Development of a Portion of a Theory Library for Mechanized Mathematics Systems

Development of a Portion of a Theory Library for Mechanized Mathematics Systems

By
Mehwish Abbasi, M.Sc. Applied Mathematics

A Thesis
Submitted to the School of Graduate Studies
in partial fulfilment of the requirements for the degree of

Master of Science
Department of Computing and Software
McMaster University

MASTER OF SCIENCE (2009) McMaster University
(Computer Science) Hamilton, Ontario

TITLE: Development of a Portion of a Theory Library for Mechanized Mathemaics Systems

AUTHOR: Mehwish Abbasi, M.Sc. Applied Mathematics (University Of Karachi)

SUPERVISOR: Dr. William M Farmer

NUMBER OF PAGES: viii, 60

ABSTRACT

A theory library for a mechanized mathematics system (MMS) is a collection of mathematical theories which serves as a database of mathematics. A powerful library plays a significant role in making an MMS useful. This thesis demonstrates some of the techniques needed for generating a large theory library for an MMS, that has capability of both computation and deduction, by developing a small portion of a theory library. In the theory library presented in this thesis, the module system *Mei*, a λ-calculus style module system that supports higher-order functors, is employed to manage mathematical theories. *Chiron*, a logic derived from von-Neumann-Bernays-Gödel set theory, is used as the underlying logic of the system, and *biform theories*, which can include both formulas and algorithms as axioms, are used to present mathematical theories. The theory library given in this thesis is based on the branch of mathematics called calculus.

ACKNOWLEDGMENTS

First of all, I would like to express my deep and sincere gratitude to my supervisor Dr. William M Farmer for his support, guidance, patience, worthy suggestions and encouragement throughout my two years of study at McMaster. He taught me tons of new things. Without his help it would not have been possible to write this thesis.

I am grateful to the examination committee members, Dr. Jacques Carette and Dr. Christopher Anand, for their valuable suggestions.

I would like to thank my husband, Iftikhar, for standing by me and encouraging me, my parents for their love, efforts in raising me and unstinting support in every part of my life. I would not be what I am today without them. Lastly, I would like to thank my sister Sana and brother Adil for helping me whenever I need them.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Mechanized Mathematics Systems

A mechanized mathematics system (MMS) is a software system that is intended to manage, automate, and improve parts of the mathematics process. The three main kinds of MMSs are:

- Computer theorem proving systems (CTPSs): These include automated theorem provers, for example, Vampire [28], interactive theorem provers or proof assistants, for example, IMPS [17] and proof checkers, for example, Logic Daemon [20].

- Computer algebra systems (CASs): These are software programs that performs symbolic computation, for example, Maple [7], Axiom [18] and Mathematica [1].

- Interactive mathematics laboratories (IMLs): Researcher are working on this kind of MMS. They do not exist at present, but they could have great impact on the way people do Mathematics [10].

In CTPSs mathematics is represented by axiomatic theories, where an axiomatic theory represents mathematical knowledge declaratively as a set of axioms. In these systems, reasoning is performed by proving conjectures. They emphasize proof checking or proof development. Their upsides are that they are based on rigorous logical foundations and support a wide range of mathematics. The downsides of CTPSs are that they are very difficult to use, they have poor support for routine computations and abstract theories are emphasized over concrete structures.

In CASs mathematics is represented by algorithmic theories, where an algorithmic theory represents mathematical knowledge procedurally as a set of algorithms. In contrast to CTPs, in these systems reasoning is performed by computation. Their advantages are that they perform quickly, carry out sophisticated symbolic computations and are relatively easy to use. Their disadvantages are that they have poor support for context guided computation and, unlike CTPSs, they are not based on a rigorous logical foundation and concrete structures are emphasized over abstract theories.

Nowadays much work is being done for developing new MMSs and improving the design of MMSs. Many conferences are held every year in which scientists from all over the world discuss and share new ideas for making MMSs better. Examples are Calculemus [3], Mathematical Knowledge Management [23], Modules and Libraries for Proof Assistants (MLPA) [24], the Conference on Autometed Deduction (CADE) [2], etc.

1.2 Objective and Approach

An MMS is not very beneficial without an extensive library. In an MMS, a mathematical library has to be well organized and developed in such a way so that new knowledge can be easily added to it and theories can be easily connected to each other. For expressing theories in MMS, we need an expressive and practical logic. Furthermore, in contemporary CTPSs mathematical knowledge is represented by axiomatic theories and deduction is supported. On the other hand, in CASs mathematical knowledge is represented by algorithmic theories and computation is supported. We want to have a system in which the reasoning engine has both deduction and compu-

tation. In order to have it we need to merge CASs and CTPSs and this can be done by having a library in which mathematical knowledge is represented by a combination of algorithmic and axiomatic theories.

This thesis illustrates, by developing a portion of theory library, how to build a theory library for an MMS that is well structured, whose underlying logic is practical, and has both the capabilities of computation and deduction. This is done by employing the module system Mei [32], proposed by Jian Xu for organizing mathematical knowledge in theory library of MMSs. Biform theories, which are both algorithmic and axiomatic theories together, are used as a way of representing theories and Chiron is used as an underlying logic of the library because biform theories are easily expressed in this logic. To put it simply, the theory library developed in this thesis represents mathematical knowledge as biform theories expressed in Chiron and based on Mei.

This research is conducted as part of MathScheme project [21] at McMaster University. MathScheme is a project whose aim is to build a mechanized mathematics system that integrates computer algebra and computer theorem proving. Its ultimate goal is to develop an interactive mathematics laboratory. The module system Mei, logic Chiron and bifrom theories all were developed as part of this project.

1.3 Organization of the Thesis

This thesis illustrates how to build a theory library of a MMSs that has the capabilities of both deduction and computation. It is organized as follows:

Chapter 2 (Background) goes over the module system *Mei* which is chosen as a system to organize mathematical library, the logic *Chiron* which is used as the underlying logic of the library and *biform theories* which are employed as a way to represent theories. It also describes what mathematical knowledge management means.

Chapter 3 (Forms of conservative extensions) presents several extensions to Mei. Each of these extensions introduces a theory extension that is conservative provided a certain obligation is satisfied.

Chapter 4 (Starting Theories) demonstrates how we can use the extensions described in the previous chapter to build theories from existing theories.

Chapter 5 (Theory Building Tools) illustrates functors as another way to develop theories from existing theories. All the notions in these functors belongs to calculus of one variable.

Chapter 6 (Sample Library) portrays how we can employ the functors shown in the last chapter to produce a theory of real numbers.

Chapter 7 (Comparison to Previous Work) discusses that which other systems has used the ideas presented in this thesis.

Chapter 8 (Conclusion) summarizes what is achieved in this thesis.

Chapter 9 (Future Work) sketches how the work presented in this thesis can be used in the future including an idea to build a whole mathematical library based on techniques this thesis exhibits.

CHAPTER 2

BACKGROUND

2.1 Mathematical Knowledge Management (MKM)

In [11] Dr. William Farmer has described MKM as follows

> MKM is a new interdisciplinary field of research in the intersection of mathematics, computer science, library science, and scientific publishing. The objective of MKM is to develop new and better ways of managing mathematical knowledge using sophisticated software tools. MKM is expected to serve mathematicians, scientists, and engineers who produce and use mathematical knowledge; educators and students who teach and learn mathematics; publishers who offer mathematical textbooks and disseminate new mathematical results; and librarians and mathematicians who catalog and organize mathematical knowledge.

Mathematical knowledge management (MKM) is a new field of research. It has largely developed in the last few years. Its aim is to develop new and better ways for organising mathematical knowledge using sophisticated tools. It is hard to organize mathematical knowledge and it is much different from other sorts of knowledge. Mathematical knowledge management has been an issue within the community of mathematicians for centuries, but now mathematicians and mathematics practitioners are more involved in it than ever before.

Before the information age, mathematical knowledge used to be stored in textbooks and journals, but the classical method of organizing mathematical knowledge is not sufficient any more and new methods are needed. Nowadays non mathematicians are also producing mathematical knowledge. We believe that there should be a place where all mathematical knowledge is stored.

Mathematical knowledge is abstract, universal, highly structured, extraordinarily interconnected, and of immense size and these attributes of mathematical knowledge make it different from other kind of knowledge [12]. For managing mathematical knowledge we need tools and methods that are different from the tools and methods required for other sorts of knowledge. The organization of mathematical knowledge can be divided into four activities, i.e. articulation, organization, dissemination and access [11].

There is an MKM research group which organizes conferences on Mathematical Knowledge Management. The work of this thesis, building a library of mathematical knowledge for MMSs, is one of the areas that MKM community is concerned about; for details see [6]. The grand challenge of MKM is to build a universal digital mathematical library (UDML).

2.2 Chiron

A logic that is both theoretically and practically expressive is required by a practical, general purpose MMS. Traditional logics such as first-order logic or simple type theory (classical higher-order logic) are not suitable for this purpose due to the absence of the practical expressivity needed for an MMS. They are designed to be used in theory, not in practice.

Chiron is a logic developed by W.M Farmer [14]. It is derived from von-Neumann-Bernays-Gödel (NBG) set theory which is a conservative extension of Zermelo-Fraenkel set theory that is designed to be a practical, general purpose logic. As it is derived from NBG set theory, it is based on familiar principles from predicate logic, set theory, and type theory. It has the same theoretical expressivity as ZF and NBG set theories, but it is much practically expressive than traditional logics. It is designed to be a

logical foundation for mechanized mathematics. It supports in an integrated manner five reasoning paradigms [15] that are commonly employed either in mathematical practice or in contemporary MMSs:

(1) Classical.

(2) Permitted undefinedness.

(3) Set theory.

(4) Type theory.

(5) Formalized syntax.

As described in [14], in Chiron, operators, terms, types and formulas are all proper expressions, while improper expressions are nondenoting (i.e., they do not denote anything). Operators are used to construct expressions. They denote operations. Types are used to restrict the values of operators and variables and to classify terms by their values. They denote superclasses. Terms are used to describe classes. They denote classes or the undefined value. Formulas are used to make assertions. They denote truth values. (illustrated in figure 2.1). The values for intuitively nondenoting types, terms, and formulas are the default values D_c (universal superclass), $\bot$, and F, respectively.

In Chiron a theory is (L, Φ, Δ) where L is a language (set of operators), Φ is a pair (Γ, Π) where Γ is a set of axioms and Π is a set of base transformers and Δ is a pair (Ψ, Σ) where Ψ is a set of theorems and Σ is a set of derived transformers. In Chiron, a language is only a set of operators.

Two main features of Chiron are that it can handle undefinedness and can reason about the syntax of expressions. Chiron reasons about syntax by using quotation and evaluation.

