Article Submitted to Journal of Symbolic Computation

MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems

Michael Kohlhase[1] and Andreas Franke[2]

[1] *School of Computer Science, Carnegie Mellon University, Pittsburgh, USA*
[2] *Informatik, Saarland University, Saarbrücken, Germany*

Abstract

In this article we describe the data model of the MBase system, a web-based, distributed mathematical knowledge base. This system is a mathematical service in MathWeb that offers a universal repository of formalized mathematics where the formal representation allows semantics-based retrieval of distributed mathematical facts.

We classify the data necessary to represent mathematical knowledge and analyze its structure. For the logical formulation of mathematical concepts, we propose a methodology for developing representation formalisms for mathematical knowledge bases. Concretely we propose to equip knowledge bases with a hierarchy of logical systems that are linked by *logic morphisms*. These mappings relativize formulae and proofs and thus support translation of the knowledge to the various formats currently in use in deduction systems. On the other hand they define higher language features from simpler ones and ultimately serve as a means to found the whole knowledge base in axiomatic set theory.

The viability of this approach is proven by developing a sorted record-λ-calculus with dependent sorts and labeled abstraction that is well-suited both for formalizing mathematical practice and supporting efficient inference services. This "mathematical vernacular" is an extension of a sorted λ-calculus by records, dependent record sorts and selection sorts.

1. Introduction

The last five years have seen a growing interest in the integration of mathematical software systems, such as computer algebra systems and deduction systems. The reason for this is that while the respective systems have reached a high degree of sophistication and maturity, they have differing, often complementary

strengths and weaknesses, and no single system is strong enough to tackle all problems. Moreover, since many of the problems are very computation-intensive, distributing sub-problems to mathematical services over the Internet seems a promising approach.

Particular interest is in the combination of computer algebra systems (CAS) and deduction systems (DS), either for the purpose of enhancing the computational power of the DS (30; 39; 7) or in order to strengthen the reasoning capabilities of a CAS (1; 8).

We can distinguish four kinds of problems that have to be overcome for an integration of two mathematical software systems:

Syntax Though most systems have a term-based interface language, normally all systems will have their own particular variant. This problem can be solved by establishing representation standards, such as the emerging OpenMath standard (15), which uses Xml (13) to define a general term language. With the imminent wider acceptance of this standard, this problem will soon be solved.

Protocol The problems of low-level communication and common control protocols have been explored e.g. in (14) and have to be decided upon in the concrete application. Empirically, all such protocols and architectures can be flexibly modeled by agent-oriented programming; we have used this in the MathWeb system (27; 26), an agent-based implementation of a mathematical software bus that uses the current de-facto standard Kqml (24) for interaction agent-languages. Even though the Kqml-support in MathWeb is not fully implemented, we can see this problem as solved in principle (see (5)).

Semantics For the integration of systems it is crucial to specify concisely and without ambiguity the meaning of the exchanged formulae, i.e. there is the problem of establishing a semantics for the communicated mathematical objects. Otherwise the results of the integrated system can be arbitrary: Recall the recent incident of the NASA Mars lander, where NASA specified rocket thrust in metric units but the contractor used pounds and inches (as a result the probe crashed on Mars instead of landing). This is well-known as the so-called *ontology problem* in distributed artificial intelligence, the accepted solution to this is to either take recourse to a common set of concepts (the ontology, see (39) for a proposal wrt. the integration of computer algebra with proof planning) or to negotiate a private ontology for the communication. The OpenMath standard recognizes this and offers the mechanism of "content dictionaries": machine-readable, but *informal* definitions of the mathematical concepts involved. Note that in contrast to the practice in distributed artificial intelligence (agent-oriented programming), the ontology is determined local to the symbols of the terms instead of globally for the communication, which seems much more appropriate for the application in mathematics.

This is at best a partial solution to the semantics problem, since the OpenMath framework does not offer any support for ensuring consistency, conciseness, or manipulation of ontologies.

Context The context problem is a variant of the semantics problem, i.e. in the communication of two mathematical software systems (or more generally agents) it is advantageous to maintain a sense of shared state. For instance, the state can be used to refer back to (parts of) previous formulae, that are kept in the so-called context. Of course it is possible to eliminate state from the communication by retransmitting the relevant parts of the context, but this can lead to an exponential increase in costs. As a consequence almost all interactive mathematical software systems use some form of context for the communication with the user. Current approaches to integration of mathematical software systems cannot deal with context, or use it in a very inflexible way, for instance the Clam-Hol interaction (12), or the ΩMEGA-Tps (10) integration have to retransmit all the necessary definitions and subgoals on every round of interaction.

