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)

What Algebraic Graph Transformations Can Do For Model Transformations

Gabriele Taentzer

10 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

What Algebraic Graph Transformations Can Do For Model Transformations

Gabriele Taentzer

Philipps-Universität Marburg
Germany

Abstract: Model transformations are key activities in model-driven development (MDD). A number of model transformation approaches have emerged for different purposes and with different backgrounds. This paper focusses on the use of algebraic graph transformation concepts to specify and verify model transformations in MDD.

Keywords: model transformation, graph transformation

1 Introduction

Model transformations play a central role in model-driven software development. They are used e.g. to refactor models, to analyze them, to translate them to intermediate models, and to generate code. We distinguish endogenous transformations taking place within one modeling language from exogenous ones which are translations between modeling languages [CH06].

Model-to-model transformations are usually distinguished from model-to-text transformations. (Compare also Eclipse modeling projects at [EMP].) Why does this distinction exist? While model-to-model transformation approaches such as QVT [QVT], ATL [EMP] and graph transformation-based approaches [SNZ08] like Atom3 [LV02], Fujaba [FNTZ98], Henshin [ABJ+10], GrGen [GBG+06], MOFLON [AKK+08], Viatra2 [VB07] and VMTS [LLMC05], transform models based on their underlying syntax structure only, model-to-text transformations performed by tools such as Jet and Velocity, are usually mixed approaches. The abstract syntax of an input model is transformed to some text in concrete syntax, often program text. These approaches are usually based on templates which can be considered as "clozes" where gaps are filled with information coming from the input model. I.e. these transformations are often performed in a weakly structured and untyped manner. No guarantees are given that the resulting text is syntactically correct. To better cope with this situation, Wachsmuth [Wac09] has equipped attribute grammars with template-based model transformations. Thus, model transformations are defined based on both, the concrete and abstract syntax definition. In contrast, model-to-model transformations are usually based on abstract syntax structures only.

Models are often considered to be visual, although this is not an inherent property of models. It is very natural to consider the underlying structure of a visual model as a graph. In case of the Eclipse Modeling Framework [EMF] which has developed to a quasi-standard modeling technology, the underlying structure of an EMF model can be considered as a graph with a spanning tree (or spanning forest, i.e. several spanning trees) established by containment relations.

In the following, we discuss the specification of model transformations by graph transformations and especially, by algebraic graph transformations, in contrast to other transformation

approaches. Since model transformations can be considered as special programs specified by a formal transformation approach, there is the basic hope that their correctness can be verified. We consider interesting properties for model transformation and discuss techniques for showing them. Finally, we give an outlook on the future of model transformations and sketch which role algebraic graph transformations can play in future.

2 Specification of model transformations

A variety of specification paradigms have been applied to model-to-model transformations such as object-oriented, rule-based, constraint-based, and imperative concepts. See [CH06] for an overview on various model transformation approaches following these paradigms purely or in combination. In the following, we highlight some features of the rule and graph-based definition of model transformations using graph transformation concepts and especially, the algebraic approach.

Basic Features. Graph transformation is a rule-based technique which can be a great advantage for using them to specify model transformations. Instead of explicitly programming each model navigation and each transformation action, model transformation can be specified at a higher level of abstraction. Rules define if-then patterns which allow a transformation specification being closer to the model domain it is applied to than usual programming. E.g. to perform a rule-based transformation a pre-defined transformation engine supporting rule matching and application is used. Therefore, the implementation of the matching algorithm is hidden in this engine. Moreover, the application order of rules needs not necessarily be specified, but may be specified if necessary, in total or just partially.

Considering the transformation of visual models, graph transformation seems to be a natural choice to manipulate their underlying graph structures. A well-defined approach such as the algebraic graph transformation [EEPT06] guarantees that the underlying structure of the resulting model is again a graph and thus, is structure consistent (i.e. no dangling edges occur). Moreover, the resulting model has to be correctly typed. Typed algebraic graph transformations have been shown to always lead to well-typed transformation results. To support a natural mapping of meta modeling concepts, the concept of node type inheritance has been developed for algebraic graph transformation.

