# Formalising Metamodel Evolution based on Category Theory

Florian Mantz,* Yngve Lamo
Bergen University College
{fma,yla}@hib.no

Alessandro Rossini, Uwe Wolter
University of Bergen
{rossini,wolter}@ii.uib.no

Gabriele Taentzer
Philipps-Universität Marburg
taentzer@informatik.uni-marburg.de

Model-driven engineering (MDE) is a branch of software engineering which aims at improving productivity, quality, and cost-effectiveness of software development by shifting the paradigm from code-centric to model-centric activities. MDE promotes models and modelling languages as the main artefacts of the development process and model transformation as the primary technique to generate (parts of) software systems out of models. Models enable developers to reason at a higher level of abstraction, while model transformation restrains developers from repetitive and error-prone tasks such as coding. Although techniques and tools for MDE have advanced considerably during the last decade, several concepts and standards in MDE are still defined semi-formally, which may not guarantee the degree of precision required by MDE.

Models can be specified using general-purpose languages like the Unified Modeling Language (UML) [8], but to fully unfold the potential of MDE, models are often specified using domain-specific languages (DSLs) which are tailored to a specific domain of concern. One way to define DSLs in MDE is by specifying metamodels, which are models that describe the concepts and define the syntax of a DSL. A model is said to *conform to* a metamodel if each element in the model is typed by an element in the metamodel and, in addition, satisfies all constraints of the metamodel.

Models and metamodels undergo complex evolutions during their life cycles. As a consequence, when a metamodel is modified, models conforming to this metamodel need to be migrated in such a way that they conform to the modified version (see Fig. 1). In the literature, this problem is referred to as metamodel evolution [7] or model co-evolution [4].

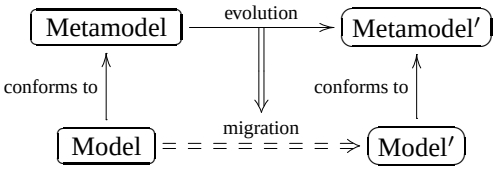


Figure 1: Model co-evolution: Metamodel evolution and model migration

To address this problem, a few prototype tools have been developed that support metamodel evolution in different scenarios, e.g., [7, 9]. However, a uniform formalisation of metamodel evolution is still lacking. The relation between metamodel- and model changes should be formalised in order to allow reasoning about the correctness of migration definitions. In addition, constraints in models and metamodels should also be handled during migration. This work proposes a formal approach to metamodel evolution which addresses some of these challenges. The approach is based on the Diagram Predicate Framework (DPF) [2], a formal diagrammatic specification framework founded on category theory [1] and graph transformation [5]. DPF provides the means to specify models with diagrammatic constraints and defines a conformance relation between models and metamodels which takes into account these constraints [10].

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

In DPF, a model is represented by a *specification* $\mathfrak{S}$. A specification $\mathfrak{S} = (S, C^{\mathfrak{S}} : \Sigma)$ consists of an *underlying graph* $S$ together with a set of *atomic constraints* $C^{\mathfrak{S}}$ which are specified by means of a *signature* $\Sigma$. A signature $\Sigma = (\Pi^{\Sigma}, \alpha^{\Sigma})$ consists of a set of *predicates* $\pi \in \Pi^{\Sigma}$, each having an arity (or shape graph) $\alpha^{\Sigma}(\pi)$, a semantic interpretation, and a proposed visualisation. An atomic constraint $(\pi, \delta)$ consists of a predicate $\pi \in \Pi^{\Sigma}$ together with a graph homomorphism $\delta : \alpha^{\Sigma}(\pi) \to S$ from the arity of the predicate to the underlying graph of the specification.

The semantics of nodes and arrows of a specification has to be chosen in a way which is appropriate for the corresponding modelling environment [11]. In object-oriented structural modelling, it is appropriate to interpret nodes as sets and arrows $\mathsf{X} \xrightarrow{\mathsf{f}} \mathsf{Y}$ as multi-valued functions $f : X \to \wp(Y)$. The semantics of a specification is defined in the fibred way [12]; i.e., the semantics of a specification $\mathfrak{S} = (S, C^{\mathfrak{S}} : \Sigma)$ is given by the set of its instances $(I, \iota)$. An instance $(I, \iota)$ of a specification $\mathfrak{S}$ consists of a graph $I$ together with a graph homomorphism $\iota : I \to S$ which satisfies the set of atomic constraints $C^{\mathfrak{S}}$.

Metamodel- and model changes can be formalised in DPF as specification transformation rules, which can be regarded as an extension of graph transformation rules [5]. In this work, possible metamodel changes are restricted to a specific set of metamodel evolution/migration rules. The migration rules are derived from metamodel evolution rules by retyping them on the model level; i.e., an isomorphic migration rule is derived from a metamodel evolution rule by replacing each metamodel element by its instance element. This rule is matched as often as possible on a model. This approach can be considered as a special kind of amalgamated graph transformation rule [3] with an empty kernel rule.

