Functors in a Web-Scalable Module System

Guided Research May 2008

Elena Agapie
Supervisors: Prof. Dr. Michael Kohlhase, Florian Rabe

Jacobs University Bremen

Abstract

Mathematical knowledge has reached a size when it has become extremely hard to organize. A new method to organize mathematical knowledge is through representing mathematical theories in a computer system. There are several languages that can organize mathematical knowledge. We will focus on the MMT language that provides a web scalable module system for mathematical theories. In MMT the theories are organized hierarchically which allows the representation of symbols and morphisms between theories. Model categories can be represented in MMT through theories. A weakness of MMT is that it cannot represent all functors between model categories. We propose an extension of MMT that will also allow the representation of functors through λ-expressions abstracting over morphisms.

1 Introduction and Related Work

Most modern sciences are built upon mathematical concepts. Nowadays, one of the most common ways to make mathematical publications available is through the internet. However, this does not also organize the information semantically. Since the mathematical information is continuously increasing (it is estimated it doubles every ten-fifteen years) there is an even stronger need to organize it [1].

Mathematical knowledge is based on a rigorous language that can in principle be formalized in logics. Much of this knowledge is represented through networks of mathematical theories. Thus all the mathematical texts can be understood with respect to these theories. Theories can be formalized as a collection of symbol declarations and axioms. Symbols are objects characteristic to the theory and axioms are logical formulas which state the laws that are governing the objects described in the theory. The biggest challenge in representing mathematics on the internet is to capture both the notation and the content of mathematics. Our work will focus on MMT ([1]), a web-scalable module for mathematical theories. MMT has a focus on modularity in organization.

MMT represents theories and theory morphisms in a theory graph. It also allows the representation of morphisms between theories or mapping the terms of one theory to the terms of another. What the system lacks is the representation of functors in this hierarchy. Functors are encountered very often in mathematics such that we would want to integrate them in a module system and MMT allows us to do that.

We will give a solution for the representation of functors between objects of a category. In MMT we can define morphism between theories that can translate symbols and axioms from one theory to

another, however we cannot represent operations that take as arguments other morphisms. The main modification that we are making is based on a rule which uses λ-calculus ([2]) to give morphisms that take some other morphism as an argument. The resulting module system is not *conservative* because functors could not be defined in the structure of the MMT language.

Theory-like structures described above have applications in **module systems**, where modules are used for encapsulating program functionality in meaningful units. Modules can be understood as a theories and morphisms provide values or implementations for the symbols declared in the modules.

MMT provides a way of representing mathematical knowledge, focusing on modularity organization, flexible reuse and management of change. It uses the representation of mathematics as networks of mathematical theories. We have chosen this system as it allows mathematical representation and it provides a well-structured system for theories. Its style of representing mathematics is based on the Little Theories Approach [3], in which separate contexts are represented by separate theories and the approach used by the Bourbaki group [4]. The latter tried to prove theorems using the smallest possible set of axioms from the theory level representation of mathematics. The translations that preserve the truth of the theorems are referred to as **theory morphisms**. The utilized format for MMT is OMDoc (Open Mathematical Documents)[5] format for representing theories, having XML as a syntactical basis. OMDoc provides the content oriented markup format for mathematical knowledge and documents. Some other module systems that present interest regarding our proposed problem are **PVS** [6] and **Isabelle** [7].

PVS is an interactive theorem prover for a variant of a classical higher order logic. Modules begin with an interface part, and declarations are available for instantiations which map symbols to terms. We encounter specifications, which consists of a collection of theories. Theories are built on top of other theories and can be parametric in certain specified types and values. PVS allows imports that can take types, constants and formulas from one theory to another. This can be done by providing the actual parameters through an "IMPORTING" clause.

Isabelle is a type theoretical logical framework based on higher-order logic. It allows mathematical formulas to be expressed in a formal language. It provides an interactive theorem prover and a structured high-level proof language. Locales are used as modules. Isabelle also uses an importing mechanism that allows imports between locales. In the locale module system every theorem proved in the locale is relative by the locale predicate and exported to the top level.

The paper is divided into 5 sections. After the introduction and related work, in Section 2 we review MMT and theory representation in MMT. Section 3 details how the functors can be represented. In 3.1 we present a short overview of categories and MMT. In 3.2 we present the solution at an intuitive level. Section 3.3 gives the syntax for representing functors. Section 3.4 presents the rules for well formed expressions in the new grammar.

