Correctness and Completeness of Generalised Concurrent Model Synchronisation Based on Triple Graph Grammars

Susann Gottmann[1], Frank Hermann[1], Nico Nachtigall[1], Benjamin Braatz[1], Claudia Ermel[2], Hartmut Ehrig[2], and Thomas Engel[1]

[1] Université du Luxembourg, Luxembourg
firstname.lastname@uni.lu
[2] Technische Universität Berlin, Germany
firstname.lastname@tu-berlin.de

Abstract. Triple graph grammars (TGGs) have been applied successfully for specifying and analysing bidirectional model transformations. Recently, a formal approach to concurrent model synchronisation has been presented, where source and target modifications are synchronised simultaneously. In addition to methods for update propagation, the approach includes a semi-automatic strategy for conflict resolution. Up to now, this approach has been limited to deterministic propagation operations.
In this paper, we generalise the approach to arbitrary TGGs and consider non-deterministic operations which might yield different results and require backtracking. We show correctness and completeness of the extended approach and provide a technique for reducing and possibly eliminating backtracking to improve efficiency.

Keywords: concurrent model synchronisation, bidirectional model transformation, triple graph grammars

1 Introduction

Bidirectional model transformations have been specified and analysed successfully using triple graph grammars (TGGs) [17,18]. More recently, TGGs have also been applied in case studies for model integration and model synchronisation [14,5,7]. Model synchronisation aims to propagate updates between interrelated domains in order to derive updated models that are consistent with each other. The formal results concerning correctness and completeness are a major advantage of TGGs in this field [12]. Consistency conditions are specified in a concise way and they are automatically respected by the synchronisation process. In an industrial project on model transformations for satellite systems, we make explicit use of these properties with great advantage [13].

Since model changes may occur concurrently in related domains and in a distributed way, model synchronisation has to cope with update propagation, update merging, and conflict resolution. In this paper, we use the concurrent model synchronisation approach based on triple graph grammars [9]. Possible conflicts are resolved in a semi-automated

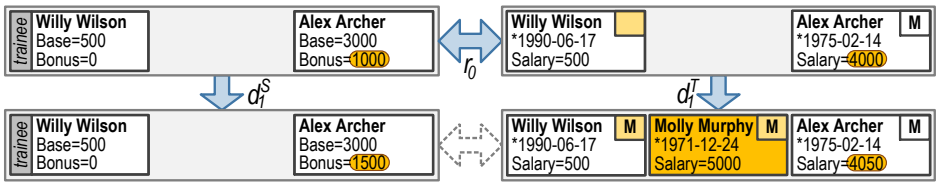

Fig. 1. Concurrent model synchronisation: running example

way. The operations are realised by model transformations based on TGGs [12] and tentative merge constructions solving conflicts [9].

Fig. 1 illustrates the running example of this paper in concrete syntax. The source domain (left) contains employees of the marketing department and trainees (indicated by a box *trainee*). The target domain (right) contains all employees. Upper right boxes indicate their assignment to departments (M = Marketing, empty = trainees). In target update d_1^T, Willy Wilson and Molly Murphy get hired by the marketing department. Bonus and salary values for Alex Archer are increased synchronously on both domains.

The synchronisation framework presented in [9] is limited to a restricted kind of TGGs that ensure deterministic propagation operations. This requirement is often complex to check for medium and large scale applications. In this paper, we generalise the approach to arbitrary TGGs, such that the operations are in general non-deterministic. This includes the deterministic case, but we do not have to check for determinism. Non-determinism means that the operations may require backtracking and may yield several possible results. In our first main result (Thm. 2.5), we show that the derived non-deterministic synchronisation framework is correct and complete for any given TGG.

In order to reduce backtracking efforts, we eliminate conflicts between operational rules by adding additional negative application conditions (NACs), called *filter NACs*. This concept was already applied successfully in the area of model transformations [10]. In a second step, the synchronisation operations are extended automatically with compatible NACs and we show in the second main result (Thm. 3.4) that these changes do not affect the correctness and completeness results for the derived synchronisation framework. In case that all relevant conflicts are eliminated, this ensures that concurrent model synchronisation can be performed efficiently, i.e., without backtracking.