This article addresses the last two problems. We contend that a society of distributed knowledge base agents in MathWeb (27; 26) can be used to establish both the semantics of communicated formulae as well as provide a flexible notion of context. To substantiate this claim, we will present and discuss the MBase system, a web-based, distributed knowledge base for mathematics that is universally accessible through MathWeb on the Internet.

The mathematical knowledge in MBase can be used to establish a centralized reference point that establishes the *semantics* of formulae, since it is both machine-readable and fully-formal. Moreover, the knowledge base agents in MathWeb can be used as ontology servers for agent communication, in particular, they can manipulate small private knowledge bases as a service for other MathWeb services, effectively providing a flexible notion of *context*. In the rest of the article, we will describe the MBase server and its underlying data model. In particular, we address the question of how to divide the task of representing and reasoning with complex knowledge base entries, such as logical formulae in a data base application. These are typically very complex (possibly cyclic) graph structures that cannot be represented or reasoned about adequately in current SQL-based data base systems. On the other hand, high-level programming languages can do this, but the amount of data that can be processed is basically limited to the size of main memory. MBase adopts a hybrid approach that tries to combine the strengths of both worlds, eliminating their relative limitations.

The current implementation (see `http://www.mathweb.org/mbase`) is still largely a prototype for testing the design decisions. It consists of the MBase server, which acts as a MathWeb service, and an `http` server that dynamically generates presentations based on HtML or Xml forms. Other mathematical services can access MBase through a system of *mediators* that are also integrated into MBase.

The primary interface format of MBase is OMDoc (43; 42), an Xml-based representation language for MBase content. Since this is an extension of the emerging OpenMath standard (15) for web-based mathematics, its syntax is logic-independent. So the mediators can first do the logic-transformation, then generate the OMDoc representation, and then create the concrete input syntax of the respective reasoning system by invoking a standard Xml style sheet processor with a specialized Xsl style sheet.

Currently, connections to the Ωmega (9), InKa (36), *λClam* (51), and Tps (4) systems are being actively developed. Semi-automated reasoning systems like these usually store large amounts of mathematical data in a file-oriented library storage mechanism. For solving a given problem, all knowledge in the library that is possibly relevant must be loaded into main memory, obviously a very inefficient usage of this resource. In this situation, the MBase service, which uses data base technology for the storage aspect allows to load the knowledge incrementally, to perform finer-grained reasoning as to which knowledge will be relevant, and to browse the knowledge beforehand, so that the user can determine the actual desired knowledge elements.

1.1. Architecture: Division of Labor

The MBase system is realized as a distributed set of MBase servers (see Fig. 1.1). Each MBase server consists of a Relational Data Base Management System (RDBMS), e.g. Oracle, which is connected to a mozart (53) process via a standard data base interface (in our case JDBC). Clients can access MBase servers as MathWeb services, and for browsing the MBase content, any MBase server provides an `http` server (see `http://mbase.mathweb.org:8000` for an example) that dynamically generates presentations based on HtML or Xml forms.

This architecture combines the storage facilities of the RDBMS with the flexibility of a concurrent, distributed, logic-based programming language (see `http://www.mozart-oz.org`).

Most importantly for MBase, mozart offers a mechanism called **pickling**, which allows for a limited form of persistence: mozart objects can be efficiently transformed into a so-called pickled form, which is a binary representation of the (possibly cyclic) data structure. This can be stored in a byte-string and efficiently read by the mozart application effectively restoring the object. This feature makes it possible to represent complex objects (e.g. logical formulae) as Oz data structures, manipulate them in the mozart engine, but at the same time store them as strings in the RDBMS.

The current implementation of MBase can be used together with different kinds of data base engines: e.g. InstantDB (see `http://www.instantdb.co.uk`), a lightweight open-source java based program for scratch-pad databases, and Oracle for archive MathWeb servers. Thus the use of JDBC as a standardized

interface allows to achieve the somewhat conflicting functionalities needed for the distributed nature of MBase (see section 3).