2 Theory Representation in MMT

Theory graphs are used to illustrate the theories as nodes and theory morphisms as edges. Theories contain symbols and axioms. The axioms are considered special symbols, following the Curry-Howard correspondence ([8]). We can represent a collection of theories through theory graphs where an **import** is a morphism which has the property that all symbols and axioms of the source theory become symbols and axioms of the target theory.

The MMT language represents theories in a hierarchical manner. Thus, when we represent the signature and axioms of a theory, the representation of another theory that extends the initial one only requires the representation of the symbols and axioms that are added to the initial theory. This translation process from one theory to another is called *import*. The MMT system also makes use of *meta-theories*. These allow to operate with a theory without choosing a particular foundational logical

system, but by using content dictionaries for the symbols in a particular logic. For e.g. $Monoid$ would use FOL as a meta-theory. This allows not only to import FOL when defining $Monoid$ but also to express a meta-relation between theories.

We will illustrate the representation of theories in the MMT system by showing how theories are represented for the following hierarchy of algebraic structures. We will consider the theories for monoids, groups, rings and ZFC (Zermelo Fraenkel set theory, with the axiom of choice) . The diagram below illustrates how these theories are represented hierarchically. We show these relations in a graph where every node represents one of the above theories and its internal symbols and the edges represent the morphisms from one theory to another. All the theories build upon FOL.

In the MMT system we express theory morphisms, via imports and views. The imports allow us to define the theories in a tree like structure and to extend a theory into other theories. The views allow us to also define substitutions from one theory to another.

Let us consider the theory of the Monoid, Group and ZFC as in Fig. 1. The theory of groups extends the theory of monoids via an import mon, and we can construct theory morphisms from Monoid and Group to ZFC. In these theories we can represent the objects from categories. For e.g. we can define the monoid of matrices $\mathbb{R}^{2\times 2}$ through a morphism $Monoid \to ZFC$ and the group of invertible matrices $GL_2(\mathbb{R})$ through a morphism $Group \to ZFC$.

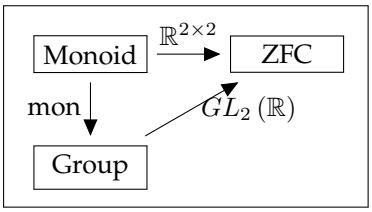

Figure 1: Theories in MMT

A theory can have one or more imports. An import from one theory S to a theory T, the axioms and symbols from S are translated to T. For example $Ring$ has imports from $Monoid$ and from $Group$. The imports are named *add* and *mult*. Through this import, $Ring$ actually imports the symbols of $Monoid$ twice. The symbols are disambiguated by named imports (see Fig. 2).

The morphism between $Monoid$ or $Group$ to ZFC is a **view** and gives an interpretation of the axioms and symbols from the source theory to ZFC, where we can consider that we have already the real numbers structure defined. The morphisms towards ZFC represents an example of views in MMT. We will discuss this example in the next section. In MMT these theories have FOL as a meta-theory. We will consider a reduced form of MMT in which we are not using meta-theories but we will assume that the above theories have imported the FOL symbols and axioms in their own theories.

The relations between the above theories are illustrated in Fig. 2. The axioms and symbols of $Monoid$, $Group$ and ZFC are detailed underneath. ZFC has a more complicated representation as a theory and we will illustrate that in natural language. They will be used as our main example further in the paper. We will only sketch the $Ring$ structure. We are using a simplified syntax instead of OpenMath for better readability.

Monoid :={
$symbols\ and\ axioms\ of\ FOL$,
$e : universe$,
$\circ : universe \to universe \to universe$,
$e_l : true\ forall[x]\ e\ \circ\ x\ eq\ x$,
$e_r : true\ forall[x]\ x\ \circ\ e\ eq\ x$,
$assoc : true\ forall[x]\ forall[y]\ forall[z]\ (x\ \circ\ y)\ \circ\ z\ eq\ x\ \circ\ (y\ \circ\ z)$
},
Group := {
$symbols\ and\ axioms\ of\ FOL$,
$mon : import\ from\ Monoid$,
$inv : universe \to universe$,