Sec. 2 generalises the formal framework for concurrent model synchronisation [11,9] to non-deterministic forward and backward propagation operations. Sec. 3 provides an automated technique for improving efficiency. After discussing related work in Sec. 4, we conclude the paper and discuss directions for future work in Sec. 5. We provide full details of the example and the formal theory in an accompanying technical report [6].

2 Non-Deterministic Concurrent Synchronisation Framework

The synchronisation of concurrent model updates in two domains means to derive a new consistent integrated model together with corresponding updates on both domains. This section generalises the formal framework in [11,9] to the non-deterministic case.

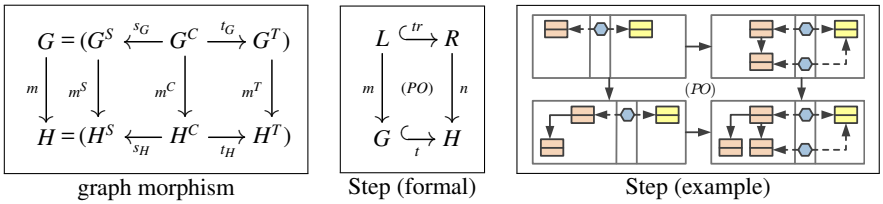

Fig. 2. Triple graph morphism and transformation step

Triple graph grammars (TGGs) are a suitable formal approach for defining a language of consistently integrated models [17,3]. An integrated model is represented by a triple graph G consisting of graphs G^S, G^C, and G^T, called source, correspondence, and target graphs, together with two mappings (graph morphisms) $s_G : G^C \to G^S$ and $t_G : G^C \to G^T$ for specifying the correspondence links between elements of G^S and G^T.

Triple graphs are related by triple graph morphisms $m : G \to H$ [17,3] consisting of three graph morphisms that preserve the associated correspondences (i.e., left diagrams in Fig. 2 commute). Triple graphs are typed over a triple type graph TG and attributed according to [3]. For a triple type graph $TG = (TG^S \leftarrow TG^C \to TG^T)$, we use $\mathcal{L}(TG)$, $\mathcal{L}(TG^S)$, and $\mathcal{L}(TG^T)$ to denote the classes of all graphs typed over TG, TG^S, and TG^T, respectively.

A triple graph grammar $TGG = (TG, S, TR)$ consists of a triple type graph TG, a triple start graph S and a set TR of triple rules, and generates the triple graph language of consistently integrated models $\mathcal{L}(TGG) \subseteq \mathcal{L}(TG)$ with consistent source and target languages $\mathcal{L}_S = \{G^S \mid (G^S \leftarrow G^C \to G^T) \in \mathcal{L}(TGG)\}$ and $\mathcal{L}_T = \{G^T \mid (G^S \leftarrow G^C \to G^T) \in \mathcal{L}(TGG)\}$. A model update $d : G \to G'$ is specified as a *graph modification* $d = (G \xleftarrow{i_1} I \xrightarrow{i_2} G')$ with inclusions $i_1 : I \hookrightarrow G$ and $i_2 : I \hookrightarrow G'$. Intuitively, all elements in $G \setminus I$ are deleted and all the elements in $G' \setminus I$ are added by d.

Example 2.1 (Triple Type Graph). Triple type graph TG in Fig. 3 specifies the structure of source and target models. In the source domain, persons are attributed with their first and last name, their detailed salary information (base and bonus) and their state of employment (trainee or permanent). Persons in the target domain are attributed with their first and last name, birth date and their total salary. The membership to a department is provided by a direct link to the department. Trainees have no link to any department.

