The Advantage of Using Co-span Graph Transformations for Meta-model Evolution*

Florian Mantz[1] and Uwe Wolter[2]

[1] Bergen University College
fma@hib.no
[2] University of Bergen
uwe.wolter@ii.uib.no

Model-driven engineering [5] (MDE) is a software engineering discipline which employs models as primary artifacts in the software development process. Software (parts) are specified using modeling languages at an high abstraction level. Model-transformations are used to automate recurring development tasks as well as to generate software artifacts for different runtime environments and testing. This can improve productivity of developers as well as quality and cost-effectiveness of software. However, to truly gain benefits from MDE it is important that used modeling languages suit well to their modeling purposes. Therefore, domain-specific modeling languages designed for specific tasks that are continuously improved have become popular. However, the evolution of modeling languages introduce a new challenge developers struggle in practice with, existing models need to be migrated after the corresponding modeling language has been evolved (see. Fig. 1). Since the manual migration of models is tedious and error-prone, different approaches have been developed that (partially) automate this task.

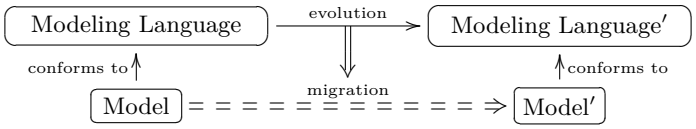

Figure 1: Model co-evolution: Modeling language evolution and model migration

In our work, we focus on the formalization of this co-evolution challenge using category theory [1]. In particular, we use algebraic graph transformations [3] to describe modeling language evolution and model migration as sequences of rule applications. Modeling languages are specified in meta-models. In this paper, we focus on the meta-model evolution task and refer to our earlier work [8, 7, 10] for the model-migration task. In algebraic graph transformation, it is prevalent to describe transformation rules as spans, while it is also possible to use co-spans [4, 6]. In our formalization we are using co-spans and in this paper we argue why co-span rules suit better to the meta-model evolution task. Additional reasons why co-spans rules also better fit to the model migration challenge can be found in [10]. Figure 2 shows a graph transformation using a span rule while Fig. 3 shows a graph transformation using a co-span rule. A graph transformation is applied by constructing either two pushouts (double pushout approach[3]) or one pushout and one pullback (sesqui pushout approach [2]).

In the following, we will explain why co-span rules suit better to the meta-model evolution challenge. In MDE, model migration approaches can be categorized into three different kind of approaches [9]:

1. *Manual specification approaches:* model migrations have to be specified manually while there is some support to migrate unchanged model parts automatically.

*This work was partially funded by NFR project 194521 (FORMGRID)

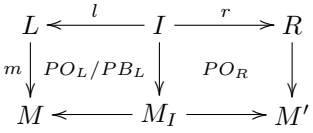

Figure 2: Graph transformation span approach

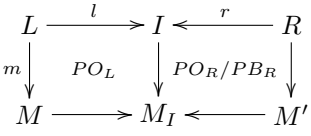

Figure 3: Graph transformation co-span approach

2. *Operator based approaches:* coupled evolution/migration operators are used to evolve the meta-model and migrate models correspondingly.
3. *Matching approaches:* a sequence of meta-model evolution steps is detected automatically and models are migrated correspondingly.

While the first kind of approaches (manual specification) are uninteresting for this paper, describing meta-model evolution as a sequence of rule applications fits obviously well to the second kind of approaches (operator based). However, using co-span rules instead of span rules to describe meta-model evolution steps helps also to formalize the third kind of approaches (matching) as we explain next. Therefore consider Fig. 4 and Fig. 5 first: both figures show a meta-model evolution of a meta-model M_1 to M_7 described by a sequence of three rule-applications using span rules in Fig. 4, respectively co-span rules in Fig. 5. In both cases, the first (M_1) and last meta-model (M_7) can be related by a span respectively co-span by composing pullbacks, respectively pushouts as shown in the example figures.