$inv_l : true\ forall[x]\ exists[y]\ y\ mon/\circ\ x\ eq\ mon/e,$
$inv_r : true\ forall[x]\ exists[y]\ x\ mon/\circ\ y\ eq\ mon/e$
},
ZFC := {
$symbols\ and\ axioms\ of\ FOL,$
$set : universe,$
%set theory and symbols such as:
$empty : set,$
$double : set \to set \to set,$ %{ x,y }
$powerset : set \to set,$
$unions : set \to set,$ %union sets in sets
...
$union : [x][y]\ unions\ (double\ x\ y),$
$inter : ...,\ subset : ...,\ disjoint : ...,$
...
% axioms of ZFC
$extensionality : ...,\ foundation : ...,\ choice : ...,\ ...$
},
Ring := {
$symbols\ and\ axioms\ of\ FOL,$
$add : import\ from\ Group, mult : import\ from\ Monoid$
$0 : add/grp/mon/e, 1 : mult/mon/e,$
$+ : add/grp/mon/\circ, * : mult/mon/\circ,$
$distrib\text{-}left : ..., distrib\text{-}right : ..., comm :$ ... axioms in ring.
}

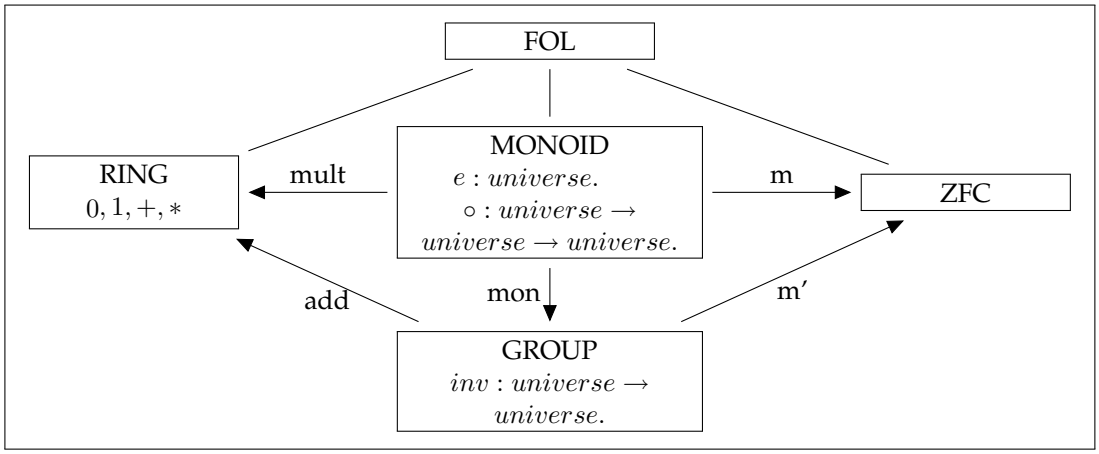

Figure 2: Theory representation

where in $Monoid$, e represents the unit of the monoid, e_l and e_r represent the left, respectively right unit, $\circ$ represents the composition law; in $Group$, inv represents the inverse, inv_l and inv_r are respectively the left and right inverse laws; $comm$ is the commutativity axiom and $assoc$ the associativity axiom. The ZFC theory also uses the FOL theory, but we are only suggesting the symbols and axioms that the theory consists of. For the purpose of our example we do not need a rigorous representation. We will only use the fact that real numbers and matrices can be defined using ZFC.

3 Representing Functors

3.1 Categories and MMT

We will define a functor between two categories and explain where the problem occurs in representing this. We will use [9] for the theoretical background.

Definition 1. *Let C and C′ be two categories. A* ***functor*** *F from C to C′ is a mapping that*

1. *associates to each object $X \in |C|$ an object $F(X) \in |C'|$,*
2. *associates to each morphism $m : X \to Y \in |C|$ a morphism $F(f) : F(X) \to F(Y) \in |C'|$ such that the following conditions hold(functors must preserve identity morphisms and composition of morphisms):*
   - *$F(id_X) = id_F(X)$ for every object $X \in C$*
   - *$F(g \circ f) = F(g) \circ F(f)$ for all morphisms $f : X \to Y$ and $g : Y \to Z$.*

Take a functor

$$F : \mathsf{Monoid} \to \mathsf{Group}$$

defined between the category Monoid and the category Group. This means the functor associates to object $M \in |\mathsf{Monoid}|$ some other object $F(M) \in |\mathsf{Group}|$ and morphisms $m : M \to M'$, where $M, M' \in |\mathsf{Monoid}|$ are associated to morphisms $F(m) : F(M) \to F(M')$.

Functors and translations of morphisms are encountered very often in mathematics. One example that we will use throughout the paper is the one that associates to the monoid $\mathbb{R}^{2\times 2}$ of matrices of size 2 the group $GL_2(\mathbb{R})$ of invertible matrices of size 2. Other examples that illustrate such functors are:

$$F : \mathsf{Ring} \to \mathsf{Field},$$

that associates to a ring its $Field\ of\ Fractions$ or

$$F : \mathsf{Ring} \to \mathsf{Ring}$$

that associates to ring its $Ring\ of\ Polynomials$.