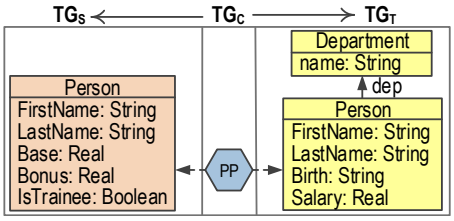

Fig. 3. Triple type graph

A triple rule $tr = (tr^S, tr^C, tr^T)$ is non-deleting and therefore, can be formalised as an inclusion from triple graph L (left hand side) to triple graph R (right hand side), represented by $tr : L \hookrightarrow R$. The application of a triple rule tr via a match morphism $m : L \to G$ yields a triple graph transformation (TGT) step $G \overset{tr,m}{\Longrightarrow} H$ with triple graph H defined by the pushout diagram (PO) in Fig. 2 (L is replaced by R in G) [18].

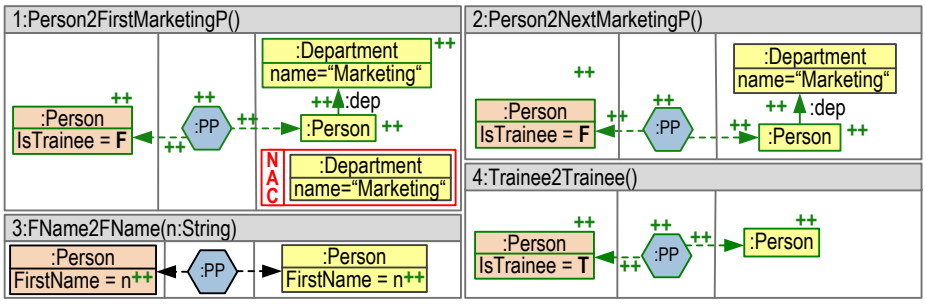

Fig. 4. Triple rules

Moreover, triple rules can be extended by negative application conditions (NACs) for restricting their application to specific matches [10].

Example 2.2 (Triple Graph Grammar). The TGG of our scenario is given by the triple type graph shown in Fig. 3, the empty start graph and the set of triple rules illustrated in compact notation in Fig. 4. All elements (nodes, edges or attributes) that are marked with ++ (green border) are added by a triple rule. Parts marked with a rectangle containing the label NAC (red border) describe negative application conditions [10]. The first triple rule Person2FirstMarketingP inserts a new department called "Marketing" into the target domain. The NAC ensures that there is no other department with the same name. In addition, this rule creates a person in both, the target and the source domain and creates a link to the marketing department (edge :dep). Attribute IsTrainee is set to **F**. Rule Person2NextMarketingP extends the model with a new person employed at the marketing department. Rule FName2FName sets the first name of a person in both, the source and the target domain. Rule Trainee2Trainee adds a trainee in both domains and creates a correspondence. The full TGG is presented in [6].

The signature of the concurrent synchronisation operation CSync is formalised in Fig. 5 (left) and specifies the type of the in- and output elements. Given an integrated model $G_0 = (G_0^S \leftrightarrow G_0^T)$, source model update $d_1^S = (G_0^S \rightarrow G_1^S)$ and target model update $d_1^T = (G_0^T \rightarrow G_1^T)$, we need to find source update $d_2^S = (G_1^S \rightarrow G_2^S)$ and target update $d_2^T = (G_1^T \rightarrow G_2^T)$ together with a new integrated model $G_2 = (G_2^S \leftrightarrow G_2^T)$ [9]. The integrated model may not be unique in the case of a non-deterministic synchronisation operation. Correctness (right of Fig. 5) ensures that any resulting integrated model $G_2 = (G_2^S \leftrightarrow G_2^T)$ is consistent (law (a)), i.e., $G_2 \in \mathcal{L}(TGG)$. Furthermore, if the given input is initially consistent and the updates do not change anything, then one of the possible outputs of operation CSync is identical to the input itself (law (b)). In case of deterministic TGGs, law (b) requires that the output is always the identity, because there is exactly one output. Completeness means that *CSync* yields at least one possible output for any input.

