Functorial Diagram Operators

Navid Roux and Florian Rabe

Computer Science, University, Erlangen-Nuremberg

Abstract. Theory operators are meta-level operators in logic that map theories to theories. Often these are functorial in that they can be extended to theory morphisms and possibly enjoy further valuable properties such as preserving inclusions. Thus, it is possible to apply them to entire diagrams at once, often in a way that the output diagram mimics any morphisms or modular structure of the input diagram. We investigate the properties of diagram operators and give numerous examples. All our examples are formalized in the Mmt meta-logical framework.

Diagrams of theories and theory morphisms have long been used successfully to build large networks of theories both in algebraic specification languages [SW83,ST88,CoF04] and in deduction systems [FGT93,KWP99,SJ95,RK13]. In particular, it enables the use of meta-level operators on theories such as union and translation to build large structured theories in a modular way.

Recently, inspired by ideas in [DOL18,CO12], the second author generalized this idea to diagram operators [SR19], where an operator is applied not to a single theory but to an entire diagram at once. For example, we can form the diagram `Magmas` of the magma-hierarchy (including theories for semigroup, commutative magma, etc.), apply an operator to enrich the diagram with all unions (band, semilattice, etc.), and finally apply another operator to obtain in one step the corresponding diagram of homomorphisms (including theories for band homomorphisms, etc.).

In particular, the latter operator to build the theory Hom_T of homomorphisms of models of T for an arbitrary first-order theory T has a number of interesting properties: for example, it is functorial and preserves inclusions between theories. In this work, we expand on this idea by studying such operators and their properties in detail.

As a running example we define an operator we call *polymorphify*, which maps a theory T of sorted first-order logic (SFOL) to a theory $\texttt{Poly}(T)$ of polymorphic SFOL: it maps every declaration in T of a

- sort s to a unary sort operators s,
- every function symbol $f\colon s_1 \to \ldots \to s_n \to s$ to a polymorphic function symbol $f\colon \{a\}\ s_1(a) \to \ldots \to s_n(a) \to s(a)$ abstracting over an arbitrary sort a,
- every predicate symbol in the same way as function symbols,
- every axiom asserting F to an axiom asserting $\{a\}\ F'$ where F' arises from F by replacing every sort, function, and predicate symbol with itself applied to a

in `Poly`(T). Applied to the hierarchy of magmas, this immediately gives us various theories of collection datatypes: polymorphic monoids (i.e., lists), polymorphic commutative monoids (i.e., multisets), and polymorphic bounded semilattices (i.e., sets) — all derived from the single concise expression `Poly(Magmas)`. Moreover, the operator is functorial, i.e., `Poly(Magmas)` contains theory morphisms and inclusions wherever `Magmas` does.

All our operators extend the implementation of diagram operators presented in [SR19], which uses the Mmt meta-framework for building formal systems [Rab20]. Every operator is declared as an Mmt symbol and implemented by a computation rule. Thus, all our operators are directly usable in Mmt specifications, and all our examples have been formalized in that way.

Because Mmt is foundation-independent per se, these diagram operators are immediately applicable to any logic defined in any logical framework implemented in Mmt. This includes any logic defined in the LATIN logic atlas [CHK+11], which itself is defined within the LF framework [HHP93]. While most interesting operators are logic-specific, we maximize reuse by implementing all our operators relative to the weakest possible logic.

References

CHK+11. M. Codescu, F. Horozal, M. Kohlhase, T. Mossakowski, and F. Rabe. Project Abstract: Logic Atlas and Integrator (LATIN). In J. Davenport, W. Farmer, F. Rabe, and J. Urban, editors, *Intelligent Computer Mathematics*, pages 289–291. Springer, 2011.

CO12. J. Carette and R. O'Connor. Theory Presentation Combinators. In J. Jeuring, J. Campbell, J. Carette, G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge, editors, *Intelligent Computer Mathematics*, volume 7362, pages 202–215. Springer, 2012.

CoF04. CoFI (The Common Framework Initiative). *CASL Reference Manual*, volume 2960 of *LNCS*. Springer, 2004.

DOL18. DOL editors. The distributed ontology, modeling, and specification language (dol). Technical report, Object Management Group, 2018.

FGT93. W. Farmer, J. Guttman, and F. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, 1993.

HHP93. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.

KWP99. F. Kammüller, M. Wenzel, and L. Paulson. Locales – a Sectioning Concept for Isabelle. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors, *Theorem Proving in Higher Order Logics*, pages 149–166. Springer, 1999.

Rab20. F. Rabe. MMT: The Meta Meta Tool. see `https://kwarc.info/people/frabe/Research/rabe_mmtsys_20.pdf`, 2020.

RK13. F. Rabe and M. Kohlhase. A Scalable Module System. *Information and Computation*, 230(1):1–54, 2013.

SJ95. Y. Srinivas and R. Jüllig. Specware: Formal Support for Composing Software. In B. Möller, editor, *Mathematics of Program Construction*. Springer, 1995.

SR19. Y. Sharoda and F. Rabe. Diagram Operators in MMT. In C. Kaliszyk, E. Brady, A. Kohlhase, and C. Sacerdoti Coen, editors, *Intelligent Computer Mathematics*, pages 211–226. Springer, 2019.

ST88. D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Control*, 76:165–210, 1988.

SW83. D. Sannella and M. Wirsing. A Kernel Language for Algebraic Specification and Implementation. In M. Karpinski, editor, *Fundamentals of Computation Theory*, pages 413–427. Springer, 1983.