We would like to have these type of common mathematical situations represented in the MMT language. For now we will only be concerned about the representation of the *object components* of functors.

Some functors can be represented in MMT using composition. For e.g.:

$$F : \mathsf{Group} \to \mathsf{Monoid}$$

is the forgetfull functor that takes a group and simply "forgets" some of the symbols and axioms, returning the resulting monoid. If we look at Fig. 2 this functor can be represented by mapping a morphism from $Group \to ZFC$ to a morphism $Monoid \to ZFC$:

$$m' \mapsto mon \bullet m'.$$

However, we would also like to represent the inverse functor. We will describe our proposed solution for representing functors between objects of a category. MMT only allows us to give morphisms between theories, which means only mapping symbols and axioms. It does not support mapping of morphisms to other morphisms. This is a limitation of MMT that we want to overcome and for which we will give us functor representation.

3.2 Intuition Behind the Solution

Let $M = \mathbb{R}^{2\times 2}$, the square matrices of size 2 over $\mathbb{R}$, a functor F : Monoid→Group and $G = F(M) = GL_2(\mathbb{R})$, the invertible matrices of size 2 over $\mathbb{R}$.

We want to have a representation function R from category theory to the MMT language. We want to find a representation of the functor F in MMT. Model categories are represented as MMT-theories and models are represented as morphisms from theories into ZFC. If we consider the representation of $\mathbb{R}^{2\times 2}$ and $GL_2(\mathbb{R})$ in MMT, these elements are the representation of the $Monoid$, respectively $Group$ axioms in ZFC (see Fig. 3). They are represented through the morphisms:

$$\mathbb{R}^{2\times 2} = m : Monoid \to ZFC,$$

$$GL_2(\mathbb{R}) = m' : Group \to ZFC.$$

To provide the representation of the functor F we have to provide a morphism that takes m and returns m' and that will thus have the type:

$$R(F) : (Monoid \to ZFC) \Rightarrow (Group \to ZFC).$$

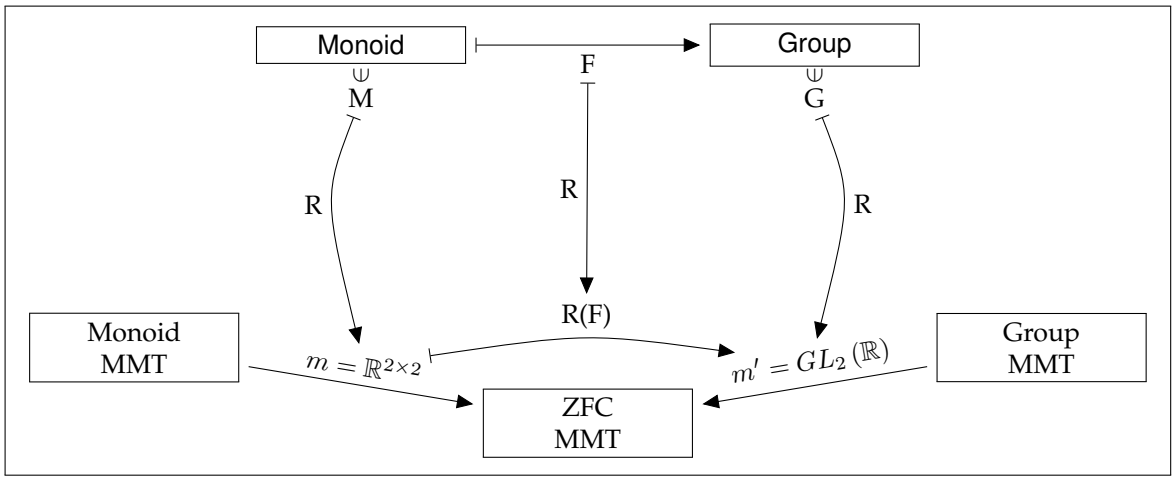

Figure 3: Functor Representation

Our proposed solution of representing a functor in MMT follows the idea of having functions that can take morphisms as parameters. The solution is motivated by λ-calculus. As illustrated in Fig. 3, $R(F)$ is the representation of the functor F. $R(F)$ is represented as a λ-abstraction that takes morphisms of the type $Monoid \to ZFC$ as an argument and returns morphisms of the type $Group \to ZFC$:

$R(F)$:=
$\lambda m : Monoid \to ZFC.($
$universe \mapsto \{x \in universe^m\},$
$e \mapsto e^m,$
$e_l \mapsto axiom\ for\ left\ unit,$
$e_r \mapsto axiom\ for\ right\ unit,$
$inv \mapsto the\ unique\ existing\ inverse\ in\ universe^m$

$inv_l \mapsto axiom\ for\ left\ inverse$
$inv_r \mapsto axiom\ for\ right\ inverse$
$\circ \mapsto \circ^m$,
$assoc \mapsto assoc^m$,
$) : Group \to ZFC$

where we have assumed that the matrix structure is defined on ZFC. Once we have assumed this, the unit or inv are representable through sets from ZFC.

We can afterwards apply the functors to a morphism. This is done by β-reduction. In our previous example, by applying the functor to $\mathbb{R}^{2\times 2}$ we are substituting in the abstraction all the occurances of morphism m with $\mathbb{R}^{2\times 2} : R(F)(\mathbb{R}^{2\times 2}) = (above\ definition)[m/\mathbb{R}^{2\times 2}]$. On the left column we wrote a mathematical equivalent of $R(F)(\mathbb{R}^{2\times 2})$ which is more intuitive:

$R(F)(\ \mathbb{R}^{2\times 2}\ ) := ($
$universe \mapsto \{x \in universe^{\mathbb{R}^{2\times 2}}\}$,
$e \mapsto e^{\mathbb{R}^{2\times 2}}$,
$e_l \mapsto axiom\ for\ left\ unit$,
$e_r \mapsto axiom\ for\ right\ unit$,
$inv \mapsto unique\ existing\ inverse\ in\ universe^{\mathbb{R}^{2\times 2}}$
$inv_l \mapsto axiom\ for\ left\ inverse$
$inv_r \mapsto axiom\ for\ right\ inverse$
$\circ \mapsto \circ^{\mathbb{R}^{2\times 2}}$,
$assoc \mapsto assoc^{\mathbb{R}^{2\times 2}}$,
$) : Group \to ZFC$

$mathematical\ equivalent := ($
$universe \mapsto \{x \in GL_2(\mathbb{R})\}$,
$e \mapsto I_2$,
$e_l \mapsto axiom\ for\ left\ unit$,
$e_r \mapsto axiom\ for\ right\ unit$,
$inv \mapsto unique\ existing\ inverse\ in\ GL_2(\mathbb{R})$
$inv_l \mapsto axiom\ for\ left\ inverse$
$inv_r \mapsto axiom\ for\ right\ inverse$
$\circ \mapsto *$,
$assoc \mapsto assoc^{\mathbb{R}^{2\times 2}}$,
$) : Group \to ZFC$

3.3 Syntax

We will for now consider a reduced grammar of the MMT language. It is based on the flat MMT grammar. This means the initial MMT grammar is equivalent with the flattened one. However it will be easier to add new productions and set rules for well formed expressions on a simpler grammar and then extend it. In particular we omit imports. Our changes are orthogonal to the omited MMT concepts, such that our solution can be extended to the general case easily.

In Fig. 4 we give the production rules that specify the formal syntax of the reduced grammar along with the extension we are adding. We are introducing λ-abstractions in the Viw rule and adapting the others. We tried to make as little modifications as possible to the initial grammar in order to integrate the language we are constructing into MMT more easily. We have marked up the modified productions in red.

The grammar is structured starting with the level of the *theory graph*. This consist of a sequence of *theory declarations* and *view declarations*. We will later find that these morphisms are previously declared as views (in the rules for well formed expressions). Theory declarations consist of *theory bodies*. The theory body is a succession of symbols and axioms . A view declaration is a typed morphism defined by a *substitution*. A substitution provides an assignment of T-objects to S-symbols. *Morphisms* can be views, the composition of two morphisms, or the application of a view to a morphism.

We added the production that presents the most interest is using views to provide a functor between the representation of objects belonging to categories. Through a λ-expression the substitution can be seen as a map between morphisms. This is where the utility of typed morphisms is shown. Unlike MMT where morphisms are not typed, now the type of the morphism is given by the theories between which it is defined. If the substitution takes a morphism of type τ and returns a morphism

of type τ' the associated type of the morphism represented by such a substitution is $\tau \Rightarrow \tau'$. λ-expressions are the elements for representing functors in MMT. For morphism applications we use β-reduction and replace all occurances of morphisms in the abstraction. We will not detail this, we just consider β-reductions to be a case of substitution.