EMF model transformations can be defined as a special kind of graph transformations not destroying the spanning tree (forest) property of models. Thus, EMF model transformations have to fulfill additional structural properties. Algebraic graph transformation can be used to ensure these additional properties [BET08]. A related model transformation approach which formalizes EMF models is Moment2 [BM09] being based on rewriting logic as implemented in Maude.

Advanced transformation concepts. Model transformation approaches based on graph transformation concepts are all rule-based techniques, but differ in the way rules are applied and how non-determinism in rule application is reduced or even eliminated. An early comparison of graph transformation approaches is presented in [AEH+99]. Later, several graph transformation tools

have been applied and compared wrt. one and the same model transformation case in [TEG+05]. In the following, we do not compare basic features but highlight some advanced transformation concepts.

Most graph transformation approaches allow to specify negative application conditions for rules, originally introduced in [HHT96]. More complex conditions can be specified by e.g. graph patterns in Viatra and nested conditions in Henshin. Furthermore, the order of rule application can be determined by specifying a control flow using e.g. story diagrams in Fujaba, activity diagrams in VMTS, ASM programs in Viatra, and transformation units in Henshin. It is up to future work to compare these control mechanisms. For better reuse of transformation parts, generic and meta-transformations can be defined in Viatra. For handling collections of flexible size such as the set of all features belonging to a class, a *forall* construct is needed which is e.g. offered by Viatra directly. In the algebraic context, we use amalgamated graph transformation where a kernel rule is applied exactly once and multi-rules which contain the kernel rule as subrule are applied as often as possible. All their applications overlap in the kernel rule application. Usually, each multi-rule application covers one collection element. As shown in [BET09], this concept can also be extended to EMF model transformation in a straightforward way.

Kinds of model transformations. Graph transformation can be used to specify endogenous as well as exogenous model transformations. Moreover, thanks to triple graph grammars [Sch94] they are also useful to specify model integrations and model synchronizations.

The well-known double-pushout approach to graph transformation [EEPT06] can be interpreted as a kind of in-place transformation where new parts are directly integrated into the existing graph. In addition, it is also allowed to delete existing graph parts from the given graph. To keep track with graph manipulations, the formal definition distinguishes an original graph from a resulting one. A partial graph morphism in between precisely defines the relation between both graphs. In-place transformations are well suited to specify model refactorings which transform models such that their structures improve while their semantics is preserved. See e.g. [MTR07, RLK+08] for the specification of model refactorings by algebraic graph transformation concepts. A model transformation tool which is based on algebraic graph transformation concepts is Henshin [ABJ+10], an Eclipse plug-in for model transformation based on the Eclipse Modeling Framework (EMF).

In contrast to in-place transformation, out-place model transformations consider source models as input models which are read but not altered, while target models are built up without using them for checking pre-conditions of transformations. A graph transformation approach which is mainly dedicated to this kind of exogenous model transformation is e.g. VMTS.

A famous approach, not only for exogenous model transformations but also for model integrations and synchronizations, are triple graph grammars (TGGs) [Sch94, KS06]. They distinguish three graphs, namely a source graph, a target graph, and a correspondence graph which is mapped to each of the other two for establishing a correspondence relation between source and target graphs. A comprehensive implementation of TGGs is given by MOFLON [AKRS06] using a declarative QVT-like language based on Fujaba. It does not only support model-to-model transformations, but also model-to-text transformations using a template-based code generator which supports XSLT and Velocity. Recently, Ehrig et al. started to formalize and to extend the

main concepts of TGGs by algebraic graph transformation concepts in order to provide a clear formal basis to characterize consistent transformation. (Compare e.g. [EEHP09].)

3 Verification of model transformations

Since model transformations are reused heavily in MDD, they should be of high quality. It is common practice to extensively test model transformations. Their verification is still in its infancy. In the following, we discuss which properties are interesting to be verified and spot on first results.