We are using Chiron because we need a logic that is designed to be more practical than traditional logics. Biform theories as well as algorithmic theories are difficult to formalize in a traditional logic without the means to reason about syntax. We have used Chiron as the underlying logic because biform theories can manipulate the

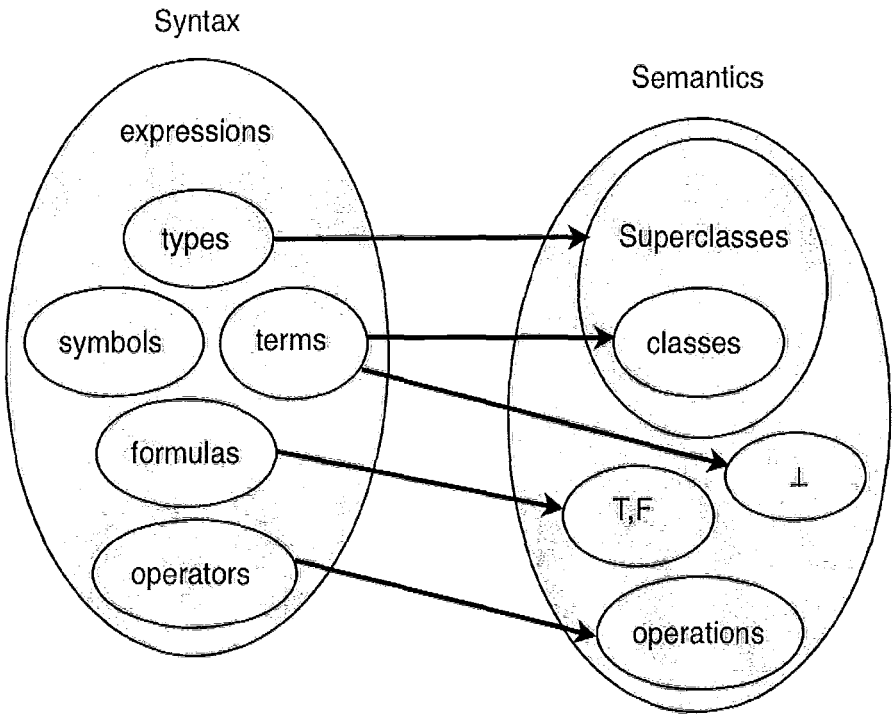

Figure 2.1: Syntax and semantics of Chiron.

syntax of expressions and for that we need a logic that can deal with the syntax of expressions. For this reason, it is an exceptionally well-suited logic for formalizing biform theories.

2.3 Biform Theories in Chiron

An *axiomatic theory* is a pair of a language and a set of sentences. Mathematical knowledge is expressed in an axiomatic theory declaratively as a set of axioms whereas an *algorithmic theory* is a pair of a language and a set of transformers. Mathematical knowledge is expressed in an algorithmic theory procedurally as a set of algorithms. The union of axiomatic theories and algorithmic theories gives biform theories. Mathematical knowledge is expressed in a biform theory both declaratively and procedurally.

A biform theory T [5] is a triple $T = (\varepsilon, \tau, \Gamma)$ where ε is a set of expressions, τ is a set of transformers for ε, and Γ is a set of formulas in ε.

A *transformer* in L is a pair $(\pi, \widehat{\pi})$ such that π is an operator $(O :: \mathsf{E}, \ldots, \mathsf{E})$ in L (with E occurring more than 0 times) and $\widehat{\pi}$ is an algorithm that implements π. The operator π serves as a name for the algorithm $\widehat{\pi}$.

A *biform theory* in Chiron [13] is a tuple $(L, (\Gamma, \Pi), (\Psi, \Sigma))$, where

(1) L is a language of Chiron.

(2) Γ is a set of sentences of L called *axioms*.

(3) Π is a set of transformers in L called *base transformers*.

(4) Ψ is a set of sentences of L called *theorems*.

(5) Σ is a set of transformers in L called *derived transformers*.

Axioms are the background assumptions and specifications of the operators in L including the operators in Π. The members of Ψ are logical consequences of axioms and may specify transformers Σ. We assume that the axioms are true and base transformers are working correctly whereas the theorems and derived transformers are logical consequences of axioms and base transformers, respectively.

The reason that Chiron has been used as underlying logic of Mei in this thesis is because a mechanized mathematics system that utilizes biform theories to represent mathematics requires a logic in which biform theories can be expressed. The axioms in biform theories that describes what a transformer means is a statement that expresses something both about the syntax of expressions and what the expression means and for that we need a logic that can deal with the syntax of expressions. Traditional logics do not have this facility. Chiron has a facility for reasoning about the syntax of expressions and thus for formalizing biform theories the most appropriate logic is Chiron.
Bringing computer theorem proving and computer algebra together is one of the main objectives of Mathscheme project. In biform theories, both logical and computation coexist and thus it plays a key role in integrating CASs and CTPSs. This is the

reason we have used biform theories in this thesis.

2.4 Mei

A good module system plays a significant role in making mathematical knowledge well organized in an MMS. In this section we will discuss Mei [32, 31], a λ-calculus style module system for organizing mathematical knowledge in MMSs. Mei is the building block of the library proposed in the thesis.

In Mei, mathematical knowledge is organized as modules called theories. A *theory* in Mei is a triple (L, Φ, Δ) where L is a language (a set of symbols used by the theory), Φ is a set of axioms and Δ is a set of theorems provable from the axioms in some sound proof system (Δ can be empty). The theories in Mei are organized according to the little theories method [16]. In Mei, we can use the following ways to build a theory: building a theory from scratch, extension of existing theory, union of existing theories, renaming of a theory.

Mei also supports building a theory from a parametrized theory, called a functor. Theories are objects and parametrized theories are functions. Functors can be defined in terms of arbitrary module expressions over the parameter theories. Functors are instantiated only when the actual parameter theory matches the argument type. An actual parameter is passed to a functor via a mechanism similar to a fitting morphism [32]. Functors in Mei are applicative and can be higher-order.

In Mei, there is a type system to classify module expressions by their values, which are theories or functors. In this thesis, two new operators are added to Mei, type operator and return type operator. *type operator* (type) when applied to a theory $T = (L, \Phi, \Delta)$ returns the type of that theory i.e.

$\mathsf{type}(T)$ *is* (L, Φ)

The *return type operator* (returntype) takes a functor $F : G \to H$ as its argument and gives back the theory H.

$\mathsf{returntype}(F)$ *is* H

By a subtyping mechanism in Mei, an object of one type (subtype) can be treated as an object of another type (supertype). Therefore if the formal parameter of a functor F is a theory T, then theories which are subtype of T can also be taken as the parameter of F. Every theory in Mei effectively has more than one type.

Mei supports a coercion mechanism , which is similar to a fitting morphism. Coercion is done by a coercion functor. A Coercion functor changes the actual parameter in a form that can be accepted by the functor where a view is used to justify how and why an object in the formal parameter can be treated as an object in the actual parameter. *view* is a generalization of theory interpretation and a subtype relation. A view from a theory T to T' is a mapping from the expressions of T (source theory) to the expressions T' (target theory). $\mathsf{view}(T, T', \rho)$ asserts that ρ is a mapping from T to T'. A view from a functor $F : G \rightarrow H$ to $F' : G' \rightarrow H'$ means $\mathsf{view}(\ G, G', \rho)$ and $\mathsf{view}(\ H, H', \rho)$. In Mei, a good theory view is a theory interpretation. If v=(T,T', ρ) is a view from a theory T to T', then its semantic $[\![\mathrm{v}]\!]$ is a functor in Mei called a coercion functor. The instantiation of functor can accept more objects as input with the help of coercion functors.

Mei has adopted the idea of parametrized modules as functors i.e. functions from modules to modules and the idea of higher-order functors from ML-family module systems and the fitting morphism style parameter passing mechanism from algebraic specification languages. It is the first system that integrates higher-order functors with fitting morphisms.

There is an analogy between Mei, which handles theories, and typed λ Calculus, which deals with classical values. The theory types are base types, functor types are function types, module expressions are terms, functor abstractions are λ abstractions, functor applications are function applications and theory operations are extra term contructors.

In this thesis, Mei is used for organizing mathematical knowledge. We will use extension, union, renaming and parametrized theories for building theories. Extension means adding language and axioms in an existing theory to get a new theory. The language L_2 of the new theory contains all the symbols of extended theory; therefore language L_1 of the new theory is a sublanguage of L_2, that is, $L_1 \subseteq L_2$ and axioms

Φ_2 and theorems Δ_2 of new theory comprises the set of axioms Φ_1 and the set of theorems Δ_1 of original theory as well as some new axioms (hence $\Phi_1 \subseteq \Phi_2$). We use conservative extensions to develop a stack of theories. The *union* operation is used to build a theory from two or more existing theories. The union of $T_1 = (L_1, \Phi_1, \Delta_1)$ and $T_2 = (L_2, \Phi_2, \Delta_2)$ is denoted by $T = T_1 \oplus T_2$. T is a theory whose language is the union of L_1 and L_2, the axioms are the union of Φ_1 and Φ_2 and the theorems are the union of Δ_1 and Δ_2.

Once again, the three main components of the proposed theory library are Mei, which is used to organize theories, Chiron, which is used as underlying logic and Biform theories, which is employed to represent theories. Fig 2.2 illustrates the way they are used.

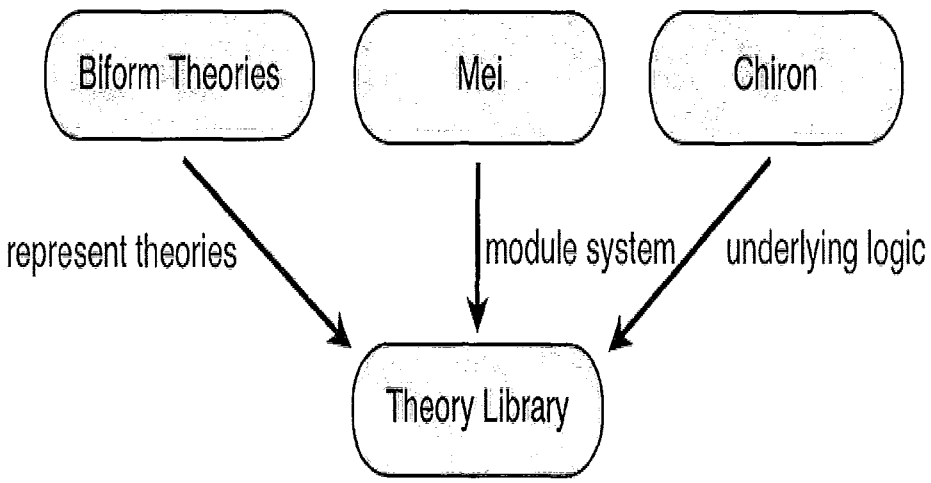

Figure 2.2: Three main components of the proposed theory library.

CHAPTER 3

FORMS OF CONSERVATIVE EXTENSION

Using Mei, we can build a theory either from scratch or we can use previously developed theories to build a new theory. Mei supports theory building operations such as theory extension, union and renaming. The following are the definitions of theory extension and conservative extension.

A theory $T_2 = (L_2, \Phi_2, \Delta_2)$ is an *extension* of another theory $T_1 = (L_1, \Phi_1, \Delta_1)$, written, $T_1 \leq T_2$, if $L_1 \subseteq L_2$, $\Phi_1 \subseteq \Phi_2$ and $\Delta_1 \subseteq \Delta_2$ (see 2.4).

A theory T_2 is a *conservative extension* of another theory T_1 if T_2 is an extension of T_1 and every sentence valid in T_2 that involves only operators of T_1 is also valid in T_1. That is, T_2 adds new machinery to T_1 without compromising the original machinery of T_1.

In this chapter, six special forms of theory extension will be presented. All of these extensions are conservative (provided certain obligations are satisfied). We are interested in conservative extensions of theories because in a conservative extension the new machinery does not compromise the old machinery. All theories and languages in this chapter are theories and languages of Chiron.

3.1 Special Forms of Theory Extension

We will define six important forms of theory extension that are conservative:

(1) **A Set of Theorems.**
This adds a set of theorems to a theory.

(2) **A Set of Definitions.**
This adds a set of new separately defined operators to a theory.

(3) **A Profile.**
This adds a set of new collectively profiled operators to a theory.

