the notion of bisimilarity induced by borrowed contexts coincides with the standard one.

Hence here we use the mixed operational semantics given in Fig. 10. The rules mainly describe message passing in mixed models, where a message is, for instance, passed from an X-node to a Y-node over various types of connectors.

We are now ready to give the main result of this section, which states the correctness of the model transformation under consideration. This theorem holds in general whenever the mixed semantics satisfies Conditions (i) and (ii) above.

Theorem 4 *The three rules of the in-situ model transformation given in Fig. 8 form a bisimulation relation $\mathscr{R}$, where each rule $L \leftarrow I \rightarrow R$ is split into a pair $(I \rightarrow L, I \rightarrow R)$ of the relation. Since bisimilarity is a congruence and borrowed context bisimilarity coincides with standard bisimilarity on source and target models, this implies that whenever a graph G_B of the source language is transformed into a graph G_U of the target language via the model transformation, then G_B is bisimilar to G_U.*

Note that in the proof we make heavy use of the up-to-context technique, which allows us to somewhat relax the requirements for bisimulation proofs given in Def. 7. More specifically, it is enough if $K \rightarrow H$ and $K \rightarrow H'$ are in relation $\mathscr{R}$ after the removal of identical contexts. Note also that in more complex scenarios the bisimulation $\mathscr{R}$ might contain additional pairs that are not model transformation rules (see [HKR+10a]).

In this fairly easy scenario one can obtain the rules of the mixed semantics by applying the transformation rules to the (original) operational semantics of the source or target languages. In the general case, it is however currently not clear to us, how to obtain a correct set of mixed

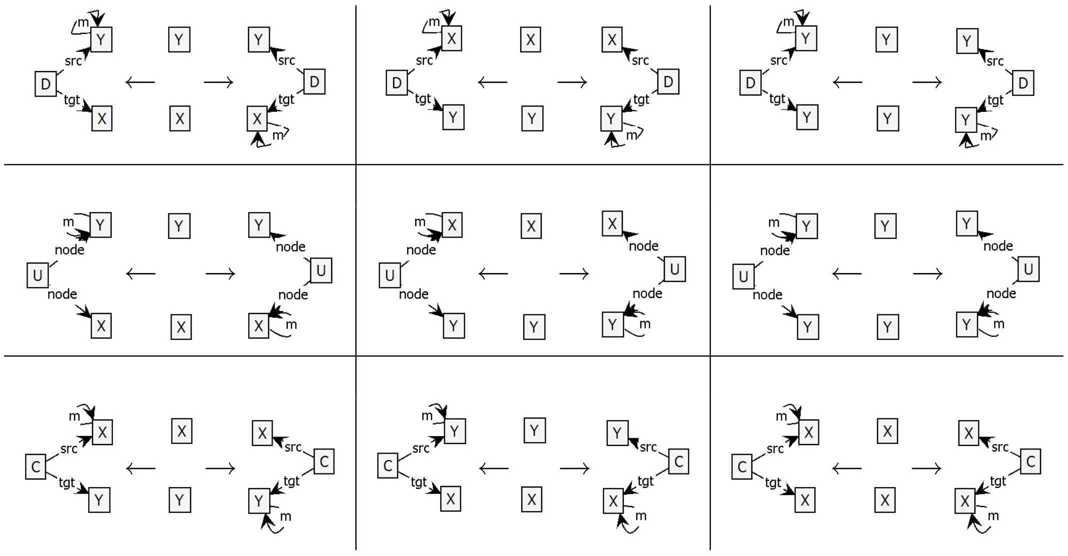

Figure 10: Additional rules of the mixed semantics

semantic rules. For small examples, the following heuristics usually gives good results:

1. Let S be the set containing all original rules of the the source and target operational semantics.
2. Choose any tranformation rule r.
3. Apply all rules in S to the left-hand side (respectively right-hand side) of r using the borrowed context technique. This gives us several borrowed context rewriting steps.
4. If there is a matching answer with a rule in S for the right-hand side (respectively left-hand side) of transformation rule r, then do nothing.
5. If there is no such matching answer, create a new “mixed” rule, providing such a valid answer. Add this new rule to S and proceed with step 2.
6. If every partial map of every rule in S has been tested on all left-hand and right-hand sides of the transformation rules, S is the mixed semantics we are looking for.