Structure and type consistency. As pointed out in the previous section, a basic property of model transformations is that resulting models are structure and type consistent. Transformation approaches such as algebraic graph transformation guarantee this property already automatically by definition without any additional verification effort. (Compare [EEPT06].)

Functional behavior. Furthermore, model transformations shall terminate. Termination is an issue for all model transformations which may contain loops. Imperative model transformation approaches offer explicit loops as control constructs, while rule-based model transformations such as those defined by graph transformation concepts can contain implicit loops occurring from the application of rules as long as applicable. Termination cannot always be shown, since model transformation approaches are usually Turing-complete. Thus, we are confronted with the halting problem in general. But there are first approaches to define sufficient termination conditions for model transformation systems, especially for algebraic graph transformation systems (see e.g. [EEPT06, VVE+06, BH10]).

The uniqueness of transformation results is another general property of model transformations to be discussed. In more detail, we like to check if the transformation result is unique up to isomorphism for a given input model. It depends on the kind of model transformation, if this strict property is needed. If e.g. a code generator provided two different programs for one and the same input model, we would expect that they both are semantically equivalent wrt. the input model. (I.e. both programs exhibit the same observable behavior when being executed.) However, since semantical equivalence is often difficult to show, proving the uniqueness of transformation results is, although more strict, more practicable. Since rule-based transformations may use an implicit application control, the uniqueness of transformations is especially important for them.

Algebraic graph transformation offers a rich theory to show that transformation systems are confluent, i.e. yield unique results only. This theory is based on a result for general rewriting systems which states that a rewrite system which is locally confluent and terminating, is confluent in general. Local confluence of graph transformations can be shown based on critical pairs, an approach which has been lifted from term rewriting systems to graph transformation systems [Plu94, EEPT06]. A critical pair shows a conflicting situation in a minimal context. If all critical pairs can be shown to be confluent, the complete transformation system is locally confluent. If critical pairs cannot be shown to be confluent, a potential conflict is reported. Usual strategies to resolve or to even avoid conflicts from the beginning reduce concurrency and enforce rule applications in fixed orders.

Syntactical and semantical correctness. Additionally to these pretty general properties, transformation results also need to be elements of target languages. This means that they have to be syntactically and semantically correct. The syntax of modeling languages is usually given by meta models [MOF], i.e. a resulting model has to be type consistent and has to obey all additional well-formedness rules of the meta model. As already pointed out, algebraic graph transformations guarantee type consistent results. Well-formedness rules which can be expressed by graph constraints [HP09] can be used as post-conditions to check transformation results. Furthermore, there is a technique to translate graph constraints to application conditions of transformation rules, i.e. to translate them to pre-conditions of transformation rules. These new pre-conditions have to compared with already existing ones to find out if the original transformation system leads to syntactically correct models only. If not, additional pre-conditions have to be taken into account to improve the transformation system. That way algebraic graph transformations provide us with an automatic and efficient procedure to check models for syntactical correctness.

Semantical correctness of transformation results is more difficult to verify. We distinguish between the static and dynamic analysis of transformation results. Static analyses of properties which can be specified by additional constraints can be performed analogously to the check for syntactical correctness. Further recent approaches apply model checking to graph transformations e.g. [RSV04] or start using a theorem prover as presented in [Str08] and [GGL+06]. To analyze the dynamic semantics of transformation results we assume that dynamic semantics have been defined for source and target languages. Let us assume that each langauge has an operational semantics given by an algebraic graph transformation and the source semantic rules have been translated to corresponding target semantic rules. In that case, Ehrig et al. have shown semantical correctness for behavior models in [EE08] by proving that each execution step in the source model corresponds to an execution step in the target model. For the special case of model refactorings, semantics preservation can be shown by bisimilarity of the original and the resulting model after a model refactoring step using the borrowed context technique (see [HHK10]).

Verification of bidirectional model transformations. In [Ste08], Stevens argues that bidirectional model transformations should have the following properties: (1) correctness meaning that forward and backward transformations have to enforce consistency between source and target models, (2) hippocraticness ensuring that forward and backward transformations are applied only if the consistency relation is not established, and (3) undoability meaning that a model change which is immediately undone leads again to a consistent model pair.