(4) **An Inductive Data Type**
This adds a new inductive data type to a theory.

(5) **A Recursive Definition.**
This adds a recursively defined operator to a theory.

(6) **An Interpreted Theory.**
This adds an interpreted theory to a theory.

3.2 Extension with Theorems

This kind of theory extension adds a set of sentences to the set of theorems of a theory T. The set of sentences is specified by Ψ.

Extended Mei Syntax
The extended syntax of Mei that expresses an extension with theorems by Ψ is

T extended with theorems by Ψ

where T is a theory and Ψ is a set of sentences not in the language of T.

Meaning As Official Mei Syntax Using Biform Theories In Chiron
The syntax

T extended with theorems by Ψ

means in Mei

T extended by $(\emptyset, (\emptyset, \emptyset), (\Psi, \emptyset))$

As described earlier in section 2.3, a biform theory in Chiron has the form (L, Φ, Δ). Here Φ is $(\emptyset, \emptyset)$ as there are neither new axioms nor new base transformers and Δ is $(\Psi, \emptyset)$ as there are new theorems but no new derived transformers.

Syntactic Form

Let T be a theory. The extension T' of T with a set of theorems is presented by the following syntactic form:

$T' = T$ extended with theorems by
Theorems
A_1
$\vdots$
A_n

This represents T extended with theorems by Ψ where $\Psi = \{A_1, A_2, \ldots, A_n\}$.

Conservativity

Let T' be T extended with theorems by Ψ where $\Psi = \{A_1, A_2, \ldots, A_n\}$. The *obligation* B of this extension is

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n$$

The new theory T' is a conservative extension of T if B is valid in T, that is, if $T \models B$.

Example

RealNumG′ $\equiv$ **RealNumG** extended with theorems by
Theorems
$\forall a, b : \mathsf{rr} \,.\, a < b \supset \exists c : \mathsf{qq} \,.\, a < c \wedge c < b$
$\forall a, b : \mathsf{rr} \,.\, a > 0 \wedge b > 0 \supset \exists c : \mathsf{nn} \,.\, c * a > b$

In the example above, the theory RealNumG which is defined in section 4.8 is extended by two theorems.

3.3 Extension with Definitions

This form of theory extension adds a set of new separately defined operators to the language of a theory T. Each operator can be given a special notation. For each operator there is one or more defining axioms. This extension is specified by (L, Γ), where L is a language of operators to be added to T and Γ is a set of sentences to be added to the axioms of T.

Extended Mei Syntax

The extended syntax of Mei that expresses an extension with definitions by (L, Γ) is

T extended with definitions by (L, Γ)

where T is a theory, L is a language whose members are not in the language of T and Γ is a set of sentences that serves as the defining axioms for these operators.

Meaning As Official Mei Syntax Using Biform Theories In Chiron

The syntax

T extended with definitions by (L, Γ)

means

T extended by $(L, (\Gamma, \emptyset), (\emptyset, \emptyset))$

Here Φ is $(\Gamma, \emptyset)$ as there are only new axioms and no new base transformers, and Δ is $(\emptyset, \emptyset)$ as there are neither new theorems nor new derived transformers.

Syntactic Form

Let T be a theory. The extension T' of T with a set of definitions is presented by the following syntactic form:

$T' = T$ extended with definitions by
Defined Operators
O_1
O_2
$\vdots$
O_n

Notation

N_1 means $O_1(e_{11}, \ldots, e_{1m_1})$

$\vdots$

N_n means $O_n(e_{n1}, \ldots, e_{nm_n})$

Defining Axioms

A_1^1

$\vdots$

$A_{p_1}^1$

A_1^2

$\vdots$

$A_{p_2}^2$

$\vdots$

A_1^n

$\vdots$

$A_{p_n}^n$

This represents T extended with definitions by (L, Γ) where L is $\{O_1, O_2, \ldots, O_n\}$ and $\Gamma = \{A_1^1, A_2^1, \ldots, A_{p_n}^n\}$. The set of operators O_i that are added to the theory T are listed under Defined Operators. The optional notation for the operators is listed under Notation, and the axioms $A_1^i, A_2^i, \ldots, A_{p_i}^i$ under Defining Axioms defines the operator O_i for each i with $1 \leq i \leq n$.

Conservativity

Let T' be T extended with definitions by (L, Γ) where $\Gamma = \{A_1^1, A_2^1, \ldots, A_{p_n}^n\}$. The *obligation* B of this extension states that there exist operations that uniquely satisfy the conjunction of all the axioms:

$$\exists! x_1, x_2, \cdots, x_n.((A_1^1 \wedge A_2^1 \wedge \cdots \wedge A_{p_{n-1}}^n \wedge A_{p_n}^n)[O_1 \mapsto x_1, O_2 \mapsto x_2, \cdots, O_n \mapsto x_n])$$

The new theory T' is conservative if the above obligation is valid in T, that is, if $T \models B$. Notice that B will usually not be directly expressible in Chiron since there are no operation variables in Chiron.

Example

RealNum′ ≡ **RealNumBase** extended with definition by

Defined Operators

(sub :: rr → (rr → rr))

(suc :: rr → rr)

(nn :: type)

Notation

$a - b$ means sub$(a)(b)$

Defining Axioms

sub $= \forall a, b : \mathsf{rr} \,.\, a - b = a + (-b)$

suc $= \lambda x : \mathsf{rr} \,.\, x + 1$

nn $\ll$ rr

$\forall a : \mathsf{rr} \,.\, (a \downarrow \mathsf{nn}) \equiv (a = 0 \vee \exists b : \mathsf{nn} \,.\, \mathsf{suc}(b) = a)$

In the example above, there are three defined operators, sub which finds the difference between two real numbers, suc which gives the successor of a real number, nn which is a type of natural numbers, there is a special notation for sub alone. It is not necessary to have a special notation for every operator. There is one axiom corresponding to each of sub and suc, but there are two axioms for nn.

3.4 Extension with a Profile

This extension adds a set $O_1, \ldots, O_n$ of operators and a profiling axiom A containing $O_1, \ldots, O_n$ to a theory T. Like an extension with definitions, this extension is also specified by a pair (L, Γ) but in this case the language L is a set of profiled operators and Γ is a set $\{A\}$ of single sentence called the *profiling axiom.*

Extended Mei Syntax

The syntax of Mei that expresses an extension with a profile by (L, Γ) is

T extended with a profile by (L, Γ)

where T is a theory, L is a language whose members are not in the language of T and

Γ is a set of a single sentence that serves as the profiling axiom for the operators.

Meaning As Official Mei Syntax Using Biform Theories In Chiron

The syntax

T extended with a profile by (L, Γ)

means

T extended by $(L, (\Gamma, \emptyset), (\emptyset, \emptyset))$

Again like an extension with definitions, Φ is $(\Gamma, \emptyset)$ as there are only new axioms and no new base transformers and Δ is $(\emptyset, \emptyset)$ as there are neither new theorems nor new derived transformers.

Syntactic Form

Let T be a theory. The extension T' of T with a profile is presented by the following syntactic form:

$T' = T$ extended with a profile by
Profiled Operators

O_1
O_2
$\vdots$
O_n

Notation

N_1 means $O_1(e_{11}, \ldots, e_{1m_1})$
$\vdots$
N_n means $O_n(e_{n1}, \ldots, e_{nm_n})$

Axioms

A_1
$\vdots$
A_n

This represents T extended with a profile by (L, Γ) where L is $\{O_1, O_2, \ldots, O_n\}$ and $\Gamma = \{A_1 \wedge A_2 \wedge \cdots \wedge A_n\}$. Under Profiled Operators, there is a set of profiled operators O_i. Under Notation, there is optional special notation for the profiled operators,

and under Axioms, each axiom constrains the value of one or more profiled operators. The profiling axiom A is the conjunction of all these axioms, that is, $A_1 \wedge A_2 \wedge \cdots \wedge A_n$.

Conservativity

Let T' be T extended with a profile by (L, T) where $L = \{O_1, O_2, \ldots, O_n\}$ and $\Gamma = \{A\}$. The *obligation* B for the extension states that there are operations that satisfy the profiling axiom:

$$\exists x_1, \ldots, x_n \,.\, A[O_1 \mapsto x_1, O_2 \mapsto x_2, \cdots, O_n \mapsto x_n]$$

Similar to an extension with definitions, T' is a conservative extension of T if $T \models B$.

Example : Positive Real Numbers

PosRealNum $\equiv$ **RealNum** extended with a profile by

Profiled Operators

(pos-real :: rr)
(neg-real :: rr)

Axioms

(0 < pos-real)
(0 > neg-real)
(pos-real = −(neg-real))

In the example above, there are two profiled operators pos-real and neg-real and there are three axioms. pos-real and neg-real are operators that give back unspecified positive and negative real numbers, respectively. Under Axioms, the first one is a constraint on pos-real, the second is a constraint on neg-real, and the third axiom is a constraint on both the profiled operators.

3.5 Extension with an Inductive Data Type

With this extension a type and a set of constructors that specify an inductively data type are added to a theory. This extension is specified by a language L that includes a type operator representing the set of elements of the inductive data type and a set of operators representing the constructors for the inductive data type. L is added to the language of the theory and a set of sentences idt-ax(L) and idt-th(L) that encode implicit properties of the inductive data type are added to the sets of axioms

and theorems of the theory, respectively. idt-ax(L) are the axioms that say that the constructors are total and whatever they construct is of type α, all the elements of type α are constructed by the constructors (no junk) and each member of α can be constructed by only one constructor (no confusion).

Extended Mei Syntax

The syntax of Mei that expresses an extension with an inductive data type by L is

T extended with an idt by L

where T is a theory and L is a language that includes a type operator not in T and other operators that represent the constructors of the type α. None of these operators are in T.

Meaning As Official Mei Syntax Using Biform Theories In Chiron

The syntax

T extended with an idt by L

means

Textended by $(L, (\text{idt-ax(L)}, \emptyset), (\text{idt -thm(L)}, \emptyset))$

In an extension with an idt, since we have implicit axioms and theorems but no transformers are added to the theory, Φ is $(\text{idt-ax(L)}, \emptyset)$ and Δ is $(\text{idt-thm(L)}, \emptyset)$.

Syntactic Form

Let T be a theory. The extension T' of T with an idt by L has the following syntactic form:

Inductive Data Type

$(s :: \alpha)$

Constructors

O_1

O_2

$\vdots$

O_n

Notation

N_1 means $O_1(e_{11}, \ldots, e_{1m_1})$

$\vdots$

N_n means $O_n(e_{n1}, \ldots, e_{m_n})$

Under Inductive Data Type, there is a type operator, and under Constructors, there are operators that build that data type. Under Notation is special notation for the operators.

Conservativity

Let T' be T extended with an idt by L where $L = \{s :: \alpha, O_1, O_2, \ldots, O_n\}$. The *obligation* B of this extension is a sentence that says that there exists a set S representing the type α and mappings M_i representing the operators O_i for each i with $1 \leq i \leq n$ such that:

- M_i gives a member of S (total) when it is applied to the arguments of O_i.
- Every member of the type α is constructed by one of the mappings M_i (no junk).
- No member of α is produced by more than one mapping M_i (no confusion).

The new theory T' is a conservative extension of T if B is valid in T, that is, T' is conservative if $T \models B$.

Example 1

Natural Numbers with an error element

$T' \equiv T$ extended with an idt by

Inductive Data Type

($\mathsf{N_e}$ :: type)

Constructors

(error :: $\mathsf{N_e}$)

(normal :: N, $\mathsf{N_e}$)

In the example above, N is a type of natural numbers and $\mathsf{N_e}$ is a new type of natural numbers plus an error element. normal "tags" a natural number as a "normal" number of $\mathsf{N_e}$. error denotes the error element of $\mathsf{N_e}$.