Definition 2.3 (Non-Deterministic Concurrent Synchronisation Problem and Framework). *Given a triple type graph TG, the* concurrent synchronisation problem *is to construct a non-deterministic operation* CSync *leading to the signature diagram*

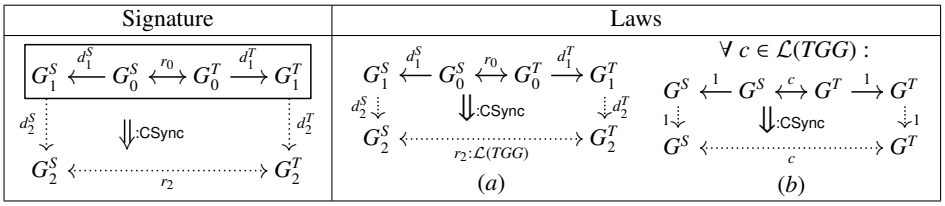

Fig. 5. Signature and laws for correct concurrent synchronisation frameworks

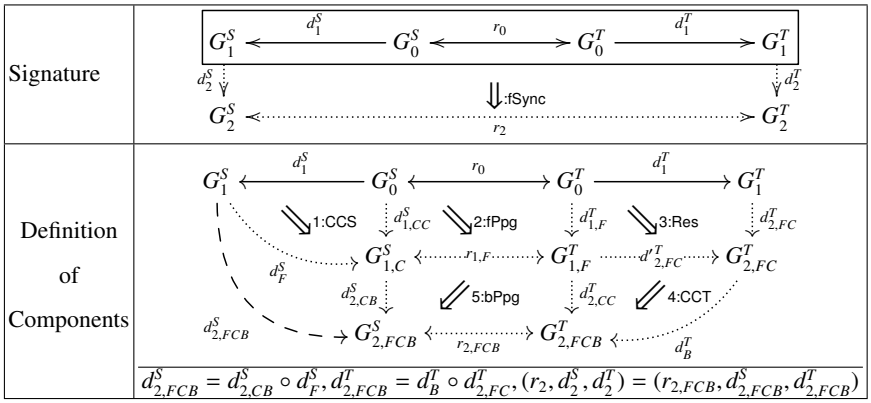

Fig. 6. Concurrent model synchronisation with conflict resolution (forward case: fSync)

in Fig. 5 with concurrent synchronisation operation CSync*. Given a triple graph grammar TGG* $= (TG, \varnothing, TR)$ *and a concurrent synchronisation operation* CSync*, the non-deterministic* concurrent synchronisation framework *CSync*$(TGG,$ CSync$)$ *is called* correct*, if laws* (*a*) *and* (*b*) *in Fig. 5 are satisfied; and it is called* complete*, if operation* CSync *is a left total relation.*

The formal approach to concurrent model synchronisation based on TGGs [9] is performed in five steps (see Fig. 6). We describe the forward case (operation fSync), i.e., the synchronisation is initiated in the source domain. The symmetric backward case (bSync) works analogously by switching the roles of source and target domains and exchanging fPpg with bPpg, and CCS (consistency creating on source domain) with CCT (consistency creating on target domain). The concurrent synchronisation operation CSync = (fSync ∪ bSync) is defined by the union of both cases.

Concept 2.4 (Execution of non-deterministic synchronisation framework). In contrast to previous work [9], we do not require deterministic TGGs, such that all five steps may yield several results and steps 1,2,4, and 5 may require backtracking. The first step (1:CCS) is executed via consistency creating operation CCS on the source domain and computes a maximal sub-model $G^S_{1,C} \in \mathcal{L}_S$ of the given model G^S_1 that is consistent with respect to the language $\mathcal{L}_S$ (language of consistent source models). We obtain source update $d^S_{1,CC}\colon G^S_0 \to G^S_{1,C}$. In general, consistency creating operations CCS (source domain, step 1) and CCT (target domain, step 4) remove struc-