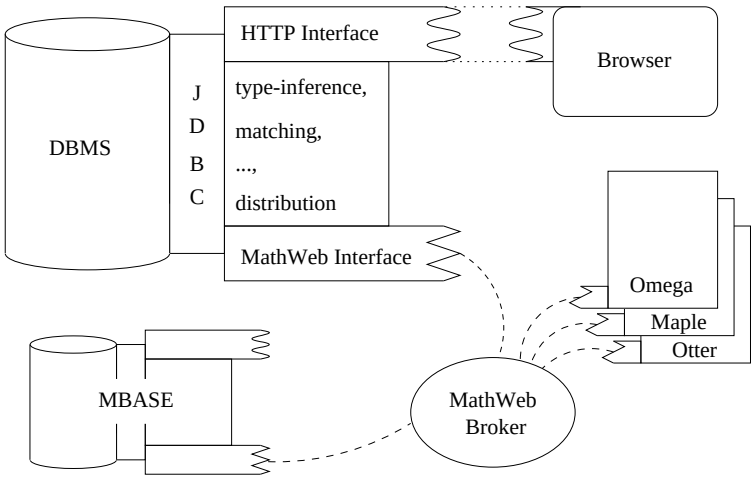

Figure 1: General System Architecture

1.2. An Example

In this section we will discuss a simple mathematical example (a version of Cantor's theorem), which will be used in the following.

> **Theorem 3.1.7** (Cantor): *Let S be a set, then S has a smaller cardinality than its power set $\wp(S)$.*
> **Proof**: We prove the assertion by diagonalization. Assume that there is a surjective mapping $F: S \longrightarrow \wp(S)$. Now let D be the set $\{a \mid a \notin F(a)\}$; we show that $D \notin \mathbf{Im}(F)$: if there were a pre-image $b \in S$ (i.e. $D = F(b)$), then assuming $b \in D$ we can obtain $b \notin D$, which is a contradiction.

The assertion of the theorem is about cardinalities of sets. Usually, the cardinality of a set S is defined to be smaller than that of T, iff there is no *surjective* mapping $F: S \longrightarrow T$. Alternatively, smaller cardinality can be defined as the absence of *injective* functions from T into S. A function $f: S \longrightarrow T$ is called surjective, iff for all $b \in T$, there is an $a \in S$ (called its pre-image), such that $f(a) = b$. The power set $\wp(S)$ of a set S is the set of all subsets of S. To illustrate these concepts it may be useful to look at a simple example: If S is the singleton set $\{a\}$, then the power set $\wp(S)$ is $\{\emptyset, S\}$; there are only two mappings from S to $\wp(S)$, $f_\emptyset: a \mapsto \emptyset$ and $f_S: a \mapsto S$, which are not surjective ($S \notin f_\emptyset(a) = \{\emptyset\}$ and $\emptyset \notin f_S(a) = \{S\}$). Thus our example supports Cantor's theorem.

In a formal reasoning system like ΩMEGA, Isabelle or Pvs, the theorem would be stated in a suitable logic, e.g. in the simply typed λ calculus

$$\forall S_{\alpha \to o}.\mathsf{smaller\_card}(S, \mathsf{powerset}(S))$$

where the symbols (constants of the logic) smaller_card and surj are defined as the λ-terms

$$\begin{aligned} \mathsf{smaller\_card} &:= \lambda M_{\alpha\to o}.\lambda N_{\beta\to o}.\neg\exists F_{\alpha\to\beta}.\mathsf{surj}(F,M,N) \\ \mathsf{surj} &:= \lambda F_{\alpha\to\beta}.\lambda M_{\alpha\to o}.\lambda N_{\beta\to o}.\forall X_\beta.NX \Rightarrow (\exists Y_\alpha.MY \wedge FY = X) \end{aligned}$$

Again, the symbol smaller_card could have been defined in terms of injectivity by a similar λ-term.

Based on this knowledge, the reasoning systems mentioned above can prove the theorem (fully automatically [Tps (11)] or interactively) by eliminating the definitions (substitution of the λ-term and subsequent β-reduction) and solving the problem at the level of the underlying calculus.

Another way to arrive at the proof is to encode the human problem solving knowledge for diagonalization proofs explicitly in the *proof planning* paradigm and use this method- and control knowledge to prove the theorem in much the same way as humans would. This results in a different, more structured proof of the theorem (16). Note that the textbook proof above also has two levels of description of the proof: one with the keyword "by diagonalization" which is sufficient for the expert to reconstruct a more detailed proof.