Example 2

Integers
$T' \equiv T$ extended with an idt by

Inductive Data Type
(Z :: type)
Constructors
(posint :: $\mathsf{N}_{\mathsf{pos}}$, Z)
(negint :: $\mathsf{N}_{\mathsf{pos}}$, Z)
(zero :: Z)

In the example above, Z is a new inductive data type of integers, $\mathsf{N}_{\mathsf{pos}}$ is a type of positive natural numbers. Its members are constructed by three constructors posint, negint and zero. posint takes an element of type natural number and gives back an element of type positive integer, negint maps an element of type natural number to an element of type negative integer and zero is a 0-ary operator that gives the number zero of type integer.

3.6 Extension with a Recursive Definition

In this extension, we add an operator O and a functional F such that the operator O denotes the least fixed point of a functional F. We can also define a system of recursively defined operators from a list of functionals such that the system of operators $(O_1, \ldots, O_n)$ denotes the least fixed point of the list $(F_1, \ldots, F_n)$ of functionals. A *functional* is a mapping that takes a function of type α as its argument and returns a function of the same type α. A *fixed point* of a functional F is a function f such that $F(f) = f$. The *least fixed point* of F is a fixed point of F which is less than or equal to (with respect to the subfunction relation) all other fixed points of F. This extension is specified by an operator O and a functional F. It adds O to the language of T and set of sentences that encode implicit properties of O, rec-ax(O, F) and rec-th(O, F), to the axioms and theorems of T, respectively. rec-ax(O, F) are the axioms that say that the operator O denotes the least fixed point of functional F.

Extended Mei Syntax

The syntax of Mei that expresses an extension with a recursive definition by L is

T extended with a recursive definition by (O, F)

where T is a theory of Chiron, O is an operator, and F is a term of type $\alpha \to \alpha$.

Meaning As Official Mei Syntax Using Biform Theories In Chiron

T extended with a recursive definition by (O, F)

means

T extended by $(\{O\}, (\text{rec-ax}(O, F), \emptyset), (\text{rec-thm}(O, F), \emptyset))$

Like an extension with an inductive data type, we have new axioms and theorems but no new transformers, so Φ is $(\text{rec-ax}(O, F), \emptyset)$ and Δ is $(\text{rec-thm}(O, F), \emptyset)$.

Syntactic Form

Let T be a theory. The extension T' of T with a recursive definitions is presented by the following syntactic form:

$T' = T$ extended with a recursive definition by
Recursively Defined Operator
$(O :: \alpha)$
where α is a function type.

Notation
α means $O(e_1, \ldots, e_m)$

Functional
F

Under Recursively Defined Operator, there is an operator that is defined using F. Under Notation, there can be a special notation for the recursively defined operator and under Functional there is a term F of type $\alpha \to \alpha$.

Conservativity

Let T' be T extended with a recursive definition by (O, F). The *obligation* B of this extension is that the functional F must be monotone. F is *monotone* if it preserves the order of functions, that is

$$\forall f, g : \alpha \,.\, f \sqsubseteq g \supset F(f) \sqsubseteq F(g).$$

T' is conservative extension of T if $T \models B$.

Example 1

Natural numbers
$T' \equiv T$ extended with a recursive definition by
Recursively Defined Operators
(factorial :: (nn $\rightarrow$ nn))

Functional
$\lambda f : (\mathsf{nn} \rightarrow \mathsf{nn}) \,.\, \lambda n : \mathsf{nn} \,.\, \mathsf{if}(n = 0, 1, f(n-1) * n)$

In this example, factorial is an operator recursively defined as the least fixed point of the given functional. Notice that the functional is monotone.

Example 2

Natural numbers
$T' \equiv T$ extended with a recursive definition by
Recursively Defined Operators
(fibonacci :: (nn $\rightarrow$ nn))

Functional
$\lambda f : (\mathsf{nn} \rightarrow \mathsf{nn}) \,.\, \lambda n : \mathsf{nn} \,.\, \mathsf{if}(n = 0, 0, \mathsf{if}(n = 1, 1, f(n-1) + f(n-2)))$

In the example above, fibonacci is an operator recursively defined as the least fixed point of the given functional.

3.7 Extension with an Interpreted Theory

This extension adds to T an interpretation of a theory T^*. The interpretation is a Mei view $\mathsf{view}(T^*, T, \rho)$, where ρ is a mapping from T^* to T.

Extended Mei Syntax

The extended syntax of Mei that expresses an extension with an interpreted theory T^* by $\mathsf{view}(T^*, T, \rho)$ is

T extended with an interpreted theory T^* by $\mathsf{view}(T^*, T, \rho)$

where T and T^* are theories and $\mathsf{view}(T^*, T, \rho)$ is a view from T^* to T.

Meaning As Official Mei Syntax Using Biform Theories In Chiron

The syntax

T extended with an interpreted theory T^* by $\mathsf{view}(T^*, T, \rho)$

means

$T \oplus (T^* \ \mathit{with}\ \rho)$

That is, the union of T and the translation of T^* by ρ.

Syntactic Form

The extension T' of T with an interpreted theory T^* is presented by the following syntactic form:

$T' = T$ extended with an interpreted theory by $T^* \mapsto T$
View
$O_1^* \mapsto O_1$
$O_2^* \mapsto O_2$
$\vdots$
$O_n^* \mapsto O_n$

This represents T extended with an interpreted theory T^* by $\mathsf{view}(T^*, T, \rho)$ where $O_1^*, O_2^*, \ldots, O_n^*$ are all the operators in the language of T^* and $O_1, O_2, \ldots, O_n$ are the operators in the language of T to which the operators of T^* are mapped. If some O^* in the language of T^* is not among $O_1, O_2, \ldots, O_n$, then O^* is mapped to itself.

Conservativity

Let T' be T extended with an interpreted theory T^* by view(T^*, T, ρ). The *obligation* B for the extension states that ρ is an interpretation of T^* in T, i.e.,

$$T^* \models A \supset T \models \rho(A)$$

for all formulas A in the language of T^*.

Example

RealNumK $\equiv$ **RealNumJ′** extended with an interpreted theory by
MonoidalMetric $\mapsto$ **RealNumJ′**

View

ms $\mapsto$ rms

space $\mapsto$ rspace

metric $\mapsto$ rmetric

$\bullet \mapsto +$

e $\mapsto 0$

In the example above, the operators ms, space, metric, $\bullet$ and e belong to the theory **MonoidalMetric** and the operators rms, rspace, rmetric, $+$ and 0 are the operators of theory **RealNumJ′**. The mapping is from the operators of **MonoidalMetric** to **RealNumJ′** (see section 6.4 for details).

CHAPTER 4

STARTING THEORIES

In this chapter we present some theories and their extensions using the forms of theory extensions discussed in the previous chapter.

4.1 Real Numbers Base

The first theory is **RealNumBase**. It is a theory of the real numbers axiomatized as a complete ordered field. The first eleven axioms say that the type of real numbers is a field, the next three axioms make it an ordered field, the next four are definitions and the last axiom is an axiom of completeness.

RealNumBase

Operators

$(\mathsf{rr} :: \mathsf{type})$
$(0 :: \mathsf{rr})$
$(1 :: \mathsf{rr})$
$(+ :: \mathsf{rr} \to (\mathsf{rr} \to \mathsf{rr}))$
$(\mathsf{neg} :: \mathsf{rr} \to \mathsf{rr})$
$(^{-1} :: \mathsf{rr} \to \mathsf{rr})$
$(* :: \mathsf{rr} \to (\mathsf{rr} \to \mathsf{rr}))$
$(<:: \mathsf{rr}, \mathsf{rr}, \mathsf{formula})$

$(\leq :: \mathsf{rr}, \mathsf{rr}, \mathsf{formula})$
$(\mathsf{pos} :: \mathsf{rr}, \mathsf{formula})$
$(\mathsf{ub} :: \mathsf{rr}, \mathsf{sets(rr)}, \mathsf{formula})$
$(\mathsf{lub} :: \mathsf{rr}, \mathsf{sets(rr)}, \mathsf{formula})$

Notation

$(-a)$	means	$\mathsf{neg}(a)$
a^{-1}	means	$^{-1}(a)$
$a + b$	means	$+(a)(b)$
$a * b$	means	$*(a)(b)$
$a < b$	means	$<(a, b)$
$a \leq b$	means	$\leq (a, b)$
$a\ \mathsf{ub}\ s$	means	$\mathsf{ub}(a, s)$
$a\ \mathsf{lub}\ s$	means	$\mathsf{lub}(a, s)$

Axioms

$\forall a, b, c : \mathsf{rr} \,.\, a + (b + c) = (a + b) + c$
$\forall a, b : \mathsf{rr} \,.\, a + b = b + a$
$\forall a : \mathsf{rr} \,.\, a + 0 = a$
$\forall a : \mathsf{rr} \,.\, a + (-a) = 0$
$\forall a, b, c : \mathsf{rr} \,.\, a * (b * c) = (a * b) * c$
$\forall a : \mathsf{rr} \,.\, a * 1 = a \wedge 1 * a = a$
$\forall a : \mathsf{rr} \,.\, a \neq 0 \supset (a * a^{-1} = 1 \wedge a^{-1} * a = 1)$
$0^{-1}\uparrow$
$\forall a, b : \mathsf{rr} \,.\, a * b = b * a$
$0 \neq 1$
$\forall a, b, c : \mathsf{rr} \,.\, a * (b + c) = (a * b) + (a * c)$
$\forall a : \mathsf{rr} \,.\, (a = 0 \wedge \neg\mathsf{pos}(a) \wedge \neg\mathsf{pos}(-a)) \vee$
$\quad (a \neq 0 \wedge \mathsf{pos}(a) \wedge \neg\mathsf{pos}(-a)) \vee$
$\quad (a \neq 0 \wedge \neg\mathsf{pos}(a) \wedge \mathsf{pos}(-a))$
$\forall a, b : \mathsf{rr} \,.\, (\mathsf{pos}(a) \wedge \mathsf{pos}(b)) \supset \mathsf{pos}(a + b)$
$\forall a, b : \mathsf{rr} \,.\, (\mathsf{pos}(a) \wedge \mathsf{pos}(b)) \supset \mathsf{pos}(a * b)$
$\forall a, b : \mathsf{rr} \,.\, a < b \equiv \mathsf{pos}(b - a)$
$\forall a, b : \mathsf{rr} \,.\, a \leq b \equiv (a < b \vee a = b)$
$\forall x : \mathsf{rr}, s : \mathsf{sets(rr)} \,.\, x\ \mathsf{ub}\ s \equiv \forall a : \mathsf{rr} \,.\, a \in s \supset a \leq x$
$\forall x : \mathsf{rr}, s : \mathsf{sets(rr)} \,.\, x\ \mathsf{lub}\ s \equiv (x\ \mathsf{ub}\ s \wedge \forall y : \mathsf{rr} \,.\, y\ \mathsf{ub}\ s \supset x \leq y)$

$\forall s : \mathsf{sets(rr)} \,.\, ((s \neq \emptyset) \wedge (\exists x : \mathsf{rr} \,.\, x \ \mathsf{ub}\ s)) \supset \exists x : \mathsf{rr} \,.\, x \ \mathsf{lub}\ s$

4.2 Real Numbers A

The theory **RealNumA** is obtained by extending theory **RealNumBase** with definitions. By this extension, two new operators sub and div are added to **RealNumBase**. sub is the difference of two real numbers and div is the division operator.