tures that cannot be translated, i.e., those that cannot appear in any consistent integrated model. In step 2, we apply forward propagation operation fPpg (Fig. 6) to propagate the changes to the target domain and we obtain target update $d^T_{1,F}$ and integrated model $G_{1,F} = (G^S_{1,C} \leftrightarrow G^T_{1,F})$. In step 3, we apply conflict resolution operation Res [9] in order to merge the two updates on the target domain: the propagated update $d^T_{1,F}$ and the given update d^T_1. This leads to a new target update $d'^T_{2,FC}\colon G^T_{1,F} \to G^T_{2,FC}$. We apply consistency creating operation CCT to obtain the maximal consistent sub-model of $G^T_{2,FC}$ and derive the target updates $d^T_B\colon G^T_{2,FC} \to G^T_{2,FCB}$ and $d^T_{2,CC}\colon G^T_{1,F} \to G^T_{2,FCB}$. Finally, we propagate update $d^T_{2,CC}$ from the target to the source domain via backward propagation operation bPpg leading to source update $d^S_{2,CB}\colon G^S_{1,C} \to G^S_{2,FCB}$ and integrated model $G_{2,FCB} = (G^S_{2,FCB} \leftrightarrow G^T_{2,FCB})$.

Our first main result (Thm. 2.5) shows that the non-deterministic concurrent synchronisation framework is correct and complete, such that all outputs are consistent and all all inputs can be processed. Termination is ensured, if each operational translation rule changes at least one translation attribute, which can be efficiently checked [9,12].

Theorem 2.5 (Correctness and Completeness of Non-Deterministic Concurrent Synchronization Framework). *Given a triple graph grammar TGG, the derived non-deterministic concurrent synchronisation framework CSync(TGG,* CSync*) is correct and complete.*

Proof (Idea). We extend the result for deterministic TGGs in [9] using the general composition and decomposition theorem for TGGs [3]. The main step is to show that backtracking of forward propagation (step 2) and backward propagation (step 5) ensures that the approach computes a correct sequence. For the full formal construction and proof see [6]. □

3 Efficiency Improvement

This section shows how to reduce and possibly eliminate backtracking of the general non-deterministic concurrent model synchronisation framework in Sec. 2 by removing conflicts between the operational rules of a TGG using the concept of filter NACs [10].

The derived operational rules of a TGG consist of consistency creating rules TR_{CC}, forward translation rules TR_{FT}, and backward translation rules TR_{BT} [12]. These rules contain additional Boolean valued translation attributes, which are used to mark the elements (nodes, edges or attributes) that are translated by the rules (TR_{FT} and TR_{BT}) or marked as consistent (TR_{CC}), respectively. The translation attributes are named tr_x, where x is the name of the respective element. See for example the rules in Fig. 7. Forward translation rules translate elements from the source domain and add the missing elements in the correspondence and target components according. Analogously, backward translation rules translate elements of the target domain and consistency creating rules mark the elements that are consistent.

A filter NAC specifies a context of a translation rule, which will always lead to an incomplete translation [10]. This means that applying the rule in any context containing the NAC pattern will require to backtrack this step. Such steps are avoided by the NAC.

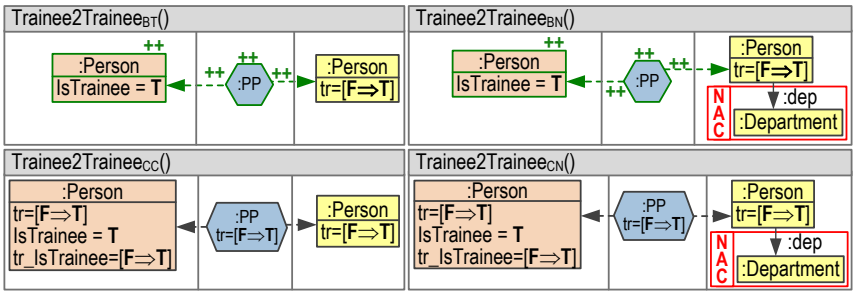

Fig. 7. Operational rules and extended operational rules for triple rule Trainee2Trainee