Next we will give a more rigorous description of the grammar.

Theory graphs	γ	::=	$. \mid \gamma, Tth \mid \gamma, Viw$
Theory declarations	Tth	::=	$T := \{\vartheta\}$
View declarations	Viw	::=	$m := \{\sigma\} : \tau$
Type	τ	::=	$(S \to T) \mid \tau' \Rightarrow \tau$
Substitutions	σ	::=	$(c \mapsto \omega)^* \mid \lambda x : \tau.\sigma$
Theory bodies	ϑ	::=	$. \mid \vartheta, c := \omega : \omega$
Morphisms	μ	::=	$m \mid \mu \bullet \mu' \mid m(\mu)$
Auxiliary morphisms	Γ	::=	$. \mid (x : S \to T)^*$
Objects	ω	::=	$c \mid \nu \mid \omega^\mu \mid @\left(\omega\,[,\omega]^+\right) \mid \beta(\omega, \Upsilon, \omega_2, \omega_3)$

Figure 4: The grammar for the expressions

Theory graphs γ - represent directed acyclic multigraphs where the nodes are the theories and the edges are views. They are given as sequences of theory and view definitions.
Theory declarations Tth - is of the form T defined by a sequence of declarations ϑ, which is called body of the theory.
View declarations *Viw* - declares a morphism declaration m that is defined by a substitution σ and has an associated type τ . If τ is of the form $S \to T$, the substitution is defined between S and T. If τ is of the form $\tau' \to \tau''$ the substitution is defined from the morphisms of type τ' to morphisms of type τ''.
Type τ - expresses the theories between which the substitution is defined, in the base case $(S \to T)$, or $(S \to T) \Rightarrow (S' \to T')$ when the substitution takes as argument a morphism $S \to T$ and returns a morphism $S' \to T'$ and so on.
Substitutions σ - In the base case the substitution is a list of mappings of symbols to terms. The λ abstraction $\lambda x : (S \to T).\{\sigma\} : S' \to T'$ reflects that a substitution $S \to T$ can take as argument the substitution $S' \to T'$. This is the representation of a functor between the objects of a category. The objects are represented as the morphisms between theories and the functor is represented as a substitution between some assumed morphism of type τ and another morphism of type τ'.
Theory bodies ϑ - are a succession of symbol names c that are typed and defined by OpenMath objects.
Morphisms μ - represent the paths in a graph. They can represent a view, a composition of morphisms or a view applied to a morphism. In the previous example of a functor between $Monoid$ and $Group$, $m(\mu)$ represents $R(F)$ that takes the morphism $\mathbb{R}^{2\times 2} : Monoid \to ZFC$ to $GL_2(\mathbb{R}) : Group \to ZFC$.
Auxiliary morphisms Γ - the declarations from Γ are morphisms between theories. This is a production rule that will be usefull when establishing the well-formedness rules of the expressions.

Objects - are a generalization of OpenMath objects. They can be:
constants - c ,
variables - $\nu \in Var$ for a fixed countable set of variable names,
morphism application - ω^μ of the morphism μ to ω
applications - $@\left(\omega\,[,\omega]^+\right)$ of ω to arguments ω_i,

bindings - $\beta\,(\omega, \Upsilon, \omega_2, \omega_3)$ a binder ω of a list of variables Υ with a condition ω_2 and body ω_3

4 Well-formed Expressions

In this section we present an inference system to define the well-formed or valid expressions. We will say that an expression is **well-formed** or **valid** if it is considered semantically meaningful, that is if its validity can be proven by the inference system provided for our production rules.

We will list the **judgements** about the expressions that we can define in Fig. 5. The judgements can be used in the inference rules for valid expressions. The judgements use contexts of the form γ or $\gamma;\Gamma$ where γ is a succession of theory and views declarations,

$$\gamma = (T := \{\vartheta\}\,|m := \{\sigma\} : \tau)^* \qquad \Gamma = (x : S \to T)^*$$

and the declarations from Γ are morphisms that were already established as well-formed.

Judgement	Intuition
$\rhd\gamma$	γ is a well formed theory graph.
$\gamma \rhd Dec$	γ can be continued with the declaration Dec (theory or view).
$\gamma \rhd^{T} Dec$	the last theory in γ , named T, can be continued with the declaration Dec .
$\gamma;\Gamma \rhd \sigma : \tau$	σ is a well formed view in the context $\gamma;\Gamma$.
$\gamma;\Gamma \rhd \mu : \tau$	μ is a well formed morphism in the context $\gamma;\Gamma$.
$\gamma;\Gamma \rhd_{T} \omega$	ω is a well formed object in the context $\gamma;\Gamma$ and with home theory T.
$\gamma \rhd^{\tau} \tau$	τ is a well formed type in a context that includes γ.