1.3. A Classification of the Relevant Knowledge

Already in the small example discussed above, we see that the statement of a mathematical theorem can depend on the availability of a (large) set of definitions of mathematical concepts (that in turn depend on other concepts). Furthermore, the proof can use previously proven theorems and lemmata, or even introduce new concepts. In addition to this purely mathematical data, a formal reasoning system needs access to other forms of knowledge (e.g. flag settings for automated theorem provers or method- and control knowledge in proof planning). For presentation to human users, other (human-related) presentation knowledge is needed. See e.g. (52), where we use MBase as a basis for the flexible presentation of an an interactive mathematics book (18).

The purpose of the MBase system is to store and manipulate all these kinds of knowledge with an emphasis on the use of structure to support an adequate information retrieval and search restriction. In this section, we will try to classify and structure them (see Fig. 1.3). This classification will serve to structure the database model presented in the next section.

As we have already seen above, we have to distinguish between purely mathematical knowledge (primary objects) and secondary objects that provide human- and machine-oriented or even administrative information or give additional structure. Concretely, we distinguish the following five categories in Fig. 1.3.

Primary objects for purely mathematical knowledge like symbols, their definitions, and theorems, lemmata, etc. and their proofs (cf. section 2.1).

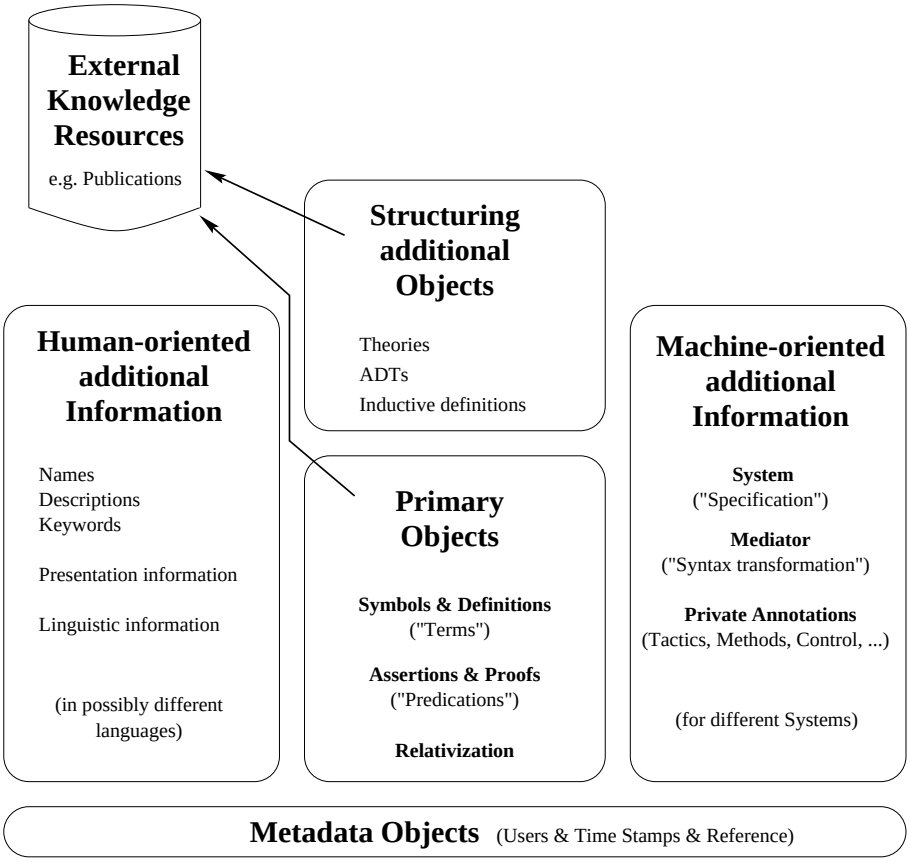

Figure 2: The structure of MBase Data

Human-oriented additional information, like names of theorems, special mathematical notations, or special linguistic specifications for text or speech generation systems (cf. section 2.2).

Machine-oriented additional information, provides similar knowledge for the interaction with automated reasoning services (cf. section 2.3).

Structuring Objects MBase uses a system of theories to reflect the large-scale structure of mathematics. Furthermore, special constructs for abstract data types and inductive definitions are supplied to ease and structure the specification of the mathematical objects (cf. section 2.4).

Relations to external knowledge resources like journals, citation databases etc.