Example 3.1 (Addition of Filter NACs). The target model update $G^T \rightarrow G'^T$ in Fig. 1 creates the person Molly Murphy belonging to the marketing department. During the synchronisation process of this update, backtracking is necessary for the following reason. Fig. 4 shows the derived and the extended operational translation rules for triple rule Trainee2Trainee in Fig. 4. These rules use translation attributes (prefix `tr`) for flagging the elements that have been translated already. Backward translation rule Trainee2Trainee$_{BT}$ (top left in Fig. 7) is applicable to the new target model G'^T in the backward propagation phase of the update. But it leads to an incomplete translation, because the adjacent edge of type : dep will remain untranslated. Instead, if one of the two rules {Person2FirstMarketing$_{BT}$, Person2NextMarketing$_{BT}$} that are derived from the triple rules in Fig. 4 would be applied, then the problematic edge would be translated in the same step. The solution for eliminating this need for backtracking is to introduce a filter NAC yielding the new rule Trainee2Trainee$_{BN}$ (top right of Fig. 7). The NAC was obtained via the automated generation technique for filter NACs in [10]. It avoids an application to persons that belong to a marketing department. The new set of backward translation rules does not require backtracking any more, which we checked with the automated analysis of the tool AGG as described in [10].

The introduction of filter NACs causes a new problem for model synchronisation: the consistency creating rules TR_{CC} used for marking the already consistent elements are no longer compatible with the modified backward translation rules (see Ex. 3.2). Extending the consistency creating rules TR_{CC} by propagating the filter NACs solves the problem of incompatibility between TR_{CC} and TR_{BT}. Moreover, this does not introduce new incompatibilities with TR_{FT}, if each filter NAC is *domain specific*, i.e., it does not forbid structure on the source and target domains simultaneously.

Example 3.2 (Extension of Consistency Creating Rules). The consistency creating rule Trainee2Trainee$_{CC}$ (bottom left of Fig. 7) is used for marking consistent occurrences of a trainee in both domains. However, the rule can also be applied to full employees. In that case, the update propagation process will finally have to backtrack. Consider our running example illustrated in Fig. 1. Trainee Willy Wilson becomes a full employee. Therefore, he gets assigned to a department. The initial target update would be an addition of an edge of type : dep. Therefore, the consistency creating rules will still mark every element of the integrated model G to be consistent except the new edge. However, the backward translation cannot continue at this point, because the additional edge

cannot be translated separately by any rule. Hence, we introduce a corresponding NAC and derive the consistency creating rule $\mathsf{Trainee2Trainee}_{CN}$ (bottom right of Fig. 7).

Concept 3.3 (Extension of non-deterministic concurrent synchronisation framework with filter NACs). Let the sets TR_{FN} and TR_{BN} be derived from the operational triple rules TR_{FT} and TR_{BT} of a given TGG, respectively, by adding domain specific filter NACs to the rules. For each of these filter NACs we propagate the filter NACs to the rules in TR_{CC} leading to an updated set TR_{CN} of consistency creating rules. The formal definition of filter NAC propagation is given in [6]. Altogether we obtain a concurrent synchronization framework with filter NACs $\mathsf{CSync}(TGG, \mathsf{CSync}_{FN})$ based on the extended sets operational rules TR_{CN}, TR_{FN}, and TR_{BN}.

Our second main result (Thm. 3.4) shows that domain specific filter NACs do not affect the formal properties of correctness and completeness. Filter NACs concerning adjacent edges as in our example can be generated automatically [10]. Using the tool AGG, TGGs can be verified to ensure deterministic operations [12]. Moreover, conservative policies can be introduced to eliminate backtracking for attribute computations [12]. Thus, concurrent model synchronisation can be performed efficiently also in cases where the TGG operations initially require backtracking, but do not require backtracking using the generated filter NACs.