RealNumA $\equiv$ **RealNumBase** extended with definitions by

Defined Operators

$(\mathsf{sub} :: \mathsf{rr} \rightarrow (\mathsf{rr} \rightarrow \mathsf{rr}))$

$(\mathsf{div} :: \mathsf{rr} \rightarrow (\mathsf{rr} \rightarrow \mathsf{rr}))$

Notation

$a - b$	means	$\mathsf{sub}(a)(b)$
a/b	means	$\mathsf{div}(a)(b)$

Defining Axioms

$\forall a, b : \mathsf{rr} \,.\, a - b = a + (-b)$

$\forall a, b : \mathsf{rr} \,.\, a/b \simeq a * b^{-1}$

4.3 Real Numbers B

Again the extension with definitions form is used to add two new operators, greater than ($>$) and greater than or equals to ($\geq$), to **RealNumA** to acquire **RealNumB**.

RealNumB $\equiv$ **RealNumA** extended with definitions by

Defined Operators

$(>:: \mathsf{rr}, \mathsf{rr}, \mathsf{formula})$

$(\geq:: \mathsf{rr}, \mathsf{rr}, \mathsf{formula})$

Notation

$a > b$	means	$> (a, b)$
$a \geq b$	means	$\geq (a, b)$

Defining Axioms

$\forall a, b : \mathsf{rr} \,.\, a > b \equiv \mathsf{pos}\ (a - b)$

$\forall a, b : \mathsf{rr} \,.\, a \geq b \equiv (a > b \vee a = b)$

4.4 Real Numbers C

The theory **RealNumC** is attained by extending **RealNumB** with definitions. It has two new operators, lb and glb, that determine the lower bound and greatest lower bound of a set, respectively.

RealNumC $\equiv$ **RealNumB** extended with definitions by

Defined Operators

$(\mathsf{lb} :: \mathsf{rr}, \mathsf{sets(rr)}, \mathsf{formula}))$

$(\mathsf{glb} :: \mathsf{rr}, \mathsf{sets(rr)}, \mathsf{formula}))$

Notation

$a\ \mathsf{lb}\ s$	means	$\mathsf{lb}(a, s)$
$a\ \mathsf{glb}\ s$	means	$\mathsf{glb}(a, s)$

Defining Axioms

$\forall x : \mathsf{rr}, s : \mathsf{sets(rr)} \,.\, x\ \mathsf{lb}\ s \equiv \forall a : \mathsf{rr} \,.\, a \in s \supset a \geq x$

$\forall x : \mathsf{rr}, s : \mathsf{sets(rr)} \,.\, x\ \mathsf{glb}\ s \equiv x\ \mathsf{lb}\ s \wedge \forall y : \mathsf{rr} \,.\, y\ \mathsf{lb}\ s \supset x \geq y$

4.5 Real Numbers D

The theory **RealNumD**, which is an extension of **RealNumC** with definitions, has an operator suc that gives the successor of a real number.

RealNumD $\equiv$ **RealNumC** extended with definitions by

Defined Operators

$(\mathsf{suc} :: \mathsf{rr} \rightarrow \mathsf{rr})$

Defining Axioms

$\mathsf{suc} = \lambda x : \mathsf{rr} \,.\, x + 1$

4.6 Real Numbers E

By appending a type of nn and an axiom defining it in theory of **RealNumD**, theory **RealNumE** is obtained. We are defining the natural numbers here as a special subcollection of real numbers.

RealNumE $\equiv$ **RealNumD** extended with definitions by

Defined Operators

$(\mathsf{nn} :: \mathsf{type})$

Defining Axioms

$\mathsf{nn} \ll \mathsf{rr}$

$\forall a : \mathsf{rr} \,.\, (a \downarrow \mathsf{nn}) \equiv (a = 0 \vee \exists b : \mathsf{nn} \,.\, \mathsf{suc}\ (b) = a)$

4.7 Real Numbers F

The theory **RealNumF** is obtained by adding a new operator zz, the type of integers, to the theory **RealNumE**

RealNumF $\equiv$ **RealNumE** extended with definitions by

Defined Operators

$(\mathsf{zz} :: \mathsf{type})$

Defining Axioms

$\mathsf{zz} \ll \mathsf{rr}$

$\forall a : \mathsf{rr} \,.\, (a \downarrow \mathsf{zz}) \equiv ((a \downarrow \mathsf{nn}) \vee ((-a) \downarrow \mathsf{nn}))$

4.8 Real Numbers G

The addition of a new operator qq that is the type of rational numbers produces the theory **RealNumG**.

RealNumG $\equiv$ **RealNumF** extended with definitions by

Defined Operators

(qq :: type)

Defining Axioms

qq $\ll$ rr

$\forall a : \mathsf{rr} \,.\, (a \downarrow \mathsf{qq}) \equiv (\exists p, q : \mathsf{zz} \,.\, q \neq 0 \wedge a = p/q)$

4.9 Real Numbers H

The theory **RealNumH** is obtained by adding factorial in **RealNumG**.

RealNumH $\equiv$ **RealNumG** extended with recursive definition by

Recursively Defined Operators

(factorial :: (nn $\rightarrow$ nn))

Functional

$\lambda f : (\mathsf{nn} \rightarrow \mathsf{nn}) \,.\, \lambda n : \mathsf{nn} \,.\, \mathsf{if}(n = 0, 1, f(n-1) * n)$

4.10 Metric Spaces Base

MetricSpacesBase is an axiomatization of theory of metric space. It comprises three operators ms, space and metric. ms is a type operator which is a type of metric spaces. space when applied to a member of ms yields a type representing the point of the metric space, it is a subtype of type of sets. metric measures the distance between two elements of the space of the metric space.

MetricSpacesBase $\equiv$ **RealNum** extended with a profile by

Profiled Operators

(ms :: type)

(space :: ms, type)

(metric :: Λm : ms . space(m) $\rightarrow$ (space(m) $\rightarrow$ rr))

Axioms

$\forall m$: ms . space(m) $\ll$ V

$\forall m$: ms, p_1, p_2 : space(m) .

metric(m)(p_1)(p_2) $\geq 0 \wedge$ metric(m)(p_1)(p_2) $= 0 \equiv p_1 = p_2$

$\forall m$: ms, p_1, p_2 : space(m) .

metric(m)(p_1)(p_2) = metric(m)(p_2)(p_1)

$\forall m$: ms, p_1, p_2, p_3 : space(m) .

metric(m)(p_1)(p_2) $\leq$ metric(m)(p_1)(p_3) + metric(m)(p_3)(p_2)

4.11 Metric Spaces

The theory **MetricSpaces** is an extension of the theory **MetricSpacesBase** with definitions by two new operators openball and closedball.

MetricSpaces $\equiv$ **MetricSpacesBase** extended with definitions by

Defined Operators

(openball :: Λm : ms . (space(m) $\rightarrow$ (rr $\rightarrow$ sets(space(m)))))

(closedball :: Λm : ms . (space(m) $\rightarrow$ (rr $\rightarrow$ sets(space(m)))))

Defining Axioms

openball : $\forall m$: ms, c : space(m), r : rr, $r \geq 0 \supset$

$\forall p$: space(m) . $p \in$ openball(m)(c)(r) $\equiv$ metric(m)(c)(p) $< r$

closedball : $\forall m$: ms, c : space(m), r : rr, $r \geq 0 \supset$

$\forall p$: space(m) . $p \in$ closedball(m)(c)(r) $\equiv$ metric(m)(c)(p) $\leq r$

4.12 Monoids

The language of the theory **Monoid** comprises four operators md, space, the identity element e and the binary operator $\bullet$.

Monoid

Defined Operators

(md :: type)

(space :: md, type)

(e :: Λm : md . space(m))

($\bullet$ :: Λm : md . (space(m) $\rightarrow$ (space(m) $\rightarrow$ space(m))))

Notation

e_m	means	$e(m)$
$p_1 \bullet_m p_2$	means	$\bullet\,(m, p_1, p_2)$

Defining Axioms

$\forall m$: md . space(m) $\ll$ V

$\forall m$: md, p_1, p_2, p_3 : space(m) .

$p_1 \bullet_m (p_2 \bullet_m p_3) = (p_1 \bullet_m p_2) \bullet_m p_3$

$\forall m$: md, p : space(m) .

$p \bullet_m \mathsf{e}_m = p \wedge \mathsf{e}_m \bullet p_m = p$

4.13 Monoidal Metric

The theory **MonoidalMetric** is the union of the theory **MetricSpacesBase** and **Monoid**. The renaming operation is used to rename the type md to the type ms.

MonoidalMetric $\equiv$ **MetricSpacesBase** $\oplus$ **Monoid** with md $\mapsto$ ms

CHAPTER 5

THEORY BUILDING TOOLS

In Mei, a parametrized theory is a functor. Mei not only supports first-order functors but it also supports higher-order functors where a higher-order functor is a mapping that takes a functor as an argument or returns a functor. A functor can be instantiated only when the actual parameter theory or functor matches the argument type. In the previous chapter we used the theory building operations extension, union and renaming to construct new theories from existing theories. In this chapter, we will employ another theory building operator "functor" to build the theory of calculus of one variable.

5.1 Binary Iterative Functor

BinaryIterative is a first-order functor that takes a theory of type (**Monoidal Metric**) as its argument, introduces a concept of applying the binary operator $\bullet$ iteratively on given function over all integers in the specified range, and returns a theory of type returntype(**BinaryIterative**).

Binary Iterative $\equiv$ functor X : type(**MonoidalMetric**) . X extended with recursive definitions by

Recursively Defined Operators

(binaryiterative :: Λm : ms . (zz $\rightarrow$ space(m)) $\rightarrow$ (zz $\rightarrow$ (zz $\rightarrow$ space(m))))

Functional

λf : Λm : ms . (zz $\rightarrow$ space(m)) $\rightarrow$ (zz $\rightarrow$ (zz $\rightarrow$ space(m))) .
λm : ms, λg : zz $\rightarrow$ space(m) . λj : zz . λk : zz .
if($k < j$, e_m, $g(j) \bullet_m f(m, g, j+1, k)$)

5.2 Limit Functor

Like **BinaryIterative**, **Limit** is a first-order functor whose formal parameter is a theory of type(**MonoidalMetric**). It adds a notion of limit of a function at a point in **MonoidalMetric** and returns a theory of type returntype (**Limit**).

Limit $\equiv$ functor X : type(**MonoidalMetric**) . X extended with definitions by

Defined Operators

(lim :: $\Lambda m_1, m_2$: ms . (space(m_1) $\rightarrow$ space(m_2)) $\rightarrow$ space(m_1) $\rightarrow$ space(m_2))

Defining Axioms

lim $= \lambda m_1$, m_2 : ms . λf : space(m_1) $\rightarrow$ space(m_2) .
a : space(m_1) . ιL : space(m_2) .
$\forall \epsilon$: rr, $\epsilon > 0$ $\supset$
$\exists \delta$: rr, $\delta > 0 \wedge \forall p$: space(m_1) . $p \neq a \wedge$ metric(m_1)(p)(a) $< \delta \supset$
metric(m_2)($f(p)$)(L) $< \epsilon$

Theorems

$\forall m_1, m_2$: ms . a : space(m_1), k : space(m_2) .
lim(m_1)(m_2)(λx : space(m_1) . k)(a) $\simeq k$
$\forall m_1, m_2$: ms . λf : space(m_1) $\rightarrow$ space(m_2) . a : space(m_1), c : space(m_2) .
lim(m_1)(m_2)(λx : space(m_1) . $c * f(x)$)(a) $\simeq$
$c *$ lim(m_1)(m_2)(f)(a)

5.3 Limit of Sequence Functor

LimitSeq takes a theory X of type(**MonoidalMetric**), defines the concept of limit of sequence in X and returns a theory of type returntype(**LimSeq**) .