2. The Database Model

In this section we will formalize and further elaborate the data base model of MBase discussed above. In particular, we make the model explicit by giving a set of data base declarations. Let us first concentrate on the primary data

base objects; before we present the data model, let us further classify them and discuss their relations.

Symbols for mathematical concepts, such as 1 for the natural number "one", + for addition, = for equality, or `group` for the property of being a group. Furthermore, there are symbols for kinds, types and sorts.

Definitions give meanings to symbols in terms of already defined ones. For example the number 1 can be defined as the successor of 0 (specified by the Peano axioms). Addition is usually defined recursively, etc. Definitions are separated from the symbols they define in MBase, since there can be more than one (equivalent) definition for a symbol in a mathematical theory, e.g. the smaller cardinality relation discussed in section 1.2. This phenomenon is made explicit in the relation **def-entails**.

A second reason for this division of concepts is that "universal" constants can be introduced as symbols without definition.

Assertions are axioms, theorems, conjectures, lemmata, etc. They all have the same structure: they are basically logical sentences. Their differences are largely *pragmatic* (theorems are normally more important in some theory than lemmata) or proof-theoretic (conjectures become theorems once there is a proof in the knowledge base).

Proofs are representations of evidence for the truth of *assertions*. Like in the case of definitions, there can in general be more than one proof for a given assertion. Furthermore, it will be initially infeasible to totally formalize all mathematical proofs needed for the correctness management of the knowledge base in one universal proof format, therefore MBase supports multiple formats for proofs or *evidence* such as e.g. a calculus-level proof, various proof scripts (ΩMEGA replay files, Isabelle proof scripts,...), references to published proofs, resolution proofs, etc. Therefore, a proof can have several

Proof Objects encapsulate the actual proof objects in the various formats. There can be more than one proof object for a given proof. Informal proofs can be formalized, formal proofs can be transformed from one format to the other (e.g. from resolution style to natural deduction style), and can even be presented in natural language by a proof presentation system like Proverb (34). Even so they represent the same "proof". In our example in section 1.2, we have described four proof objects for the same proof: the sketch consisting only of phrase "we prove the assertion by diagonalization", its elaboration in the textbook example, the Tps proof and the proof-planning proof.

The universal proof format used in MBase is derived from the **P**roof plan **D**ata **S**tructure ($\mathcal{PDS}$) introduced in the ΩMEGA system (9) to facilitate hierarchical proof planning and proof presentation at more than one level of abstraction. In a $\mathcal{PDS}$, nodes justified by tactic applications are expanded, but the information about the tactic itself is not discarded in the process

as in tactical theorem provers like Isabelle or NuPrL. Thus proof nodes may have justifications at multiple levels of abstractions in a hierarchical proof data structure.

Examples In mathematical practice, examples play an important role just as proofs, e.g. in concept formation (as witnesses for definitions or as either supporting evidence, or as counterexamples for conjectures). Therefore, examples are given status as primary objects in MBase, even though they are still very seldom actually used in mechanized reasoning systems. Conceptually, we model an example for a mathematical concept $\mathbf{C}$ as a triple $(\mathcal{W}, \mathbf{A}, \mathcal{P})$, where $\mathcal{W} = (\mathcal{W}_1, \ldots, \mathcal{W}_n)$ is an n-tuple of mathematical objects, $\mathbf{A}$ is an assertion of the form $\mathbf{A} = \exists W_1 \ldots W_n . \mathbf{B}$, and $\mathcal{P}$ is a proof that shows $\mathbf{A}$ by exhibiting the witnesses $\mathcal{W}_i$ for W_i. The example $(\mathcal{W}, \exists W_1 \ldots W_n . \neg \mathbf{B}, \mathcal{P})$ is a counter-example to an assertion of the form $\mathbf{T} := \forall W_1 \ldots W_n . \mathbf{B}$, and $(\mathcal{W}, \mathbf{A}, \mathcal{P})$ a supporting example for $\mathbf{T}$.

Consider for instance the structure $\mathcal{W} := (A^*, \circ)$ of the set of words over an alphabet A together with word concatenation $\circ$. Then $(\mathcal{W}, \exists W . \texttt{mon}(W), \mathcal{P}_1)$ is an example for the concept of a monoid (with the empty word as the neutral element), if e.g. $\mathcal{P}_1$ uses $\mathcal{W}$ to show the existence of W. The example $(\mathcal{W}, \exists V_{\texttt{mon}} . \neg \texttt{group}(\texttt{V}), \mathcal{P}_2$ and a proof that uses $\mathcal{W}$ as a witness for V, it is a counterexample to the conjecture $\mathbf{C} := \forall V_{\texttt{mon}} . \texttt{group}(\texttt{V})$, since $\mathbf{Q} \Rightarrow \neg \mathbf{C}$.