Theorem 3.4 (Correctness of Concurrent Synchronization Frameworks with Efficiency Improvement by Filter NACs). *Given a triple graph grammar TGG and a set of domain specific filter NACs for the operational translation rules that have been propagated to the consistency creating rules TR_{CC}. Then, the derived non-deterministic concurrent synchronisation framework with domain specific filter NACs $\mathsf{CSync}(TGG, \mathsf{CSync}_{FN})$ is correct and complete.*

Proof (Idea). The equivalence result of complete forward translation sequences with and without filter NACs in [10] has to be extended to the more complex case of forward and backward propagation. The precondition of domain specific filter NACs ensures that the propagation of filter NACs towards the consistency creating rules has no effect for further operational rules. For the full technical details and proof see [6]. □

4 Related Work

Triple Graph Grammars were introduced in [17] and since then have been applied successfully, among others, for (concurrent) model synchronisation [5,11,9] using the generated operations for bidirectional model transformations [18,10]. The (concurrent) model synchronisation approach we use in this paper is inspired by the symmetric delta lens (sd-lens) approach introduced in [2].

Several works focus on correctness properties and functional behaviour of the model synchronisation based on triple graph grammars [11,9,16]. In [4], a categorical merge construction for two conflicting model updates is given. In [9], a general synchronisation framework for concurrent model updates is given using the results from [4] for

resolving conflicts between concurrent model updates. In this work, we extend the concept of filter NACs, which were introduced in [10] for model transformation, to concurrent model transformation.

Xiong et. al. [19] introduce requirements, which shall hold for all bidirectional model updates, namely consistency, stability and preservation. These requirements are in close correspondence with the laws of correctness and identity ensured for the synchronisation framework in this present paper. The concurrent synchronisation framework in [19] requires conflict-free updates as input and is based on model difference approaches. Our approach does not require conflict freeness and occurring conflicts are resolved in a semi-automated way. Moreover, model updates are formalised as graph modifications, which are either available directly from the used modelling tool or can be derived automatically from model difference plugins.

In [15] an efficient control algorithm for bidirectional model transformation based on triple graph grammars is introduced and its correctness, completeness and efficiency is proven. The idea of the control algorithm is to determine dependencies of forward rules (and backward rules, respectively) in order to reduce non-deterministic behaviour in selecting the appropriate rule sequence for the transformation. This work is extended to model synchronisation in [16]. In both works, this algorithm either provides correct models as result or an error. In our approach, conflicts do not lead to errors, but are made explicit and solved semi-automatically via operations (operation Res, CCS and CCT).

5 Conclusion

In this paper, we have shown how concurrent modifications in source and target models that are linked by a TGG can be synchronised. More precisely, we have first introduced a non-deterministic concurrent synchronisation framework generalising the existing approach [9] to arbitrary TGGs. Furthermore, we have used filter NACs to improve the efficiency of the forward and backward propagations avoiding backtracking in cases, where parts of the models would remain untranslated.

We already applied TGGs and Henshin [1,8] in a large-scale industrial project with the satellite operator SES for the translation of satellite control procedures between different programming languages [13]. The satellite Astra 2F is the first satellite running on the translated software and is operational in space since 2012. In future work, we will apply the presented concepts to synchronisation case studies in this field. Particularly, we apply TGGs to visualise and generate source code of satellite procedures at SES.

Acknowledgements. Supported by the Fonds National de la Recherche, Luxembourg (3968135, 4895603).