Figure 5: Main Judgements

The rules for construction of theory graphs are illustrated in Fig. 6.

$$\frac{}{\rhd \cdot}\,(Gph_{\varnothing}) \qquad \frac{\rhd\gamma \quad \gamma \rhd Dec}{\rhd \gamma, Dec}\,(Gph_{Dec})$$

$$\frac{\rhd\gamma \quad T\ new\ in\ \gamma}{\gamma \rhd T := \{\cdot\}}\,(Th_{\varnothing}) \qquad \frac{\rhd\gamma \quad \gamma, T := \{\vartheta\} \rhd^{T} Dec}{\gamma \rhd T := \{\vartheta, Dec\}}\,(Th_{Dec})$$

$$\frac{\rhd\gamma \quad \gamma \rhd \sigma : \tau \quad m\ new\ in\ \gamma \quad \gamma \rhd^{\tau} \tau}{\gamma \rhd m := \{\sigma\} : \tau}\,(Viw_{\sigma})$$

$$\frac{x : \tau\ in\ \Gamma}{\gamma;\Gamma \rhd x : \tau}\,(\Gamma_{add})$$

Figure 6: Structure of Theory Graphs

Rule ($Gph_{\varnothing}$) allows the definition of an empty theory graph. Rule (Gph_{Dec}) allows the construction of a theory graph by adding new declarations Dec if their well formedness has been established.

Dec can be theory level statements or view statements

$$T := \{\vartheta\} \qquad or \qquad m := \{\sigma\} : \tau.$$

Rule ($Tth_{\varnothing}$) says that a new empty theory can be added to the graph. Rule (Tth_{Dec}) allows the addition of a new declaration Dec to a theory T, if the well formedness of Dec has been established.

Rule (Viw_{σ}) adds a view to the theory graph given that its name is new and the substitution that defines it was already correctly defined. Rule (Γ_{add}) allows the use of morphism under the context $\gamma; \Gamma$ if the morphism was previously declared in Γ.

The well-formedness of types is given through rules (Typ_{base}) and (Typ) in Fig. 7. Rule (Typ_{base}) allows the use of a type if the theories on which the morphism is declared are well defined. Rule (Typ) allows the use of type $\tau \Rightarrow \tau'$ only if the types τ or in τ'. However, for representing the object component of functors we are only concerned of types such as $(S \to T) \Rightarrow (S' \to T')$.

$$\frac{T,\ S\ in\ \gamma}{\gamma \rhd^{\tau} S \to T}\ (Typ_{base}) \qquad \frac{\gamma \rhd^{\tau} \tau \quad \gamma \rhd^{\tau} \tau'}{\gamma \rhd^{\tau} \tau' \Rightarrow \tau}\ (Typ)$$

Figure 7: Structure of Types

The rules for the construction of views are given in Fig. 8. Rule (Viw_{base}) defines the well-formedness of the base substitution case. We can map symbols c_i to ω_i objects given they are well-formed over S and T. We used $dom(S)$ to denote the undefined symbols declared in S. We omit the conditions ensuring that all ω_i are well-typed with respect to the types of the c_i. Rule (Viw_{λ}) defines the correctness of a substitution using λ-abstraction. It is based on the corresponding rule from λ-calculus.

$$\frac{\gamma; \Gamma \rhd_T \omega_i \quad dom(S) = \{c_1, ..., c_n\}}{\gamma; \Gamma \rhd c_1 \mapsto \omega_1, ..., c_n \mapsto \omega_n : S \to T}\ (Viw_{base})$$

$$\frac{\gamma; \Gamma, x : \tau' \rhd \sigma : \tau}{\gamma; \Gamma \rhd \lambda x : \tau'.\sigma : \tau' \Rightarrow \tau}\ (Viw_{\lambda})$$

Figure 8: Structure of Views

The rules for the construction of morphisms are illustrated in Fig. 9. We will take as a hypothesis a morphism between theories S and T. Rule (M_{Viw}) handles the definition of a morphism m if it was previously declared as a view. Rule ($M_{\bullet}$) gives the rule for morphism composition. Rule (M_{ViwApp}) gives the correctness of views applied to morphisms.