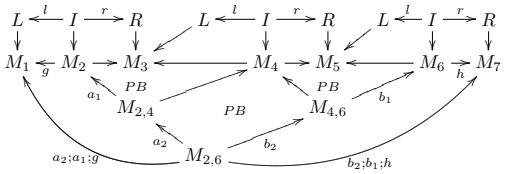

Figure 4: Meta-model span composition

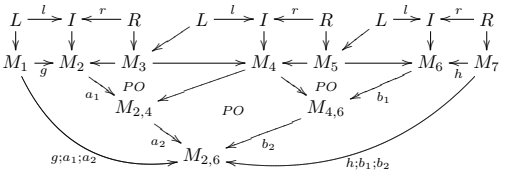

Figure 5: Meta-model co-span composition

Figure 6 shows a concrete example for meta-model composition with co-spans. A sequence of three rules "Move Attribute", "Merge Class", "Add Container" has been applied to a simple meta-model. Only the meta-models are shown and their compositions by pushouts. By composing morphisms meta-model M_1 and M_7 can be related by co-span $M_1 \to M_{2,6} \leftarrow M_7$. Looking at this meta-model co-span it seems to be feasible that a tool could create it without knowing any possible rule application sequence. A tool can analyze the ids of the meta-model elements instead. (We assume that ids of meta-model elements do not change and also that merged elements can be identified by the corresponding set of ids.)

Having a meta-model co-span, the co-span can be incrementally decomposed using the following procedure:

Procedure 1 (Decompose meta-model co-span)**.** *(see Fig. 7)*
Given meta-model co-span $M_1 \overset{a_{max}}{\to} M_{max} \overset{b_{max}}{\leftarrow} M_N$ *and a set of co-span rules* R*.*

1. *Find a triple match* $<m_L, m_I, m_R>$ *of a co-span rule* $r \in R$ *in* $M_1 \to M_{max} \leftarrow M_N$*.*
2. *Apply co-span rule* r *to meta-model* M_1*.*
3. *Obtain* $d : M_2 \to M_{max}$ *as mediating morphism from the left pushout* PO_L *of* r*'s rule application.*

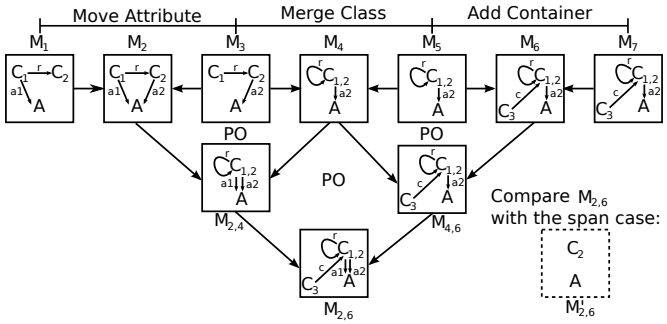

Figure 6: Example: Meta-model co-span composition

4. *Construct* $M_3 \leftarrow M_{sub} \rightarrow M_N$ *as pullback of* $M_3 \overset{h;d}{\rightarrow} M_{max} \overset{b_{max}}{\leftarrow} M_N$.
5. *Construct new meta-model co-span* $M_3 \overset{a_{max-1}}{\rightarrow} M_{max} \overset{b_{max-1}}{\leftarrow} M_N$ *as pushout of* $M_3 \leftarrow M_{sub} \rightarrow M_N$.
6. *Continue to decompose co-span* $M_3 \overset{a_{max-1}}{\rightarrow} M_{max-1} \overset{b_{max-1}}{\leftarrow} M_N$ *(recursion).*
   Stop if $a_{max-1} = id = b_{max-1}$ *or no more triple matches found.*