For triple graph grammars, the consistency relation between source and target models is specified by triple rules. In [EEHP09], the equivalence of triple transformations and their corresponding forward and backward transformations is investigated. Thus, conditions for the correctness of forward and backward transformations are elaborated. Furthermore, results concerning completeness and termination of forward and backward transformations, always wrt. their triple transformations, are considered. A forward transformation is called complete if each model of the source language can be transformed to a model of the target language, and vice versa. It is up to future work to develop further results which are concerned with hippocraticness and undoability as stated above.

We presented essential verification techniques for model transformations here and sketched

how they can be verified based on algebraic graph transformation. It is up to future work, to perform an in-depth comparison of different verification techniques for model transformations. Of course, we have to keep in mind that only a few model transformation approaches are formally defined such that verifications can be performed.

4 Outlook

Model transformations form an interesting research field with a lot of new research problems to be solved. In addition future work already mentioned, we spot on a selection of topics in the following where the application of algebraic graph transformation concepts seems to be very promising:

The application of algebraic graph transformation concepts to EMF model transformation as presented in [BET08] shows that formal concepts can be well applied in a practical setting. Furthermore, we showed the important property that resulting EMF models are structure and type consistent by construction. We expect that the rich theory of algebraic graph transformation can be easily adapted to this kind of EMF model transformations leading to interesting verification techniques for EMF model transformations.

As stated above, distinct transformation approaches have emerged for model-to-model and model-to-text transformations. We can observe a recent trend where model-to-model transformation approaches such as ATL are also applied to model-to-text transformations. This means that transformation results, being texts in this case, are computed on the basis of abstract syntax structures. Approaches like JaMoPP [HJSW09] for example, define meta models for Java and provide model parsers and printers. Thus having a meta model for a textual language at hand, model-to-model transformation approaches and especially graph transformation approaches can be applied guaranteeing well-structured and well-typed transformation results also for model-to-text transformations. Moreover, verification of model-to-text transformations concerning interesting properties would become possible. It is up to future work to test and probably improve the efficiency of model-to-model transformation implementations compared to model-to-text transformations. Especially a comparison with extended model-to-text transformation approaches such as the one by Wachsmuth [Wac09] would be interesting.

Last but not least, we can observe that the deployment of model transformation techniques in software engineering increases and more complex forms of model transformations are needed where not only one input and one output model are considered but a number of models are involved. See e.g. multi-directional model transformations in [KS06]. Example scenarios are coherent refactorings of several heterogeneous models and/or code, code generation from several interrelated models such as those in the Graphical Modeling Framework [GMF] yielding a number of code files and model weaving.

To conclude, algebraic graph transformation seems to be a promising formal basis for different kinds of model transformations. It is especially promising to be used for specifying and reasoning about larger networks of model transformations, since it is based on category theory which provides us with rigorous structuring concepts.

Bibliography

[ABJ+10] T. Arendt, E. Biermann, S. Jurack, C. Krause, G. Taentzer. Henshin: Advanced Concepts and tools for In-Place EMF Model Transformation. In *Model Driven Engineering Languages and Systems, 13th International Conference, MoDELS 2010, Oslo, Norway. Proceedings*. LNCS 6394, pp. 121–135. Springer, 2010. To appear.

[AEH+99] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Plump, A. Schürr, G. Taentzer. Graph transformation for specification and programming. *Sci. Comput. Program.* 34(1):1–54, 1999. doi:http://dx.doi.org/10.1016/S0167-6423(98)00023-9

[AKK+08] C. Amelunxen, F. Klar, A. Königs, T. Rötschke, A. Schürr. Metamodel-based tool integration with moflon. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*. Pp. 807–810. ACM, New York, NY, USA, 2008. doi:http://doi.acm.org/10.1145/1368088.1368206

[AKRS06] C. Amelunxen, A. Königs, T. Rötschke, A. Schürr. MOFLON: A Standard Compliant Metamodeling Framework with Graph Transformations. In *Model Driven Architecture - Foundations and Applications: Second European Conference*. Pp. 361–375. Springer Verlag, LNCS 4066, 2006.