LimitSeq $\equiv$ functor X : type(MonoidalMetric) . X extended with definitions by

Defined Operators

(limseq :: Λm : ms . (nn $\rightarrow$ space(m)) $\rightarrow$ space(m))

Defining Axioms

limseq $= \lambda m$: ms . λf : nn $\rightarrow$ space(m) .
ιL : space(m) . $\forall \epsilon$: rr, $\epsilon > 0$ $\supset$
$\exists N$: nn . $\forall n$: nn . $n \geq N \supset$
metric(m)($f(n)$)(L) $< \epsilon$

5.4 Continuity Functor

Continuous is instantiated by a theory of type returntype(**Limit**) and produces a theory that defines the continuity of a function at a point. It determines whether the function is continuous at the given point or not.[1]

Continuous $\equiv$ functor X : returntype(**Limit**) . X extended with definitions by

Defined Operators

(cont :: ms, ms, V, V, formula)

Defining Axioms

[1] In this case, the operator is not written as a curried function because in Chiron a function cannot return a formula.

$$\forall m_1, m_2 : \mathsf{ms},\ f, a : \mathsf{V} \ .\ \mathsf{cont}(m_1)(m_2)(f)(a) \equiv$$
$$f \downarrow (\mathsf{space}(m_1) \to \mathsf{space}(m_2)) \wedge$$
$$a \downarrow \mathsf{space}(m_1) \wedge$$
$$\mathsf{lim}(m_1)(m_2)(f)(a) = f(a)$$

5.5 Derivative Functor

The **Derivative** functor adds the concept of derivative of a function at a point to the parameter theory.

Derivative $\equiv$ functor X : returntype(**Limit**) . X extended with definitions by

Defined Operators

$(\mathsf{der} :: \Lambda m : \mathsf{ms} \ .\ (\mathsf{space}(m) \to \mathsf{rr}) \to \mathsf{space}(m) \to \mathsf{rr})$

Defining Axioms

$\forall m : \mathsf{ms} \ .\ \lambda f : \mathsf{space}(m) \to \mathsf{rr},\ a : \mathsf{space}(m) \ .$
$\mathsf{der}(m)(f)(a) \equiv \mathsf{lim}(m)(\lambda x : \mathsf{space}(m) \ .\ \frac{f(a+x)-f(a)}{\mathsf{metric}(m)(x)(0)})(0) \downarrow$

Theorems

$\forall m : \mathsf{ms} \ .\ \forall f : \mathsf{space}(m) \to \mathsf{rr} \ .\ a, c : \mathsf{space}(m) \ .$
$\mathsf{der}(m)(\lambda x : \mathsf{space}(m) \ .\ c * f(x))(a) \simeq c * \mathsf{der}(m)(f)(a)$
$\forall m : \mathsf{ms} \ .\ \lambda c : \mathsf{space}(m) \ .\ \mathsf{der}(m)(\lambda x : \mathsf{space}(m) \ .\ c)(a) = 0$

5.6 Infinite Series Functor

The formal parameter of the functor **InfSum** is the union of returntype(**LimSeq**) and returntype(**BinaryIterative**). This functor gives the sum of an infinite series.

InfSum $\equiv$ functor X : returntype(**LimSeq**).
functor Y : returntype(**BinaryIterative**) .
(X $\oplus$ Y) extended with definitions by

Defined Operators

(infsum :: Λm : ms . (zz $\rightarrow$ space(m)) $\rightarrow$ zz $\rightarrow$ space(m))

Defining Axioms

infsum $=$ λm : ms, f : zz $\rightarrow$ space(m), j : zz . limseq(m)(λk : zz . BinaryIterative(m)(f)(j)(k))

CHAPTER 6

SAMPLE LIBRARY

In chapter 4, we used the theory building operators extension, union and renaming to add various notions to the theory **RealNumBase**. In this chapter, we will illustrate how to add new machinery in a theory using functors and how the system proposed in the previous chapter works on real numbers. We will start off by extending the theory **RealNumH** (given in section 4.9) and then apply all the functors that have been shown in the previous chapter on the resulting theory to finally obtain the theory of **RealNum**.

6.1 Real Numbers I

The theory **RealNumI** is obtained by extending theory **RealNumH** with the profiled operator rms. rms contains exactly one element, which is intended to represent the real numbers as metric space.

RealNumI $\equiv$ **RealNumH** extended with a profile by

Profiled Operators

rms :: type

Axioms

$\forall x, y$: rms . $x = y$

6.2 Real Numbers J

The extension of **RealNumI** by two new operators rspace and rmetric gives **RealNumJ**.

RealNumJ $\equiv$ **RealNumI** extended with definitions by

Defined Operators

rspace :: rms, type

rmetric :: Λm : rms . rspace(m) $\rightarrow$ rspace(m) $\rightarrow$ rr

Defining Axioms

$\forall m$: rms . rspace(m) $=_{\text{ty}}$ rr

$\forall m$: rms . x, y : rspace(m) . rmetric(m)(x)(y) $= |x - y|$

6.3 Real Numbers J′

The extension of **RealNumJ** with theorems yields **RealNumJ′**.

RealNumJ′ $\equiv$ **RealNumJ** extended with theorems by

Theorems

$\forall m$: rms . x : rspace(m) . $|x| \geq 0 \wedge |x - y| = 0 \equiv x = y$

$\forall m$: rms . x, y : rspace(m) . $|x - y| = |y - x|$

$\forall m$: rms . x, y, z : rspace(m) . $|x - y| \leq |x - z| + |z - y|$

6.4 Real Numbers K

We will show that **RealNumJ′** has a structure of **MonoidalMetric** by interpreting theory of **MonoidalMetric** to **RealNumJ′** employing a view (for details of view see section 2.4). In other words, **RealNumJ′** is not of type(**MonoidalMetric**) but using fitting morphism we can coerce it to type(**MonoidalMetric**).

The interpretation from **MonoidalMetric** to **RealNumJ′** is the view

view(type(**MonoidalMetric**), type(**RealNumJ′**), ms $\mapsto$ rms, space $\mapsto$ rspace,
metric $\mapsto$ rmetric, $\bullet \mapsto +$, e $\mapsto$ 0)

The extension of theory **RealNumJ′** by an interpreted theory is a theory **RealNumK**

RealNumK $\equiv$ **RealNumJ′** extended with an interpreted theory by
MonoidalMetric $\mapsto$ **RealNumJ′**

View

ms $\mapsto$ rms
space $\mapsto$ rspace
metric $\mapsto$ rmetric
$\bullet \mapsto +$
e $\mapsto$ 0

6.5 Real Numbers with Limit

Until now, we have only used the method of extension to get new theories. In the rest of the sections of this chapter we will use functors to produce new theories. All these functors are first order and have been defined in Chapter 5. The first functor we will use is **Limit**. As defined in the previous chapter, it is instantiated by theory of type(**MonoidalMetric**), therefore we will first employ a view to have a mapping from the language of **MonoidalMetric** to **RealNumK** and then apply the functor **Limit** on **RealNumK**.

Let W= (type(**MonoidalMetric**), type(**RealNumK**),
ms $\mapsto$ rms, space $\mapsto$ rspace, metric $\mapsto$ rmetric, $\bullet \mapsto +$, e $\mapsto$ 0).

RealNumL $\equiv$ **Limit** (**RealNumK** `with view` W) extended with definitions by

Defined Operators

(rlim :: $\Lambda m_1, m_2$: rms . (space(m_1) $\rightarrow$ space(m_2)) $\rightarrow$ space(m_1) $\rightarrow$ space(m_2))

Defining Axioms

$\mathsf{rlim} \equiv \mathsf{lim}$

Theorems

$\forall m_1, m_2 : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2),\ a : \mathsf{space}(m_1) \,.$
$\mathsf{lim}(m_1)(m_2)(f+g)(a) \simeq$
$\mathsf{lim}(m_1)(m_2)(f)(a) + \mathsf{lim}(m_1)(m_2)(g)(a)$

$\forall m_1, m_2 : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2),\ a : \mathsf{space}(m_1) \,.$
$\mathsf{lim}(m_1)(m_2)(f-g)(a) \simeq$
$\mathsf{lim}(m_1)(m_2)(f)(a) - \mathsf{lim}(m_1)(m_2)(g)(a)$

$\forall m_1, m_2 : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2),\ a : \mathsf{space}(m_1) \,.$
$\mathsf{lim}(m_1)(m_2)(f * g)(a) \simeq$
$\mathsf{lim}(m_1)(m_2)(f)(a) * \mathsf{lim}(m_1)(m_2)(g)(a)$

$\forall m_1, m_2 : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2),\ a : \mathsf{space}(m_1) \wedge$
$\mathsf{lim}(m_1)(m_2)(g)(a) \neq 0 \,.$
$\mathsf{lim}(m_1)(m_2)(f/g)(a) \simeq$
$\mathsf{lim}(m_1)(m_2)(f)(a) / \mathsf{lim}(m_1)(m_2)(g)(a)$

Here $f+g$, $f-g$, etc. are defined in the usual way. The application of **Limit** to the theory **RealNumK** returns a theory **RealNumL** which has in its language a new operator rlim.

6.6 Real Numbers with Limit of Sequence

The functor **LimSeq** is instantiated by a theory of type type(**MonoidalMetric**). As **RealNumL**, which we obtain by applying the functor **Limit** to **RealNumK**, has a structure of **MonoidalMetric** therefore it can instantiate the functor **LimSeq**.

RealNumM $\equiv$ **LimSeq** (**RealNumL**) extended with definitions by

Defined Operators

$(\mathsf{rlimseq} :: \Lambda m : \mathsf{rms} \,.\, (\mathsf{nn} \to \mathsf{space}(m)) \to \mathsf{space}(m))$

Defining Axioms

rlimseq $\equiv$ limseq

This gives back a theory **RealNumM** that has a notion of rlimseq.

6.7 Real Numbers with Continuity

Continuous is a functor whose formal parameter is a theory of type returntype(**Limit**). It can be applied to the theory **RealNumM**, that possesses the concept of limit, after a mapping from returntype(**Limit**) to type(**RealNumM**).

Let W= (returntype(**Limit**), type(**RealNumM**), lim $\mapsto$ rlim).

RealNumN $\equiv$ **Continuous**(**RealNumM** `with view` W) extended with definitions by

Defined Operators

(rcont :: rms, rms, V, V, formula)

Defining Axioms

rcont $\equiv$ cont

Theorems

$\forall m_1, m_2 : \mathsf{rms} \;.\; \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2) \;.\; a : \mathsf{space}(m_1) \;.$
$\mathsf{rcont}(m_1)(m_2)(f)(a) \wedge \mathsf{rcont}(m_1)(m_2)(g)(a) \supset$
$\mathsf{rcont}(m_1)(m_2)(f+g)(a)$

$\forall m_1, m_2 : \mathsf{rms} \;.\; \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2), \;.\; a : \mathsf{space}(m_1) \;.$
$\mathsf{rcont}(m_1)(m_2)(f)(a) \wedge \mathsf{rcont}(m_1)(m_2)(g)(a) \supset$
$\mathsf{rcont}(m_1)(m_2)(f * g)(a)$

$\forall m_1, m_2 : \mathsf{rms} \;.\; \lambda f, g : \mathsf{space}(m_1) \to \mathsf{space}(m_2) \;.\; a : \mathsf{space}(m_1) \;.$
$\mathsf{rcont}(m_1)(m_2)(f)(a) \wedge \mathsf{rcont}(m_1)(m_2)(g)(a) \wedge g(a) \neq 0 \supset$
$\mathsf{rcont}(m_1)(m_2)(f/g)(a)$

This adds a new operator rcont and its corresponding axiom in the theory **RealNumM**

6.8 Real Numbers with Derivative

The theory **RealNumN** can be taken as the actual parameter of the functor **Derivative** which takes theory of type returntype(**Limit**) as its argument.