All in all, we have the structure given in Fig. 2 for the primary objects. In the following we will briefly discuss the concrete realization of the primary objects in MBase and then go on to discuss the other categories of database objects from Fig. 1.3. The metadata used in MBase is relatively standard, they include things like bibliographic reference (we use the well-known Dublin Core schema, cf. `http://purl.org/dc/` or see (43) and things like time stamps and user reference for creation and modification of objects.

2.1. Modeling Primary Database Objects

To implement the primary knowledge elements described above, MBase currently uses tables for the six primary objects and a variety of relations. This realization of the data model is geared towards an underlying SQL data base, and can be subject to change, when suitable object-oriented DBMS become available.

`symbol` The type of a symbol must be unique, it is represented as a pickled mOZart object (indicated in the data type OzPickle). For the data base, this is a string of arbitrary length. MBase uses OzPickles for complex (logical) data structures, which can be read into the mOZart process for logical processing.

`definition` At the moment, MBase supports simple, inductive/recursive, and implicit definitions as primary objects. In the latter case, the content of the

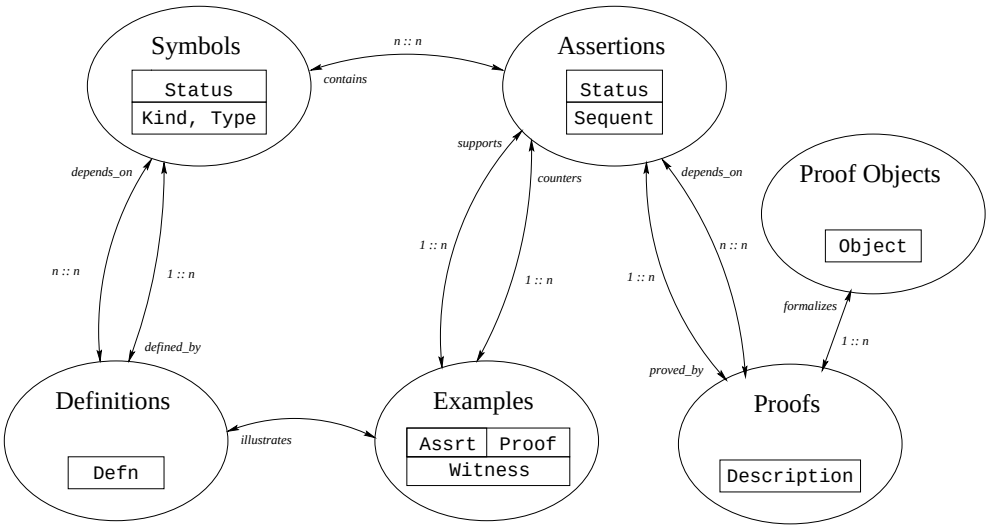

Figure 3: The structure of MBase

definition, which is a complex term (and therefore expressed as an OzPickle) can define more than one the `symbol`.

`assertion` Assertions are logical formulae (represented as OzPickles) that have a `status` flag that represents the pragmatics of theorem-hood. At the moment MBase supports the values `problem`, `axiom`, `theorem`, `lemma` for the `status` attribute.

`proof` Proofs are general descriptive objects that represent proof ideas. They serve as objects that for the relations `proof-depends-on` and `proved-by`. The intuition behind this decision is that if two proof objects depend on different definitions/assertions, then they are different "Platonic" proofs. In particular, if an informal proof (say from a mathematical textbook) is formalized in some calculus and additional dependencies become apparent, then these are also (implicit) dependencies of the original, informal proof.

`proof-object` Since there are as many proof formats as deduction systems and mathematical traditions, we cannot make any assertion about the representation of proof objects at the moment. Instead we assume the least common denominator and provide strings of unbounded length for the proof objects assuming that deduction systems can always write proofs to files.