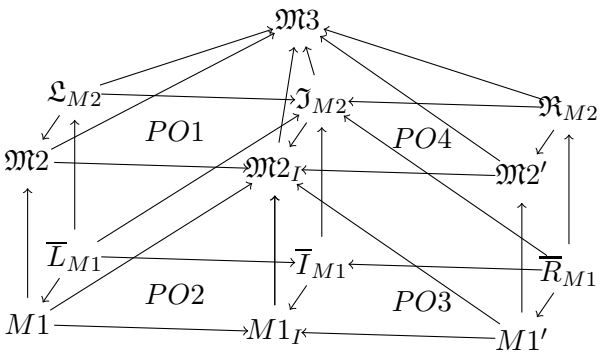


Figure 2: Relations of metamodel- and model changes

Figure 2 shows the graph homomorphisms between a metamodel evolution rule and a model migration rule. These rules are formulated using the cospan double pushout (Cospan DPO) approach [6], which first extends a graph and then reduces it. Equivalence to the original DPO approach is shown by Ehrig et al. in [6]. This approach has been chosen since it allows the models to be adapted in-place. Firstly, a metamodel is extended by the pushout over the span $\mathfrak{M}2 \leftarrow \mathfrak{L}_{M2} \rightarrow \mathfrak{I}_{M2}$ (PO1). Afterwards, a conforming model is extended by PO2 over the span $M1 \leftarrow \overline{L}_{M1} \rightarrow \overline{I}_{M1}$ and then reduced by PO3 over the span $\overline{I}_{M1} \leftarrow \overline{R}_{M1} \rightarrow M1'$. Finally, the metamodel is reduced by PO4 over the span $\mathfrak{I}_{M2} \leftarrow \mathfrak{R}_{M2} \rightarrow \mathfrak{M}2'$. The application sequence of these pushouts allow that models stay type conform during the entire migration process.

Figure 3 shows the graph homomorphisms between a metamodel evolution rule and a model migration rule in more detail. The metamodel evolution rule is represented by the cospan $\mathfrak{L}_{M2} \rightarrow \mathfrak{I}_{M2} \leftarrow \mathfrak{R}_{M2}$, whereas copies of the derived isomorphic migration rule are represented by the family of cospans $L_i \rightarrow I_i \leftarrow R_i$ with $1 \leq i \leq n$. The applicable model migration rule is represented by the cospan $\overline{L}_{M1} \rightarrow \overline{I}_{M1} \leftarrow \overline{R}_{M1}$, which is constructed by the disjoint union. The disjoint union can be charac-

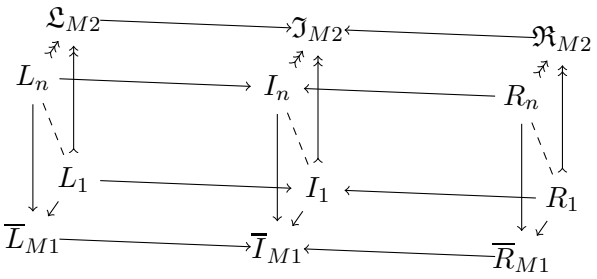


Figure 3: Amalgamated migration rule

terised as coproduct in a corresponding rule category. That fact that the disjoint union is also typed over the metamodel evolution rule can be shown by the universal property of coproducts.

This migration rule deduction strategy is only useful for a subset of metamodel evolution rules. Therefore, the metamodel evolution rules are extended by DPF's atomic constraints. These atomic constraints restrict the possible application of metamodel evolution rules to those cases where the deduction strategy is sufficient. For example, a multiplicity constraint [1..*] added to an arrow of the LHS graph of the metamodel evolution rule prevents this arrow from matching with a metamodel arrow having multiplicity constraint [0..*]. Currently, each arrow $\mathsf{X} \xrightarrow{\mathsf{f}} \mathsf{Y}$ in the metamodel evolution rule is required to be total and surjective, i.e., $|f(x)| \geq 1$ and $\forall y \in Y,\ \exists x \in X : y \in f(x)$. Furthermore, LHS and RHS graphs of metamodel evolution rules with loops and nodes being targets of more than one arrow are not considered for the automatic deduction of migration rules yet.

## References


[1] M. Barr and C. Wells. *Category Theory for Computing Science (2nd Edition)*. Prentice Hall, 1995.

[2] Bergen University College and University of Bergen. *Diagram Predicate Framework Web Site*. http://dpf.hib.no/.

[3] E. Biermann, C. Ermel, and G. Taentzer. Formal foundation of consistent EMF model transformations by algebraic graph transformation. *SoSyM (Online First)*, pages 1–24, 2011.

[4] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio. Automating Co-evolution in Model-Driven Engineering. In *EDOC 2008*, pages 222–231. IEEE Computer Society, 2008.

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

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

[7] M. Herrmannsdoerfer, S. Benz, and E. Jürgens. COPE - Automating Coupled Evolution of Metamodels and Models. In *ECOOP 2009*, volume 5653 of *LNCS*, pages 52–76. Springer, 2009.

[8] Object Management Group. *Unified Modeling Language Specification*, May 2010. http://www.omg.org/spec/UML/2.3/.

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

[10] A. Rutle. *Diagram Predicate Framework: A Formal Approach to MDE*. PhD thesis, Department of Informatics, University of Bergen, Norway, 2010.

[11] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter. A Diagrammatic Formalisation of MOF-Based Modelling Languages. In *TOOLS 2009*, volume 33 of *LNBIP*, pages 37–56. Springer, 2009.

[12] U. Wolter and Z. Diskin. From Indexed to Fibred Semantics – The Generalized Sketch File. Technical Report 361, Department of Informatics, University of Bergen, Norway, October 2007.