RealNumO $\equiv$ **Derivative** (**RealNumN**) extended with definitions by

Defined Operators

$(\mathsf{rder} :: \Lambda m : \mathsf{rms} \,.\, (\mathsf{space}(m) \to \mathsf{rr}) \to \mathsf{space}(m) \to \mathsf{rr})$

Defining Axioms

$\mathsf{rder} \equiv \mathsf{der}$

Theorems

$\forall m : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m) \to \mathsf{rr} \,.\, a : \mathsf{space}(m) \,.$
$\mathsf{rder}(m)(f + g)(a) \simeq$
$\mathsf{rder}(m)(f)(a) + \mathsf{rder}(m)(g)(a)$

$\forall m : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m) \to \mathsf{rr} \,.\, a : \mathsf{space}(m) \,.$
$\mathsf{rder}(m)(f - g)(a) \simeq$
$\mathsf{rder}(m)(f)(a) - \mathsf{rder}(m)(g)(a)$

$\forall m : \mathsf{rms} \,.\, \lambda f, g : \mathsf{space}(m) \to \mathsf{rr} \,.\, a : \mathsf{space}(m) \,.$
$\mathsf{rder}(m)(f * g)(a) \simeq$
$\mathsf{rder}(m)(f)(a) * g(a) + f(a) * \mathsf{rder}(m)(g)(a)$

6.9 Real Numbers with Binary Iterative

The application of **BinaryIterative**, which takes a theory of type type(**MonoidalMetric**) as its argument, on **RealNumO** produces the theory of **RealNumP**.

RealNumP $\equiv$ **BinaryIterative** (**RealNumO**) extended with definitions by

Defined Operators

(binaryiterative :: Λm : rms . (zz $\rightarrow$ space(m)) $\rightarrow$ (zz $\rightarrow$ (zz $\rightarrow$ space(m))))

Defining Axioms

rbinaryiterative $\equiv$ binaryiterative

6.10 Real Numbers with Infinite Series

The formal parameter of the functor **InfiniteSum** is returntype(**LimSeq**) $\oplus$ returntype(**BinaryIterative**). By employing a view, **RealNumP** can be taken as its argument.

Let W= (returntype(**LimSeq**) $\oplus$ returntype(**BinaryIterative**), type(**RealNumP**), rLimSeq $\mapsto$ LimSeq, rBinaryIterative $\mapsto$ BinaryIterative).

RealNum $\equiv$ **InfiniteSum** (**RealNumP** `with view` W) extended with definitions by

Defined Operators

(rinfsum :: Λm : rms . (nn $\rightarrow$ space(m) $\rightarrow$ nn $\rightarrow$ nn $\rightarrow$ nn $\rightarrow$ space(m))

Defining Axioms

rinfsum $\equiv$ infsum

6.11 Real Numbers without using Functors

Now we demonstrate how we can use extensions with definitions and recursion to get the theory of **RealNum** from **RealNumK** without using functors.

RealNumL' $\equiv$ **RealNumL** extended with recursive definitions by

Recursively Defined Operators

(binaryiterative$'$ :: (zz $\rightarrow$ rr) $\rightarrow$ (zz $\rightarrow$ (zz $\rightarrow$ rr)))

Functional

$$\lambda f : (\mathsf{zz} \to \mathsf{rr}) \to (\mathsf{zz} \to (\mathsf{zz} \to \mathsf{rr})) \,.$$
$$\lambda g : \mathsf{zz} \to \mathsf{rr} \,.\, \lambda j : \mathsf{zz} \,.\, \lambda k : \mathsf{zz} \,.$$
$$\mathsf{if}(k < j,\, e_m,\, g(j) \bullet_m f(g, j+1, k))$$

The theory **RealNumL′** when extended with the following defined operators gives the theory of **RealNum**.

RealNum $\equiv$ **RealNumL′** extended with definitions by

Defined Operators

$\mathsf{rmetric}' :: \mathsf{rr} \to \mathsf{rr} \to \mathsf{rr}$

$\mathsf{rlim}' :: (\mathsf{rr} \to \mathsf{rr}) \to \mathsf{rr} \to \mathsf{rr}$

$\mathsf{rlimseq}' :: (\mathsf{nn} \to \mathsf{rr}) \to \mathsf{rr}$

$\mathsf{rder}' :: (\mathsf{rr} \to \mathsf{rr}) \to \mathsf{rr} \to \mathsf{rr}$

$\mathsf{rcont}' :: \mathsf{rr} \to \mathsf{rr}, \mathsf{rr}, \mathsf{formula}$

$\mathsf{rinfsum}' :: (\mathsf{zz} \to \mathsf{rr}) \to (\mathsf{zz} \to \mathsf{rr})$

Defining Axioms

$\forall x, y : \mathsf{rr} \,.\, \mathsf{rmetric}'(x)(y) = |x - y|.$

$\mathsf{rlim}' = \lambda f : \mathsf{rr} \to \mathsf{rr} \,.$
$\quad a : \mathsf{rr} \,.\, \iota L : \mathsf{rr} \,.$
$\quad\quad \forall \epsilon : \mathsf{rr},\, \epsilon > 0 \supset$
$\quad\quad\quad \exists \delta : \mathsf{rr},\, \delta > 0 \wedge \forall p : \mathsf{rr} \,.\, p \neq a \wedge \mathsf{rmetric}'(p)(a) < \delta \supset$
$\quad\quad\quad\quad \mathsf{rmetric}'(f(p))(L) < \epsilon$

$\mathsf{rlimseq}' = \lambda f : \mathsf{nn} \to \mathsf{rr} \,.$
$\quad \iota L : \mathsf{rr} \,.\, \forall \epsilon : \mathsf{rr} \,.\, \epsilon > 0 \supset$
$\quad\quad \exists N : \mathsf{nn} \,.\, \forall n : \mathsf{nn} \,.\, n \geq N \supset$
$\quad\quad\quad \mathsf{rmetric}'(f(n))(L) < \epsilon$

$\forall f : \mathsf{rr} \to \mathsf{rr},\, a : \mathsf{rr} \,.\, \mathsf{rcont}'(f)(a) \equiv \mathsf{rlim}'(f)(a) = f(a)$

$\forall f : \mathsf{rr} \to \mathsf{rr},\ a : \mathsf{rr}\ .\ \mathsf{rder}'(f)(a) \equiv \mathsf{rlim}'(\lambda x : \mathsf{rr}\ .\ \frac{f(a+x)-f(a)}{x})(0)\downarrow$

$\mathsf{rinfsum}' = \forall f : \mathsf{nn} \to \mathsf{rr},\ j : \mathsf{nn}\ .\ \mathsf{rlimseq}'(\lambda k : \mathsf{nn}\ .\ \mathsf{rbinaryiterative}'(f)(j)(k))$

This shows that we can get at **RealNum** from **RealNumI** by two different ways, either by extensions or by using functors. When we use functors, we can apply the functor that we need and we do not have to unnecessarily employ other functors that we do not need, for instance, we have a theory of returntype(**Limit**), then we can apply functor **Derivative** on it to have the notion of derivative in our theory. On the other hand, when we use extensions, we have concept of derivative along with other operators which we may not need. This is the benefit of having functors.

6.12 Real Numbers with Definite Integral

RealNum when extended with rint gives **RealNum'**.

RealNum' $\equiv$ **RealNum** extended with definitions by

Defined Operators

$\mathsf{rint} :: (\mathsf{rr} \to \mathsf{rr}) \to (\mathsf{rr} \to (\mathsf{rr} \to \mathsf{rr}))$

Defining Axioms

$\lambda f : \mathsf{rr} \to \mathsf{rr},\ a, b : \mathsf{rr}\ .\ \mathsf{rint}(f)(a)(b) =$
$\quad \mathsf{rlimseq}'(\lambda n : \mathsf{nn}\ .\ \mathsf{rinfsum}'(\lambda i : \mathsf{nn}\ .\ f(a+(i*(b-a))/n)*((b-a)/n))(0)(n))$

Theorems

$\lambda f : \mathsf{rr} \to \mathsf{rr}\ .\ a, b : \mathsf{rr}\ .$
$\quad \mathsf{rint}(f)(a)(b) \simeq -\mathsf{rint}(f)(b)(a)$
$\lambda f : \mathsf{rr} \to \mathsf{rr}\ .\ a, b, c : \mathsf{rr}\ .$
$\quad \mathsf{rint}(f)(a)(c) + \mathsf{rinf}(f)(c)(b) \simeq \mathsf{Inf}(f)(a)(b)$

Figure 6.1 illustrates the way we have built the abstract theory of calculus of real numbers on top of metric space. In Chapter4, the theory of **RealNumBase** and **Monoid** was built from scratch and then a couple of extensions were made to add more machinery in **RealNumBase**. **MetricSpacesBase** was also developed by

extending real numbers. In Chapter 6, we showed that real numbers has structure of **MonoidalMetric** by using a fitting morphism.

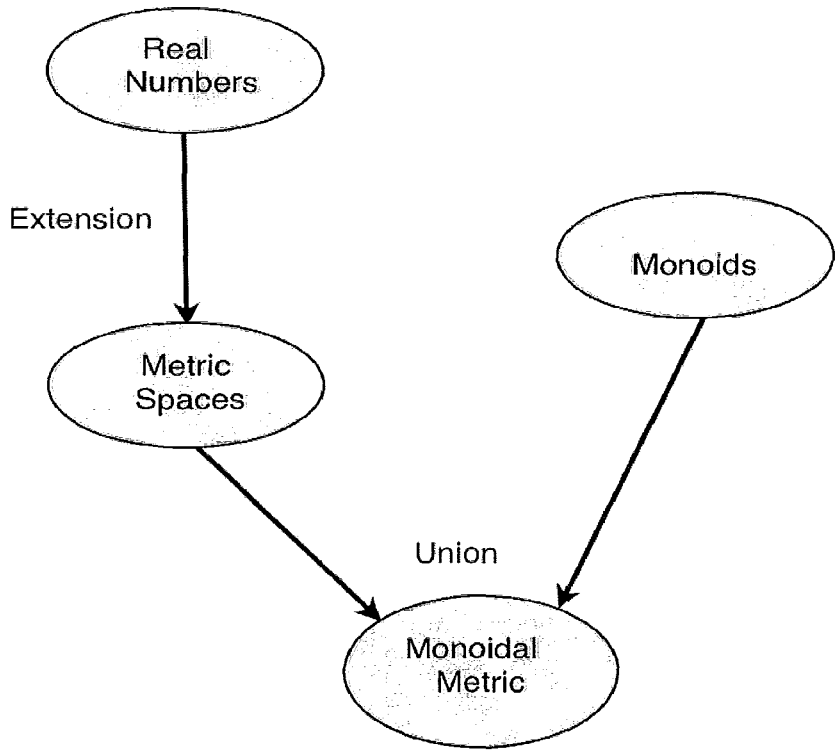

Figure 6.1: This shows how the theories in library are related to each other.