Certain proof formats, like ND proofs and $\mathcal{PDS}$ can be represented as λ-terms, which are supported by the MBase logic, so these can be encoded as Oz-pickles. This has the advantage that the `depends-on`-relations can be automatically checked or computed by MBase. It is intended to support more and more proof formats directly in MBase in the future, so that machine support can be extended.

`example` As examples are just triples consisting of an object, an assertion and a proof, their structure is very simple. The three relations of illustrating a

concept, supporting/countering a universal theorem mentioned above are condensed in to one, with intended meaning specified by a `role` attribute.

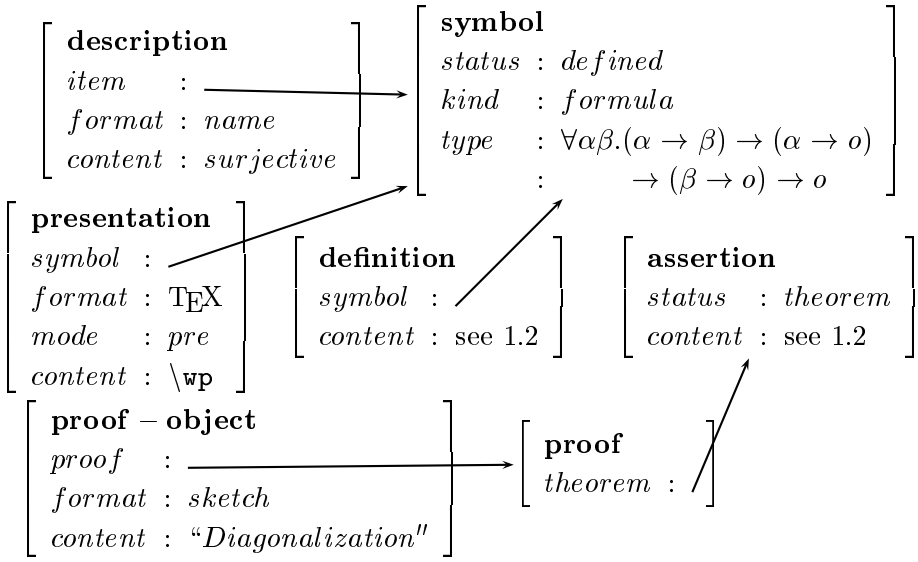

Figure 4: Example Records for "surjective" and Cantor's Theorem

The relations in Fig. 2 contain the data for the list-valued slots in the primary objects. When we upgrade the database model to an object-oriented paradigm, e.g. the emerging standard OQL, the binary many-to-many relations will be represented as methods.

definition-entailment A symbol may be primitive (in which case its `status` must be `primitive`) or defined. In the latter case, it can have more than one definition, all of which must be proven equivalent. MBase stores these equivalence theorems as the set of entailment theorems for a given symbol given by the relation `def-entails`, where the value of the `theorem` attribute must be of the form "`Item` ⇒ `Entailed-Item`".

The DBMS ensures that for any defined symbol, the `def-entails` relation must be connected on the set of its definitions (i.e. any pair (d, d') of definitions must be in the transitive closure of **definition-entailment**).

depends-on/local-in These relations specify dependency and locality information for primary knowledge elements. These are invaluable for definition and proof expansion, e.g. during proof verification and for structuring the knowledge in the repository (see section 2.4).

Actually, this relation is currently implemented by sub-relations `def-depends-on`, `proof-depends-on`, and `contains`, which make explicit which symbols/lemmata are used in a definition or assertion, and a relation `theory-depends-on`, which specifies the inheritance relation among theories.

2.2. Human-Oriented Information

In this section we will address the database facilities that provide the knowledge necessary for presenting the primary knowledge to humans, which will serve as input to mediators between the MBase and the presentation services. The intention of storing such knowledge (even the choice of the mediator itself) in a centralized knowledge base server is that this information serves only as a default, which can be overridden by local personal preferences. The mediators, which we envision as Oz functors (Ozlets) are a good tool to implement a flexible and customizable presentation component.

description This relation annotates primary objects with descriptive strings, the `format` slot specifies whether the string is a proper description, a name (e.g. for a named theorem like Gentzen's "Hauptsatz"), keywords and the like. They give sets of supplementary (administrative and search) information for the objects.

presentation These objects represent the presentation information for symbols in various natural languages, presentation formalisms (such as ASCII, MathML (37), LaTeX, HtML (50), ...) or fonts. It is a central concern in MBase to separate content information from presentation information, therefore, we have not included the presentation information into the symbols themselves.