Using this heuristics one might even create a smaller set of rules for the mixed semantics than by applying the transformation rules to the rules for the operational semantics in every possible way (see [HKR$^+$10a]).

6 Related Work

There are several other approaches based on triple graph transformation, e.g. using constraint-patterns [OGLE09]. While these patterns can lead to a more compact specification, there are fewer results for several of the listed challenges, e.g. the handling of termination and therefore completeness is more complex and not ensured in general.

As mentioned before, there are already suitable techniques for the analysis of functional behaviour of model transformations based on plain graph transformation systems [EEL+05]. However, plain graph transformation systems do not show some of the important benefits of triple graph transformation, as, for instance, completeness and the general notion of syntactical correctness with respect to the triple patterns specified by the intuitive triple rules. Furthermore, plain graph transformation systems are unidirectional while triple graph transformation systems automatically provide bidirectional model transformations.

The work closest to ours for showing the semantical correctness of model transformations in the sense of showing behaviour preservation for a transformation between models of different types is [GGL+06]. They present a mechanised proof of semantics preservation for a transformation of automata to PLC-code, based on TGG rules. This proof faced some problems since it was not trivial to present graph transformation within Isabelle/HOL.

As opposed to model transformation between different source and target models, there has been more work on showing behaviour preservation in refactoring. The methods presented in [KCKB05, PC07, NK06, GSMD03] address behaviour preservation in model refactoring, but are in general limited to checking a certain number of models. The employment of a congruence result is also proposed in [BHE08] which uses the process algebra CSP as a semantic domain. In [BEH07] it is shown how to exploit confluence results for graph transformation systems in order to show correctness of refactorings. A number of approaches to showing correctness of refactorings also focus on preserving specific aspects instead of the full semantics (see [MT04]).

7 Conclusion

In order to provide validated model transformations, which are a major component in model driven architecture (MDA), there is a strong need for formal analysis and verification. We have shown that triple graph transformation is an adequate technique providing both, an intuitive way of specification and a formal basis for which several analysis techniques as well promising execution algorithms are available. The two lists of challenges for model transformations in Sec. 2 contain many different and important aspects and, depending on the concrete model transformation, there may be some of them that cannot be achieved.

Even though, the presented approach in Sec. 3 based on triple graph transformation shows many capabilities and many of the listed challenges can be achieved or handled adequately, respectively. The available results discussed in Sec. 4 include for instance syntactical correctness and completeness and the specification of model transformations is performed in an intuitive and elegant way. While the general analysis of functional behaviour of a model transformation will be a part of future work we have exemplarily shown how the specified model transformation can be analyzed with respect to behaviour preservation and therefore, with respect to semantical correctness.

For this purpose we transformed in Sec. 5 the model transformation based on a triple graph grammar into an in-situ model transformation based on plain graph grammars. In a next step we introduced a proof technique for showing that a transformation preserves the behaviour of a model. A similar method was introduced by us in [HKR+10b, HKR+10a] for a different model transformation. In [HKR+10b, HKR+10a] it was even necessary to work with weak, saturated bisimilarity with negative application conditions due to the higher complexity of the case study. However, the general idea can just as well be presented and understood with the simpler case study presented in this paper.

Currently we have not yet mechanized the technique, but we have started to work on an implementation. One drawback is the fact that it is necessary to find a suitable mixed semantics, which might become quite large and unwieldy. Hence we are currently working on a more straightforward approach that combines triple graph grammars with borrowed contexts, by asking that each borrowed context step of the source model must be answered by a borrowed context step of the target model (and vice versa) in such a way that the labels can be translated into each other via the model transformation rules. However there are some remaining technical difficulties (e.g., what happens if the label can only be partially translated?) yet to be solved. For both approaches it is not yet clear to which extent they will scale. We believe that additional proof techniques will be necessary to treat more realistic examples.