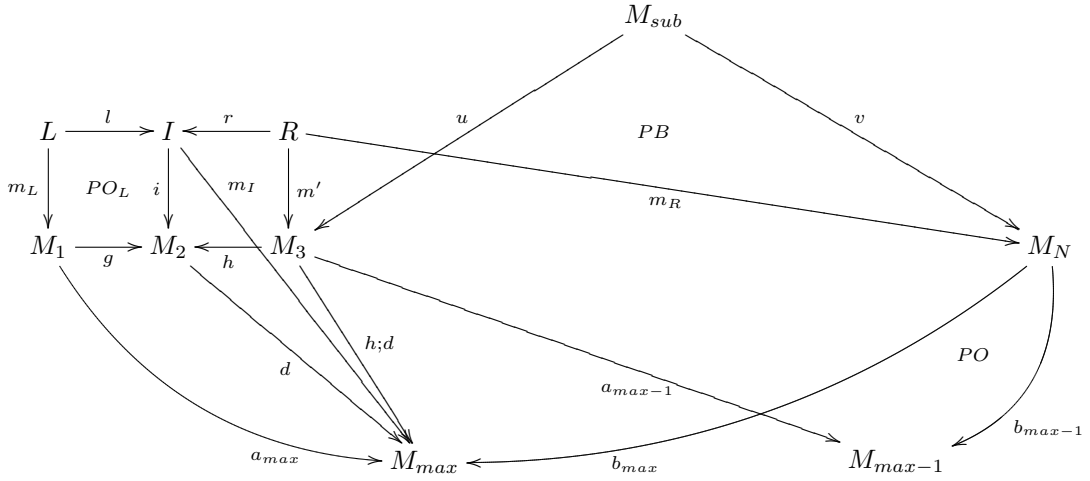

Figure 7: Meta-model co-span decomposition

Remark 1.

1. *Rule matches can be searched in parallel.*
2. *Procedure 1 is also applicable if the sesqui pushout approach is used. (The pushout property of the second rule application is not used in the procedure.)*
3. *If non-injective rules are used, non-injective matching may be required.*

Figure 8 shows an example decomposition step. In each further decomposition step the difference between M_{max-1} and M_{sub} becomes smaller. A particular sequence is not guaranteed.

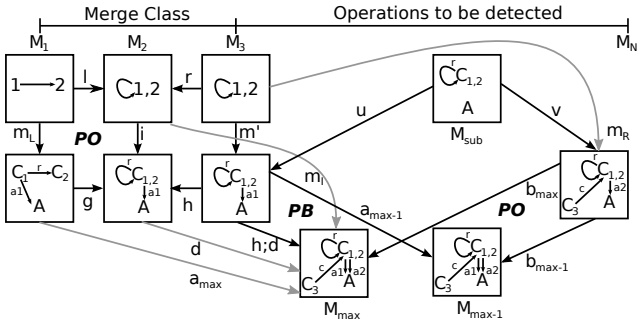

Figure 8: Example: Meta-model co-span decomposition

Having a meta-model span relation such a decomposition is hardly feasible; in Fig. 6 the dashed part shows the intermediate meta-model in case we would have used a meta-model span. Hence, a co-span approach seem to fit better to the meta-model evolution challenge and in particular to the formalization of matching approaches.

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science (3^{rd} Edition*. Les Publications CRM, Montreal, 1999.

[2] A. Corradini, T. Heindel, F. Hermann, and B. König. Sesqui-Pushout Rewriting. In A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, editors, *ICGT 2006*, volume 4178 of *LNCS*, pages 30–45. Springer, September 2006.

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

[4] H. Ehrig, F. Hermann, and U. Prange. Cospan DPO Approach: An Alternative for DPO Graph Transformation. *EATCS Bulletin*, 98:139–149, 2009.

[5] M. Fowler. *Domain-Specific Languages*. Addison-Wesley Professional, 2010.

[6] Y. Lamo, F. Mantz, A. Rutle, and J. de Lara. A declarative and bidirectional model transformation approach based on graph co-spans. In *Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming*, PPDP '13, pages 1–12, New York, NY, USA, 2013. ACM.

[7] F. Mantz, G. Taentzer, and Y. Lamo. Co-Transformation of Type and Instance Graphs Supporting Merging of Types with Retyping. In *GCM 2012*, pages 47–58, September 2012. `http://gcm2012.imag.fr/proceedingsGCM2012.pdf`.

[8] F. Mantz, G. Taentzer, and Y. Lamo. Well-formed Model Co-evolution with Customizable Model Migration. *ECEASST*, page (accepted paper), March 2013.

[9] L. Rose, D. Kolovos, R. F. Paige, and F. A. C. Polack. Model Migration with Epsilon Flock. In L. Tratt and M. Gogolla, editors, *ICMT 2010*, volume 6142 of *LNCS*, pages 184–198. Springer, 2010.

[10] G. Taentzer, F. Mantz, and Y. Lamo. Co-Transformation of Graphs and Type Graphs With Application to Model Co-Evolution. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *ICGT 2012*, volume 7562 of *LNCS*, pages 326–340. Springer, 2012.