[BET08] E. Biermann, C. Ermel, G. Taentzer. Precise Semantics of EMF Model Transformations by Graph Transformation. In Czarnecki et al. (eds.), *Model Driven Engineering Languages and Systems, 11th International Conference, MoDELS 2008, Toulouse, France, September 28 - October 3, 2008. Proceedings*. Lecture Notes in Computer Science 5301, pp. 53–67. Springer, 2008.

[BET09] E. Biermann, C. Ermel, G. Taentzer. Lifting Parallel Graph Transformation Concepts to Model Transformation based on the Eclipse Modeling Framework. *ECEASST* 26, 2009.

[BH10] D. Bisztray, R. Heckel. Combining Termination Criteria by Isolating Deletion. In *Graph Transformations - 5th International Conference, ICGT 2010, Enschede, The Netherlands, September 27 - October 2, 2010. Proceedings*. LNCS 6372, pp. 203–217. Springer, 2010.

[BM09] A. Boronat, J. Meseguer. MOMENT2: EMF Model Transformations in Maude. In *XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2009), San Sebastián, Spain, September 8-11, 2009*. Pp. 178–179. 2009.

[CH06] K. Czarnecki, S. Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal* 45(3):621–646, 2006.

[EE08] H. Ehrig, C. Ermel. Semantical Correctness and Completeness of Model Transformations using Graph and Rule Transformation. In Ehrig et al. (eds.), *Proc. International Conference on Graph Transformation (ICGT'08)*. LNCS 5214, pp. 194–210. Springer Verlag, Heidelberg, 2008.

[EEHP09] H. Ehrig, C. Ermel, F. Hermann, U. Prange. On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars. In Schürr and Selic (eds.), *ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09)*. Volume 5795, pp. 241–255. Springer LNCS, 2009.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[EMF] EMF. Eclipse Modeling Framework. http://www.eclipse.com/emf.

[EMP] EMP. Eclipse Modeling Project. http://www.eclipse.org/modeling/.

[FNTZ98] T. Fischer, J. Niere, L. Torunski, A. Zündorf. Story Diagrams: A new Graph Rewrite Language based on the Unified Modeling Language. In Engels and Rozenberg (eds.), *Proc. of the 6^{th} Int. Workshop on Theory and Application of Graph Transformation*. LNCS 1764, pp. 296–309. Springer, November 1998.

[GBG+06] R. Geiß, G. V. Batz, D. Grund, S. Hack, A. Szalkowski. GrGen: A Fast SPO-Based Graph Rewriting Tool. In *Graph Transformations, Third International Conference, ICGT 2006, Natal, Rio Grande do Norte, Brazil, September 17-23, 2006, Proceedings*. LNCS 4178, pp. 383–397. Springer, 2006.

[GGL+06] H. Giese, S. Glesner, J. Leitner, W. Schäfer, R. Wagner. Towards Verified Model Transformations. In *In Proc. of MODEVA workshop associated to MODELS'06*. Pp. 78–93. Le Commissariat l'Energie Atomique - CEA, 2006.

[GMF] GMF. Graphical Modeling Framework. http://www.eclipse.com/gmf.

[HHK10] F. Herrmann, M. Hülsbusch, B. König. Specification and Verification of Model Transformations. *ECEASST* 30, 2010. In this volume.

[HHT96] A. Habel, R. Heckel, G. Taentzer. Graph Grammars with Negative Application Conditions. *Fundam. Inform.* 26(3/4):287–313, 1996.

[HJSW09] F. Heidenreich, J. Johannes, M. Seifert, C. Wende. JaMoPP: The Java Model Parser and Printer. Technical report TUD-FI09-10, Technical University of Dresden, Institut für Software- und Multimediatechnik, 2009. Technical Report.

[HP09] A. Habel, K.-H. Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science* 19:1 – 52, 2009.

[KS06] A. Königs, A. Schürr. Tool Integration with Triple Graph Grammars - A Survey. *Electronic Notes in Theoretical Computer Science 148, 113-150*, 2006.