Note however that the in-situ transformation rules are not without merits: in the case of system migration, where we migrate piece by piece of an evolving system from one version to another, we might well have such mixed intermediate states which have to be handled. Think of a heterogeneous LAN, where one wants to replace the mail server, the firewall and the file server. The complete system must be in working order all the time, but in many cases the exchange of the components will not happen synchronously. In such a setting we want to show that also the hybrid models preserve the behaviour and the migration does not disrupt the correct working of the system.

Acknowledgements: We would like to thank Arend Rensink, Maria Semenyak, Christian Soltenborn and Heike Wehrheim for joint work on a case study, which gave us the ideas on which we based Sec. 5.

Bibliography

[AGG09] TFS-group, TU Berlin. AGG. 2009. http://tfs.cs.tu-berlin.de/agg.

[BEH07] L. Baresi, K. Ehrig, R. Heckel. Verification of Model Transformations: A Case Study with BPEL. In *Proc. of TGC '07 (Trustworthy Global Computing)*. Pp. 183–199. Springer, 2007. LNCS 4661.

[BHE08] D. Bisztray, R. Heckel, H. Ehrig. Verification of Architectural Refactorings by Rule Extraction. In *FASE '08*. LNCS 4961, pp. 347–361. Springer, 2008.

[EEE+07] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, G. Taentzer. Information Preserving Bidirectional Model Transformations. In Dwyer and Lopes (eds.), *Fundamental Approaches to Software Engineering*. LNCS 4422, pp. 72–86. Springer, 2007. http://tfs.cs.tu-berlin.de/publikationen/Papers07/EEE+07.pdf

[EEHP09] H. Ehrig, C. Ermel, F. Hermann, U. Prange. On-the-Fly Construction, Correctness and Completeness of Model Transformationsbased on Triple Graph Grammars: Long Version. In Schürr and Selic (eds.), *ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09)*. lncs 5795, pp. 241–255. Springer, 2009. To appear.
http://tfs.cs.tu-berlin.de/publikationen/Papers09/EEHP09.pdf

[EEL+05] H. Ehrig, K. Ehrig, J. de Lara, G. Taentzer, D. Varró, S. Varró-Gyapay. Termination Criteria for Model Transformation. In Wermelinger and Margaria-Steffen (eds.), *Proc. Fundamental Approaches to Software Engineering (FASE)*. Lecture Notes in Computer Science 2984, pp. 214–228. Springer Verlag, 2005.
http://tfs.cs.tu-berlin.de/publikationen/Papers05/EEL+05.pdf

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theor. Comp. Science. Springer Verlag, 2006.
http://www.springer.com/3-540-31187-4

[EHS09] H. Ehrig, F. Hermann, C. Sartorius. Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions. *ECEASST* 18, 2009.
http://eceasst.cs.tu-berlin.de/index.php/eceasst/issue/view/27

[EK06] H. Ehrig, B. König. Deriving Bisimulation Congruences in the DPO Approach to Graph Rewriting with Borrowed Contexts. *Mathematical Structures in Computer Science* 16(6):1133–1163, 2006.

[GGL+06] H. Giese, S. Glesner, J. Leitner, W. Schäfer, R. Wagner. Towards Verified Model Transformations. In *3rd International Workshop on Model Development, Validation and Verification (MoDeVa)*. Pp. 78–93. Le Commissariat á l'Energie Atomique - CEA, Genova, Italy, 2006.

[GSMD03] P. V. Gorp, H. Stenten, T. Mens, S. Demeyer. Towards automating source-consistent UML refactorings. In *UML 2003*. LNCS 2863, pp. 144–158. Springer, 2003.

[HEGO10] F. Hermann, H. Ehrig, U. Golas, F. Orejas. Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars. Technical report, TU Berlin, Fak. IV, 2010. To appear, available online: http://tfs.cs.tu-berlin.de/publikationen/Papers10/HEGO10b.pdf.

