Understanding Mathematical Theory Formation via Theory Intersections in MMT

Dennis Müller & Michael Kohlhase

Computer Science, Jacobs University Bremen
http://kwarc.info

Abstract. We look at the concept of *theory intersections* inMMT, as a way to refactor theories in formal libraries, potentially yielding new interesting mathematical theories in the process. We argue, that theory intersections are a historically validated method to do so and present a concrete implementation in MMT.

Introduction An important driver of mathematical development is the discovery of new mathematical objects, concepts and theories. Even though there are many different situations that give rise to a new theory, it seems that a common instance is the discovery of commonalities between apparently different mathematical structures (if such exist). In fact, many of the algebraic theories in Bourbaki naturally arise as the "common part" of two (or more) different mathematical structures – for instance, one could interpret the theory of groups as the common theory of $(\mathbb{Z}, +, 0)$ and the set of symmetrical operations on e.g. a square.

In [1], the second author proposes a notion of *theory intersection* – elaborating an earlier formalization from [2] to MMT [3, 4],that captures this phenomenon in a formal setting: Let two theories S and T, a partial theory morphism $S \xrightarrow{\sigma} T$ with domain D and codomain C, and its partial inverse δ (as in Figure 1 on the left) be given. Then we can pass to a more modular theory graph, where $S' := S \backslash D$ and $T' := T \backslash C$ (as in Figure 1 on the right). In this case we think of the equivalent theories D and C as the *intersection* of theories S and T along σ and δ.

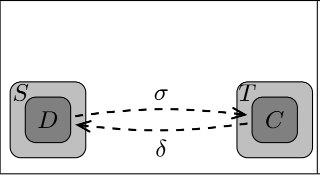
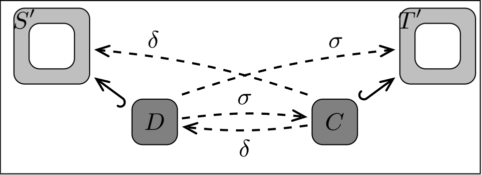
Situation	Theory Intersection
	

Fig. 1: Theory Intersection

Theory Intersection by Example To fortify our intuitions, we examine a concrete mathematical example in detail.

Positive	Strings

theory PosPlus = {
$\mathbb{N}$: type
+ : $\mathbb{N}$ −> $\mathbb{N}$ −> $\mathbb{N}$; assoc, comm
}
	theory StrConc = {
	A^* : type
	:: : A^* −> A^* −> A^*; assoc
	ε : A^*, unit
	}

k = pmorph PosPlus −> StrConc
{$\mathbb{N}$:= A^*, + := ::, assoc := assoc}
	l = pmorph StrConc −> PosPlus
	{A^* := $\mathbb{N}$, :: := +, assoc := assoc}

Fig. 2: Positive Integers and Strings

We start out with a theory PosPlus of positive natural numbers with addition and intersect it with StrConc of strings with concatenation (as in Figure 2). Note that we do not start with modular developments; rather the modular structure is (going to be) the result of intersecting with various examples. We have extended the MMT syntax with the keyword pmorph for partial theory morphisms.

Positive	Strings

theory A = {
$\mathbb{N}$: type
+ : $\mathbb{N}$ −> $\mathbb{N}$ −> $\mathbb{N}$; assoc
}
theory PosPlus = { include A
comm : ...
}
	theory B = {
	A^* : type
	:: : A^* −> A^* −> A^*; assoc
	}
	theory StrConc = { include B
	ε : A^*, unit
	}

Fig. 3: Positive Integers and Strings, Refactored

Now the intersection as proposed above yields Figure 3, which directly corresponds to the schema in Figure 1. Note, that the new pair of (equivalent) theories is completely determined by the partial morphism k; the only thing we have to invent are the names – and we have not been very inventive here.

Intuitively, the intersection theory is the theory of semigroups which is traditionally written down in MMT as in Figure 4 on the right. And indeed, sg is a renaming of both A and B.

```
sg = theory {
  G : type
  ∘ : G −> G −> G; assoc
  }
```

Fig. 4: Semigroups

For this situation, we should have a variant theory intersection operator that takes a partial morphism and returns *one* intersection theory.

In this situation, we want a theory intersection operator that follows the schema on the left (Figure 5).

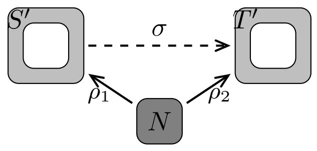


Fig. 5: Unary TI

Let us call this operation *unary TI* and the previous one *binary TI*. To compute it, we need to specify a name N for the new theory and two renamings $\rho_1 : N \to \mathrm{Dom}(\sigma)$ and $\rho_2 : N \to \mathsf{Img}(\sigma)$. Note that in this TI operator, the intersection is connected to the "rest theories" via MMT structures – rather than mere inclusions – which carry the assignments induced by the partial morphisms suitably composed with the renamings.

In our example we can obtain the theory sg from Figure 4 via the renamings $\rho_1 := \{\mathsf{G} := \mathbb{N}, \circ := +, \mathsf{assoc} := \mathsf{assoc}\}$ and $\rho_2 := \{\mathsf{G} := A^*, \circ :=::, \mathsf{assoc} := \mathsf{assoc}\}$.

Unary TI is often the more useful operation on theory graphs, but needs direct user supervision, whereas binary TI is fully automatic if we accept generated names for the intersection theories.