CHAPTER 7

COMPARISON TO PREVIOUS WORK

There are many ways to build a library of mathematics or to organize mathematical knowledge. One of the most common and old methods of organizing mathematics is by dividing mathematics into different categories. In all MMSs, theory libraries are built using different systems for organizing theories. There are some MMSs that employ module systems, for example, in theorem provers IMPS, PVS [25], Isabelle [26], Coq [8] , Automath [9] and in computer algebra systems Maple [7], Mathematica [1], Axiom [18], Aldor [30], Focal [29] and Magma [4], modules are used to organize theories. The module systems are also used in programming and specification languages for organizing large software developments and specifications, for instance, in ML-family module systems, algebraic specification languages Maude, Specware, CASL, etc.

There are several module systems that are designed for building mathematical libraries, for example, Focal [29], Coq [8] and Aldor [30]. The theory library proposed in this thesis is built by using Mei. Mei is similar to Module System for Mathematical Theories [27]. Both systems are independent of their underlying logic. For comparison of Mei with the module systems used in the above mentioned MMSs, see [32].

The library proposed in this thesis is based on calculus. There are many other MMSs in which real analysis is formalized, for instance, IMPS, HOL [22], PVS, ACL2 [19], Isabelle, Coq, etc.

The library shown in this work employs biform theories. Although the theories presented in this thesis do not use transformers, they could. So far only one other library is built using biform theories and that is the library of theory types by Huan Zhang [33]. Huan's library is based on module systems of typed programming languages and algebraic specification languages and it is independent of underlying logic. Both the work done, in this thesis and by Huan Zhang were carried out at the same time. In this thesis, the module system Mei has been employed for organizing mathematical knowledge but other module systems, like module system for mathematical theories (MMT), are also appropriate for biform theories.

The portion of theory library demonstrated in this thesis is different from other libraries because it is a well organized library which has capability of both deduction and computation.

CHAPTER 8

CONCLUSION

A theory library in a mechanized mathematics system is a place where the mathematical knowledge is stored. Until now, the library of an MMS contains either axiomatic theories or algorithmic theories and has either symbolic computation capabilities of computer algebra systems or the formal deduction capabilities of theorem proving systems but not both because of which we have two different types of MMSs, theorem provers and computer algebra systems.

In this thesis, some extensions has been made to the module system Mei. Six special kinds of theory extensions has been added to Mei. A theory now can be extended with theorems, with definitions, with a profile, with an inductive data type, with a recursive definition or with an interpreted theory.

Furthermore, a small portion of a theory library for an MMS has been presented and the key idea is to illustrate how the module system Mei, the logic Chiron and biform theories, as a way of representing theories, could be used to develop a theory library. In this sample library, machinery for calculus of one variable has been developed.

The most significant feature of this library is that it is a novel library that expresses theories both axiomatically and algorithmically thus allowing the power of both computation and deduction to applied.

CHAPTER 9

FUTURE WORK

In future, the following work can be done:

- The portion of the theory library that has been presented in this thesis can be used as a model to build a calculus portion in another theory library.
- Biform theories have been used in this thesis but none of the theories contain transformers. Some transformers can be added to these theories, for example, to do symbolic differentiation, symbolic integration, etc.
- The calculus machinery presented in this work is univariate and can be expanded to multivariate.
- The theory **RealNumBase** has been developed from scratch, it could be build up from simpler theories, that is to say, from simple algebraic theories such as a theory of a monoid, etc.
- This work can be translated to Open Mathematical Documents (OMDoc) to make it widely accessible to other theory library builders.
- The concept given in this thesis of employing Mei, Chiron and biform theories to develop a theory library that has computation and deduction power is quite simple and can help to create a bigger theory library that contains theories from all branches of Mathematics.

APPENDIX A – LIST OF SYMBOLS

In this chapter, we provide an overview of Chiron's symbols that are used in this thesis.

Symbols in Chiron	
Operator	$(s :: k_1, \ldots, k_{n+1})$
Operator application	$(s :: k_1, ..., k_{n+1})(e_1, ..., e_n)$
Variable	$(x : \alpha)$
Type application	$\alpha(a)$
Dependednt function type	$\Lambda x : \alpha \,.\, \beta)$
Simple function type	$\alpha \rightarrow \beta$
Function application	$f(a)$
Function abstraction	$(\lambda x : \alpha \,.\, b)$
Conditional term	$\mathsf{if}(A, b, c)$
Existential quantification	$(\exists x : \alpha \,.\, B)$
Unique existential	$(\exists! x : \alpha \,.\, B)$
Universal quantification	$(\forall x : \alpha \,.\, B)$
Definite description	$(\iota x : \alpha . B)$
indefinite description	$(\epsilon x : \alpha . B)$
Quotation	$\lceil e \rceil$
Evaluation	$[\![a]\!]_k$
Set type	V
Type equality	$\alpha =_{\mathsf{ty}} \beta$
Term equality	$a = b$
Quasi-equality	$a \simeq b$

Continued on next page

Continued from previous page	
Formula equality	$A \equiv B$
Truth	T
Falsehood	F
Negation	$\neg A$
Disjunction	$A \vee B$
Conjunction	$A \wedge B$
Implication	$A \supset B$
Definedness in a type	$a \downarrow \alpha$
Definedness	$a \downarrow$
Type order	$\alpha \ll \beta$
Canonical undefined term	$\perp_c$
Conditional type	$\mathsf{if}(A, \beta, \gamma)$
Conditional formula	$\mathsf{if}(A, B, C)$
Class membership	$a \in b$
Subclass	$a \subset b$

BIBLIOGRAPHY

[1] B. Buchberger, "*Mathematica:* a system for doing mathematics by computer?." Preprint, 1993.

[2] "CADE : the Conference on Automated Deduction." Web site at `http://www.cadeconference.org//`.

[3] "Calculemus Project: Systems for Integrated Computation and Deduction." Web site at `http://www.calculemus.net/`.

[4] J. Cannon and C. Playoust, *Algebraic programming with Magma I : An introduction to the Magma language.* Springer-Verlag NewYork, 2006.

[5] J. Carette and W. M. Farmer, "High-level theories," in *Intelligent Computer Mathematics* (J. C. S. Autexier, V. S. J. Rubio, M. Suzuki, and F. Wiedijk, eds.), vol. 5144 of *Lecture Notes in Computer Science (LNCS)*, pp. 232–245, 2008.

[6] J. Carette and W. M. Farmer, "A review of mathematical knowledge management," in *Intelligent Computer Mathematics* (J. Carette, L. Dixon, C. S. Coen, and S. Watt, eds.), vol. 5625 of *Lecture Notes in Computer Science (LNCS)*, pp. 233–246, 2009.

[7] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt, *Maple V Language Reference Manual.* Springer-Verlag, 1991.

[8] Coq Development Team, *The Coq Proof Assistant Reference Manual, Version 8.2*, 2008. Available at `http://coq.inria.fr/doc/`.

[9] N. G. de Bruijn, "A survey of the project AUTOMATH," in *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism* (J. P. Seldin and J. R. Hindley, eds.), pp. 579–606, Academic Press, 1980.

[10] W. M. Farmer, "The interactive mathematics laboratory," in *Proceedings of the 31st Annual Small College Computing Symposium (SCCS '98)*, pp. 84–94, April 1998.

[11] W. M. Farmer, "MKM: A new interdisciplinary field of research," *ACM SIGSAM Bulletin*, vol. 38, pp. 47–52, 2004.

[12] W. M. Farmer, "Mathematical knowledge management," in *Encyclopedia of Knowledge Management* (D. G. Schwartz, ed.), pp. 599–604, Information Science Reference, 2005.

[13] W. M. Farmer, "Biform theories in chiron," in *Towards Mechanized Mathematical Assistants* (M. Kauers, M. Kerber, R. R. Miner, and W. Windsteiger, eds.), vol. 4573 of *Lecture Notes in Computer Science (LNCS)*, pp. 66–79, 2007.

[14] W. M. Farmer, "Chiron: A multi-paradigm logic," in *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec* (R. Matuszewski and A. Zalewska, eds.), vol. 10(23) of *Studies in Logic, Grammar and Rhetoric*, pp. 1–19, University of Białystok, 2007.

[15] W. M. Farmer, "Chiron: A set theory with types, undefinedness, quotation, and evaluation," SQRL Report No. 38, McMaster University, 2007. Revised 2008.

[16] W. M. Farmer, J. D. Guttman, and F. J. Thayer, "Little theories," in *Automated Deduction—CADE-11* (D. Kapur, ed.), vol. 607 of *Lecture Notes in Computer Science*, pp. 567–581, Springer-Verlag, 1992.

[17] W. M. Farmer, J. D. Guttman, and F. J. Thayer, "The IMPS user's manual," Tech. Rep. M-93B138, The MITRE Corporation, 1993. Available at `http://imps.mcmaster.ca/`.

[18] R. D. Jenks and R. S. Sutor, *Axiom : The Scientific Computation System.* Springer-Verlag, 1992.

[19] M. Kaufmann, P. Manolios, and J. S. Moore, *Computer-Aided Reasoning: An Approach.* Kluwer Academic Publishers, 2000.

[20] "Logic Daemon." Web site at `http://logic.tamu.edu/`.

[21] "MathScheme : An Integrated Framework For Computer Algebra And Computer Theorem Proving." Web site at `http://imps.mcmaster.ca/mathscheme/`.

[22] M.J.C.Gordon and T.F.Melham, *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic.* Cambridge University Press, 1993.

[23] "MKM : Mathematical Knowledge Management." Web site at `http://www.mkm-ig.org/`.

[24] "Modules and Libraries for Proof Assistants." Web site at `http://www.itu.dk/research/pls/wiki/index.php/MLPA-09`.

[25] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas, "PVS: Combining specification, proof checking, and model checking," in *Computer Aided Verification: 8th International Conference, CAV '96* (R. Alur and T. A. Henzinger, eds.), vol. 1102 of *Lecture Notes in Computer Science*, pp. 411–414, Springer-Verlag, 1996.

[26] L. C. Paulson, *Isabelle: A Generic Theorem Prover*, vol. 828 of *Lecture Notes in Computer Science.* Springer-Verlag, 1994.

[27] F. Rabe and M. Kohlhase, "A web-scalable module system for mathematical theories," *Journal of Symbolic Computation*, 2009.

[28] A. Riazanov and A. Voronkov, "The design and implementation of vampire," *AICOM*, vol. 15, pp. 91–110, 2002.

[29] The FoC Development Team, *The FoC Reference Manual, Version 0.0*, 2003. Available at `http://www-spi.lip6.fr/foc/`.

[30] S. M. Watt, P. A. Broadbery, P. Iglio, S. C. Morrison, J. M. Steinbach, and R. S. Sutor, "Aldor compiler user guide." Available at `http://www.aldor.org/docs/HTML/`, 2001.

[31] J. Xu, "Mei — a module system for mechanized mathematics system," in *Proceedings of Programming Languages for Mechanized Mathematics (PLMMS 2007)* (J. Carette and F. Wiedijik, eds.), (Hagenburg, Austria), p. 17, June 2007.

[32] J. Xu, *Mei — A Module System For Mechanized Mathematics System.* PhD thesis, McMaster University, 2008.

[33] H. Zhang, "A language and a library of algebraic theory-types," Master's thesis, McMaster University, 2009.