Towards Establishing Consistency between Graph Transformation Rules and Atomic Graph Constraints Using Multi-Amalgamation

Jens Kosiol[1], Lars Fritsche[2], Nebras Nassar[1],
Andy Schürr[2], and Gabriele Taentzer[1]

[1] Philipps-Universität Marburg
{kosiolje,nassarn,taentzer}@mathematik.uni-marburg.de
[2] TU Darmstadt
{lars.fritsche,andy.schuerr}@es.tu-darmstadt.de

In this paper, we report about work in progress in which we develop a new kind of correct-by-construction transformations in the setting of algebraic graph transformation. The theory of algebraic graph transformation [1] has proved to be a suitable formal framework to reason about model transformations. Meta-models are formalized as type graphs and instances as typed graphs. The application of a transformation is given by two pushouts. Nested conditions and constraints [3] allow to express (first-order) properties of typed graphs or morphisms which are not expressible by typing alone. In many application scenarios, like instance generation and editing or refactoring of instances, it is desirable that transformations preserve consistency w. r. t. constraints. Given a constraint c and a rule r, Habel and Pennemann [3] developed the construction of so-called *c-guaranteeing* and *c-preserving rules*. Both constructions equip the rule r with an application condition that ensures that the updated rule is only applicable in such a way that the constraint c is fulfilled after application. The application condition of the c-preserving rule is logically weaker since it assumes the constraint c to hold before application of the rule. The basic idea behind their approach is to compute all essentially different ways in which the rule r might be applied such that the constraint c holds after application and requiring one of them to hold. But a rule may be rendered inapplicable by this construction: If there is no way to apply a rule r such that the constraint c is fulfilled after application, it gets equipped with the application condition `false`, meaning that the rule is not applicable any longer. Depending on the application scenario, this may constitute an undesirable restriction of the language defined by the given set of rules.

To supplement the just described approach, we developed an alternative construction for special kinds of rules and constraints. Our idea is to complement the action of a rule r in such a way that a given constraint c holds after application of the rule instead of calculating an application condition. Since we intend to not alter but complement the action of a rule, we restrict ourselves to the cases of monotonic rules, i. e., rules which only create structure, and atomic constraints as defined in [1]. Some instance G satisfies such an atomic constraint c – which may be compactly notated as $\forall(P, \exists C)$ where P is a subobject of C – if for all subobjects of G that are isomorphic to P there exists a subobject isomorphic to C which includes the image of P. The crucial idea of our approach is

to calculate all possible ways in which the application of a monotonic rule r may lead to a new match for the precondition P of a constraint $\forall(P, \exists C)$. These are the different ways in which application of the rule can introduce a new violation of the constraint. They can significantly differ from each other and thus require diverging actions to resolve the would-be introduced violation of the constraint. Hence, for each such situation we derive a rule that includes the original rule r as subrule but additionally creates structure that complements the new match for P to a new match for C. All these rules are collected into a so-called *interaction scheme* [2]. Applying an interaction scheme means to apply their common subrule – here, the original rule r – once and every other rule from the interaction scheme as often as possible but with fixed partial match given by the match of the subrule. The arising rule is called *multi-amalgamated*. In this way, every image for P that gets newly created by an application of the rule r is complemented to an image of C by application of one of the rules of the interaction scheme. Formalizing two variants of this construction in the setting of adhesive categories and proving that they guarantee or, respectively, preserve a constraint c under certain circumstances is the main contribution of this paper.

Afterwards we discuss some possibilities to refine our technique, namely inclusion of application conditions, support of more general kinds of rules, and (limited) support for situations in which recursion may occur, i. e., in situations where complementing an image for P to one of C in an instance G may lead to an additional image of P. To examine the practicability of our approach, we plan to implement and use it to automatically derive more complex edit-operations out of given simple ones and to automatically design multi-amalgamated triple graph rules [4] out of simple ones (which are always monotonic).

Acknowledgments This work was partially funded by the German Research Foundation (DFG), projects "Triple Graph Grammars (TGG) 2.0" and "Generating Development Environments for Modeling Languages".

References

[1] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. *Fundamentals of algebraic graph transformation*. Monographs in Theoretical Computer Science. Berlin, Heidelberg, and New York: Springer, 2006.

[2] Ulrike Golas, Annegret Habel, and Hartmut Ehrig. "Multi-amalgamation of rules with application conditions in M-adhesive categories". In: *Math. Struct. in Comp. Science* 24 (4 2014).

[3] Annegret Habel and Karl-Heinz Pennemann. "Correctness of high-level transformation systems relative to nested conditions". In: *Math. Struct. in Comp. Science* 19 (2 2009), pp. 245–296.

[4] Erhan Leblebici, Anthony Anjorin, Andy Schürr, and Gabriele Taentzer. "Multi-amalgamated Triple Graph Grammars". In: *Proc. of ICGT '15*. Ed. by Francesco Parisi-Presicce and Bernhard Westfechtel. Cham: Springer, 2015, pp. 87–103.