Reducing Partial Morphisms to Renamings In the previous example it is noteworthy, that the morphisms k and l are fully inverse to each other. In fact, they are *renamings*. By a renaming, we mean a partial morphism $S \xrightarrow{\sigma} T$ such, that for every constant c declared in S, $\sigma(c)$ is again a constant (as opposed to a complex term).

Naturally, theory intersections are a lot simpler for renamings. In fact, we only need a single renaming $S \xrightarrow{\sigma} T$ to intersect along (since the inverse of a renaming is again a renaming and uniquely determined). It turns out, that we can always reduce the situation in Figure 1 to the much easier case, where we have a single renaming $S' \xrightarrow{\sigma} T'$, where S' and T' are conservative extensions of our original theories:

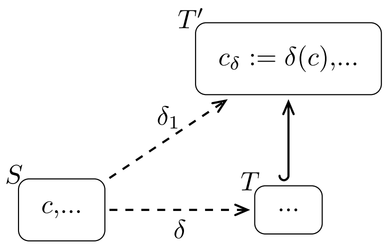


Fig. 6: Obtaining a Renaming From a Morphism

Let $S \xrightarrow{\delta} T$ be a partial theory morphism and $\delta(c) = t$ for a complex term t. We can then conservatively extend T to a theory T_1 that contains a new defined constant $c_\delta := t$ and adapt δ to a new morphism δ_1 with $\delta_1(c) = c_\delta$ (and $\delta_1 \restriction S \setminus \{c\} = \delta \restriction S \setminus \{c\}$), as in Figure 6. If we repeat this process for every mapping $\delta(c) = t$ that is not already a simple renaming, we yield a new conservative extension T' such, that the corresponding morphism δ' (accordingly adapted) is a renaming.

Doing the same with a partial inverse $T \xrightarrow{\rho} S$, we ultimately get a single renaming $S' \xrightarrow{\sigma} T'$ that is equivalent to the original pair of partial morphisms. We can therefore w.l.o.g. reduce ourselves in every instance to this case.

Implementation in MMT We have implemented a method for unary TI in the current MMT api, as well as a method for finding partial theory morphisms (using the internal *Viewfinder*, although the found morphisms are not technically views except according to the current internal specification of MMT) between MMT theories.

The intersection method takes as parameters two theories, a name for the intersection and a list of pairs of declarations, which can be obtained from a pair of morphisms, a single morphism or the Viewfinder. It returns the intersection theory and the refactored versions of (conservative extensions of) the original theories, depending on the original morphisms provided.

Both methods are integrated into a refactoring panel, which is part of the MMT plugin for jEdit. To intersect two theories, the user can either provide one or two morphisms between the theories or allow the Viewfinder to pick for them. The declarations in the intersection theory can optionally be named.

An annotated video demonstration of the refactoring panel and its components can be found on Youtube [5].

The Viewfinder looks specifically for renamings between theories, using a (in principle flexible) judgment-based approach; i.e. two declarations are (usually) only matched, if they occur in some judgments (i.e. axioms or theorems) that can be matched. This reduces the search space, increasing efficiency and avoiding "meaningless" morphisms that do not preserve some of the properties of a declaration.

If we accept automatically generated names for the theory intersections, we can eliminate user interaction and use the Viewfinder to automate the intersection operation on a set of theories without having to provide the morphisms to intersect along beforehand. This could be used to refactor whole libraries in a more modular way (in concordance with the *little theories* approach [6]).

The Viewfinder has been tested on the LATIN library [7] with experimental success.

We do not yet have a heuristic to evaluate (the resulting) theories automatically with respect to their interest; however, given a morphism between two interesting theories, we have observed that the corresponding intersection tends to be interesting as well, as long as the morphism is not meaningless itself.

Conclusion We have presented theory intersections and the related implemented methods in MMT. As a next step, we will evaluate the intersection method on a suitable set of theories.

References

[1] Michael Kohlhase. "Mathematical Knowledge Management: Transcending the One-Brain-Barrier with Theory Graphs". In: *EMS Newsletter* (June 2014), pp. 22–27. URL: `http://www.ems-ph.org/journals/newsletter/pdf/2014-06-92.pdf`.

[2] Immanuel Normann. "Automated Theory Interpretation". PhD thesis. Bremen, Germany: Jacobs University, 2008. URL: `https://svn.eecs.jacobs-university.de/svn/eecs/archive/phd-2008/normann.pdf`.

[3] Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: `http://kwarc.info/frabe/Research/mmt.pdf`.

[4] Fulya Horozal, Michael Kohlhase, and Florian Rabe. “Extending MKM Formats at the Statement Level”. In: *Intelligent Computer Mathematics*. Ed. by Johan Jeuring, John A. Campbell, et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 65–80. URL: http://kwarc.info/kohlhase/papers/mkm12-p2s.pdf.

[5] *Theory Intersections in MMT*. URL: https://www.youtube.com/watch?v=qXKaGuV7kLY.

[6] William M. Farmer, Josuah Guttman, and Xavier Thayer. “Little Theories”. In: *Proceedings of the 11th Conference on Automated Deduction*. Ed. by D. Kapur. LNCS 607. Saratoga Springs, NY, USA: Springer Verlag, 1992, pp. 467–581.

[7] M. Codescu, F. Horozal, et al. “Project Abstract: Logic Atlas and Integrator (LATIN)”. In: *Intelligent Computer Mathematics*. Ed. by J. Davenport, W. Farmer, F. Rabe, and J. Urban. Springer, 2011, pp. 289–291.