The well formed terms follow the same rules as in the MMT language, thus the OpenMath standard. They are well formed relative to a theory T and the context γ. The rules are shown in Fig. 10. Rule (Con) says that a symbol is well formed under a theory T if it is in the symbol declarations of the theory. Rule (@) gives the rule for the application of ω_1 to the arguments that follow it. Rule (M_{app}) handles the correctness of the morphism application of μ to ω, which allows to move objects

of S to T along a morphism. We are refering to the MMT paper for the rules of well formedness for variables and bindings.

$$\frac{m := \{\_\}\, S \to T \; in \; \gamma}{\gamma; \Gamma \rhd m : S \to T}\,(M_{Viw}) \qquad \frac{\gamma; \Gamma \rhd \mu : R \to S \quad \gamma; \Gamma \rhd \mu' : S \to T}{\gamma; \Gamma \rhd \mu \bullet \mu' : R \to T}\,(M_{\bullet})$$

$$\frac{\gamma; \Gamma \rhd m : \tau' \Rightarrow \tau \quad \gamma; \Gamma \rhd \mu : \tau'}{\gamma; \Gamma \rhd m(\mu) : \; \tau}\,(M_{ViwApp})$$

Figure 9: Morphisms

$$\frac{T := \{\vartheta\}\; in\; \gamma \quad c : \_ = \_\; in\; \vartheta}{\gamma; \Gamma \rhd_T c}\,(Con)$$

$$\frac{\gamma; \Gamma \rhd_T \omega_i}{\gamma; \Gamma \rhd_T @\,(\omega_1, ..., \omega_n)}\,(@) \qquad \frac{\gamma; \Gamma \rhd_S \omega \quad \gamma; \Gamma \rhd \mu : S \to T}{\gamma; \Gamma \rhd_T \omega^{\mu}}\,(M_{app})$$

Figure 10: Well-formed Terms

5 Conclusion and future work

Our work focused on representing functors in a web-scalable module system. Therefore the MMT language modular structure makes it easy to represent mathematical theories and morphisms between theories. Since the current structure of MMT allows the representation of only a few functors, sing morphism composition, we are proposing a new solution for representing functors.

We used λ-calculus to represent functors in the MMT language. Our solution is built upon a simplified version of the MMT grammar. The solution for representing functors uses λ-abstractions to represent them. Since we can represent category objects through morphisms between theories, functors are just λ-abstractions that take the morphisms as arguments. The new productions are integrated in the flat grammar of MMT which makes the solution scalable to the entire MMT language.

We have defined defined rules for well formed expressions in the new language. Some of the rules from the MMT language are adjusted to fit with the structure of the extended grammar. After defining the well formedness rules we can represent functors between model categories that can be represented in the structure of theories from MMT.

The next step is to integrate functors with the entire MMT language. We will first add imports to our current rules and than integrate all the other rules from MMT. The final purpose is to integrate the modifications made on the reduced grammar to the entire MMT language. Since we worked on the flat grammar, this is possible and requires the adjustment of the language to the new productions.

Our work needs to be continued towards studying the possible integration of the entire functor components. Respectively we will attempt to represent also the morphism component of the functors. We will also consider making λ-abstractions over parameters. For e.g. in our running example, we would like to take as a parameter not only the morphism $\mathbb{R}^{2\times 2} : Monoid \to ZFC$, but also to take

as a parameter the size of the matrix we are considering, such as n, a parameter when considering $\mathbb{R}^{n\times n}$ matrices.

References

[1] Florian Rabe, Michael Kohlhase. A Web-Scalable Module System for Mathematical Theories. *to be submitted*, 2008.

[2] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

[3] William M. Farmer and Joshua D. Guttman and F. Javier Thayer. Little Theories. In *Conference on Automated Deduction*, pages 567–581, 1992.

[4] Bourbaki Nicolas. *Theorie des ensembles*. 1970.

[5] Michael Kohlhase. OMDoc: An Open Markup Format for Mathematical Documents (Version 1.2). *Number 4180 in Lecture Notes in Artificial Intelligence*, 2006.

[6] S. Owre and N. Shankar and J.M. Rushby and D.W.J. Stringer-Calvert. PVS Language Reference, 2001.

[7] Clemens Ballarin. Locales and Locale Expressions in Isabelle/Isar. *Third International Workshop of the Types Working Group, TYPES 2003*.

[8] Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume 1. North Holland, 1958. Second edition, 1968.

[9] Steve Awodey. *Category Theory (Oxford Logic Guides)*. Oxford University Press, USA, July 2006.