[HEOG10] F. Hermann, H. Ehrig, F. Orejas, U. Golas. Formal Analysis of Functional Behaviour of Model Transformations Based on Triple Graph Grammars. In *Proc. Int. Conf. on Graph Transformation*. 2010. accepted, online available at http://tfs.cs.tu-berlin.de/publikationen/Papers10/HEOG10.pdf).

[HKR+10a] M. Hülsbusch, B. König, A. Rensink, M. Semenyak, C. Soltenborn, H. Wehrheim. Full Semantics Preservation in Model Transformation – A Comparison of Proof

Techniques. Technical report TR-CTIT-10-09, Centre for Telematics and Information Technology, University of Twente, 2010.

[HKR+10b] M. Hülsbusch, B. König, A. Rensink, M. Semenyak, C. Soltenborn, H. Wehrheim. Showing Full Semantics Preservation in Model Transformation – A Comparison of Techniques. In *Proc. of iFM '10 (Integrated Formal Methods)*. Springer, 2010. LNCS, to appear.

[HP09] A. Habel, K.-H. Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science* 19(2):245–296, 2009.

[KCKB05] M. van Kempen, M. Chaudron, D. Kourie, A. Boake. Towards proving preservation of behaviour of refactoring of UML models. In *SAICSIT '05*. Pp. 252–259. 2005.

[KW07] E. Kindler, R. Wagner. Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios. Technical report TR-ri-07-284, Universität Paderborn, 2007.

[LM00] J. J. Leifer, R. Milner. Deriving Bisimulation Congruences for Reactive Systems. In *Proc. of CONCUR 2000*. Pp. 243–258. Springer, 2000. LNCS 1877.

[MT04] T. Mens, T. Tourwé. A Survey of Software Refactoring. *IEEE Trans. Software Eng.* 30(2):126–139, 2004.

[NK06] A. Narayanan, G. Karsai. Towards Verifying Model Transformations. In *GT-VMT '06*. ENTCS 211, pp. 185–194. 2006.

[OGLE09] F. Orejas, E. Guerra, J. de Lara, H. Ehrig. Correctness, Completeness and Termination of Pattern-Based Model-to-Model Transformation. In Kurz et al. (eds.), *Proc. of the 3rd Int. Conf. on Algebra and Coalgebra in Computer Science (CALCO'09)*. Lecture Notes in Computer Science 5728, pp. 383–397. Springer, 2009.

[PC07] J. Pérez, Y. Crespo. Exploring a Method to Detect Behaviour-Preserving Evolution Using Graph Transformation. In *Third International ERCIM Workshop on Software Evolution*. Pp. 114–122. 2007.

[RKE08] G. Rangel, B. König, H. Ehrig. Deriving Bisimulation Congruences in the Presence of Negative Application Conditions. In Amadio (ed.), *Proc. Foundations of Software Science and Computational Structures (FOSSACS'08)*. Lecture Notes in Computer Science 4962, pp. 413–427. Springer Verlag, 2008.
doi:10.1007/978-3-540-78499-9
http://www.springerlink.com/content/e950520638346408/

[RLK+08] G. Rangel, L. Lambers, B. König, H. Ehrig, P. Baldan. Behavior Preservation in Model Refactoring using DPO Transformations with Borrowed Contexts. In *Proc. International Conference on Graph Transformation (ICGT'08)*. Lecture Notes in Computer Science 5214. Springer Verlag, Heidelberg, 2008.

[Sch94] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In Tinhofer (ed.), *WG94 20th Int. Workshop on Graph-Theoretic Concepts in Computer Science*. Lecture Notes in Computer Science 903, pp. 151–163. Springer Verlag, Heidelberg, 1994.

[SK08] A. Schürr, F. Klar. 15 Years of Triple Graph Grammars. In *Proc. Int. Conf. on Graph Transformation (ICGT 2008)*. Pp. 411–425. 2008.
doi:10.1007/978-3-540-87405-8_28