[LLMC05] T. Levendovszky, L. Lengyel, G. Mezei, H. Charaf. A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS. *Electron. Notes Theor. Comput. Sci.* 127(1):65–75, 2005. doi:http://dx.doi.org/10.1016/j.entcs.2004.12.040

[LV02] J. de Lara, H. Vangheluwe. AToM^3: A Tool for Multi-formalism and Metamodelling. In *Fundamental Approaches to Software Engineering, 5th International Conference, FASE 2002, held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002, Grenoble, France, April 8-12, 2002, Proceedings*. LNCS 2306, pp. 174–188. Springer, 2002.

[MOF] MOF. Meta Object Facility (MOF) Core. URL: http://www.omg.org/spec/MOF.

[MTR07] T. Mens, G. Taentzer, O. Runge. Analysing refactoring dependencies using graph transformation. *Software and System Modeling* 6(3):269–285, 2007.

[Plu94] D. Plump. Critical Pairs in Term Graph Rewriting. In *Mathematical Foundations of Computer Science 1994, 19th International Symposium, MFCS'94, Kosice, Slovakia, August 22 - 26, 1994, Proceedings*. LNCS 841, pp. 556–566. Springer, 1994.

[QVT] QVT. MOF 2.0 Query / Views / Transformation (QVT). URL: http://www.omg.org/spec/QVT.

[RLK+08] G. Rangel, L. Lambers, B. König, H. Ehrig, P. Baldan. Behavior Preservation in Model Refactoring Using DPO Transformations with Borrowed Contexts. In *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*. Lecture Notes in Computer Science 5214, pp. 242–256. Springer, 2008.

[RSV04] A. Rensink, Schmidt, D. Varró. Model Checking Graph Transformations: A Comparison of Two Approaches. In Ehrig et al. (eds.), *International Conference on Graph Transformations (ICGT)*. Lecture Notes in Computer Science 3256, pp. 226–241. Springer Verlag, Berlin, 2004.

[Sch94] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In Mayr et al. (eds.), *Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, Herrsching, Germany, June 16-18, 1994, Proceedings*. Lecture Notes in Computer Science 903, pp. 151–163. Springer, 1994.

[SNZ08] A. Schürr, M. Nagl, A. Zündorf (eds.). *Applications of Graph Transformations with Industrial Relevance, Third International Symposium, AGTIVE 2007, Kassel, Germany, October 10-12, 2007, Revised Selected and Invited Papers*. Lecture Notes in Computer Science 5088. Springer, 2008.

[Ste08] P. Stevens. Towards an Algebraic Theory of Bidirectional Transformations. In *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*. Lecture Notes in Computer Science 5214, pp. 1–17. Springer, 2008.

[Str08] M. Strecker. Modeling and Verifying Graph Transformations in Proof Assistants. *Electron. Notes Theor. Comput. Sci.* 203(1):135–148, 2008. doi:http://dx.doi.org/10.1016/j.entcs.2008.03.039

[TEG+05] G. Taentzer, K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovsky, U. Prange, D. Varro, S. Varro-Gyapay (eds.). *Model Transformation by Graph Transformation: A Comparative Study*. 2005. http://sosym.dcs.kcl.ac.uk/events/mtip05.

[VB07] D. Varró, A. Balogh. The model transformation language of the VIATRA2 framework. *Sci. Comput. Program.* 68(3):214–234, 2007.

[VVE+06] D. Varró, S. Varró-Gyapay, H. Ehrig, U. Prange, G. Taentzer. Termination Analysis of Model Transformations by Petri Nets. In *Graph Transformations, Third International Conference, ICGT 2006, Natal, Rio Grande do Norte, Brazil, September 17-23, 2006, Proceedings*. LNCS 4178, pp. 260–274. Springer, 2006.

[Wac09] G. Wachsmuth. A Formal Way from Text to Code Templates. In *Fundamental Approaches to Software Engineering, 12th International Conference, FASE 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*. LNCS 5503, pp. 109–123. Springer, 2009.