References

1. Arendt, T., Biermann, E., Jurack, S., Krause, C., Taentzer, G.: Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations. In: Proc. MoDELS'10. LNCS, vol. 6394, pp. 121–135. Springer (2010)

2. Diskin, Z., Xiong, Y., Czarnecki, K.: From State- to Delta-Based Bidirectional Model Transformations. In: Proc. ICMT'12, LNCS, vol. 6142, pp. 61–76. Springer (2010)
3. Ehrig, H., Ehrig, K., Ermel, C., Hermann, F., Taentzer, G.: Information Preserving Bidirectional Model Transformations. In: Proc. FASE'07. LNCS, vol. 4422, pp. 72–86. Springer (2007)
4. Ehrig, H., Ermel, C., Taentzer, G.: A Formal Resolution Strategy for Operation-Based Conflicts in Model Versioning Using Graph Modifications. In: Proc. FASE'11. LNCS, vol. 6603, pp. 202–216. Springer (2011)
5. Giese, H., Wagner, R.: From model transformation to incremental bidirectional model synchronization. SoSyM 8, 21–43 (2009)
6. Gottmann, S., Hermann, F., Nachtigall, N., Braatz, B., Ermel, C., Ehrig, H., Engel, T.: Correctness of Generalisation and Customisation of Concurrent Model Synchronisation Based on Triple Graph Grammars. Tech. Rep. 2013-08, Technische Universität Berlin, Fak. IV (2013), http://www.eecs.tu-berlin.de/menue/research/technical_reports/parameter/en/
7. Greenyer, J., Kindler, E.: Comparing relational model transformation technologies: implementing Query/View/Transformation with Triple Graph Grammars. SoSyM 9, 21–46 (2010)
8. EMF Henshin – Version 0.9.6 (2013), http://www.eclipse.org/henshin/
9. Hermann, F., Ehrig, H., Ermel, C., Orejas, F.: Concurrent Model Synchronization with Conflict Resolution Based on Triple Graph Grammars. In: Proc. FASE'12. LNCS, vol. 7212, pp. 178–193. Springer (2012)
10. Hermann, F., Ehrig, H., Golas, U., Orejas, F.: Efficient Analysis and Execution of Correct and Complete Model Transformations Based on Triple Graph Grammars. In: Proc. MDI'10. pp. 22–31. MDI '10, ACM (2010)
11. Hermann, F., Ehrig, H., Orejas, F., Czarnecki, K., Diskin, Z., Xiong, Y.: Correctness of Model Synchronization Based on Triple Graph Grammars. In: Proc. MoDELS'11. LNCS, vol. 6981, pp. 668–682. Springer (2011)
12. Hermann, F., Ehrig, H., Orejas, F., Czarnecki, K., Diskin, Z., Xiong, Y., Gottmann, S., Engel, T.: Model synchronization based on triple graph grammars: correctness, completeness and invertibility. SoSyM pp. 1–29 (2013)
13. Hermann, F., Gottmann, S., Nachtigall, N., Braatz, B., Morelli, G., Pierre, A., Engel, T.: On an Automated Translation of Satellite Procedures Using Triple Graph Grammars. In: Proc. ICMT'13, LNCS, vol. 7909, pp. 50–51. Springer (2013)
14. Kindler, E., Wagner, R.: Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios. Tech. Rep. TR-ri-07-284, Department of Computer Science, University of Paderborn, Germany (2007)
15. Lauder, M., Anjorin, A., Varró, G., Schürr, A.: Bidirectional Model Transformation with Precedence Triple Graph Grammars. In: Proc. ECMFA'12. LNCS, vol. 7349, pp. 287–302. Springer (2012)
16. Lauder, M., Anjorin, A., Varró, G., Schürr, A.: Efficient Model Synchronization with Precedence Triple Graph Grammars. In: Proc. ICGT'12. LNCS, vol. 7562, pp. 401–415. Springer (2012)
17. Schürr, A.: Specification of Graph Translators with Triple Graph Grammars. In: Proc. WG'94. LNCS, vol. 903, pp. 151–163. Springer (1994)
18. Schürr, A., Klar, F.: 15 Years of Triple Graph Grammars. In: Proc. ICGT'08. pp. 411–425. No. 5214 in LNCS, Springer (2008)
19. Xiong, Y., Song, H., Hu, Z., Takeichi, M.: Synchronizing concurrent model updates based on bidirectional transformation. SoSyM 12, 89–104 (2013)