As we have mentioned above, the primary interface language for MBase is the Xml-based OMDoc, which is geared towards semantical markup. The presentation markup in formats as the ones mentioned above is often generated using a so-called Xsl (20) style sheet (i.e. a set production rules for presentation markup) by an Xsl transformer (the rule interpreter). The upcoming generation of Internet browsers like Mozilla, Netscape Navigator 6, or MS Internet Explorer 5 contain integrated Xsl transformers and can thus be used to view the presentation form of the OMDoc representations directly.

The information needed for the Xsl style sheets is partly global (mostly pertaining to the grammar of the format and the default appearance of symbols; this is specified by the style sheet designers), and partly local to the symbols (a specialized production rule whose head matches the Xml element for the respective symbol; and can specified in the presentation objects). Thus a presentation object normally contains an Xsl production rule tailored to a particular format.

Thus for each OMDoc document $\mathcal{D}$ generated by the appropriate mediator for the interaction with a human user, MBase also generates a specialized style sheet from the presentation objects of all the symbols used in $\mathcal{D}$. Together these result in a presentation in the desired output format.

MBase also supports an abbreviated form of the presentation objects, that only contains a string (e.g. the string `\subseteq` for representing the subset relation

$\subseteq$ in TEX) and a `mode` token which controls whether the string is inserted in a prefix/infix/postfix way. The appropriate Xsl-presentation is then computed from these values on the fly. Finally, if the `mode` is `def`, then the presentation object can be an Oz-functor that produces the presentation object from the necessary arguments. This possibility for writing presentation objects is more flexible than the one above, but certainly less declarative and portable.

2.3. Machine-Oriented Information

Next to the presentation of knowledge to human users, the presentation of formulae to different *mathematical services* is a central issue in MBase. Different theorem provers currently have vastly differing communication formalisms, which may differ both in the underlying logic, as well as in the concrete syntactical representation used. The latter issue is a largely software-technological issue that can be solved by either standardizing the language (e.g. by our OMDoc format), and/or by the mediator approach (implementing a translating mediator for any language pair). The issue of the underlying logic is more serious, since the nature of the logic directly influences the applicability and efficiency of a given mathematical service.

In section 4 we present a system of languages interconnected by *relativizations*, i.e. logical morphisms that map formulae and proofs from more expressive languages to less expressive ones. Since so far, all occurring logical morphisms could be given in terms of definition expansions, MBase provides a grouping construct for logical morphisms, and a mediator that does definition expansion wrt. to this set of definitions. In this architecture, MBase keeps a table that maps mathematical services to logic morphisms, and when it outputs formulae to this system first applies the appropriate logic morphism (by the relativization mediator) and then the appropriate syntax generator for this system. For input from another mathematical service, it only uses the parser.

Furthermore, many of the mathematical services that will use MBase as clients maintain specialized mathematical knowledge which they need for theorem proving. For instance, InKa and *λClam* annotate terms with so-called wave-fronts/holes, or more generally colors. Tactical theorem provers need to keep store and retrieve their tactics, whose format differs from system to system. Proof planners like Ωmega, Clam or *λClam* furthermore have specialized methods and control knowledge. Proof presentation systems like Proverb (34) need to store linguistic knowledge about the mathematical concepts they present in natural language.

All of this "private" supplementary information shares the fact that it is intimately connected to the knowledge elements already in MBase. Moreover, most of this knowledge is now stored in special files in the respective systems. Therefore MBase offers the possibility to store these files in special knowledge elements that can store long byte strings. Storing this knowledge in MBase as opposed to storing it in the service has the advantage that the knowledge can

participate in the structuring mechanisms provided by MBase, thus enabling "just-in-time" loading of the necessary information. Note that MBase does not make an efficient management in the theorem prover unnecessary, but only gives the necessary infrastructure to cope with large sets of information.

Over time, the general availability for study of the data for private annotations may even lead to cross-system adoption of the underlying intuitions and in the long run even to standards in representing the involved knowledge.

2.4. Structuring the Knowledge base

In almost all library systems of proof development environments (see e.g. (IMPS; IsabelleKB; ILF; PVS)), the set of knowledge elements is structured by a so-called "theory" concept. Theories group sets of knowledge elements into subsets that e.g. are to be loaded at the same time. In some systems, like ΩMEGA and Imps (23), theories are simple sets of elements, in others, like Isabelle or Pvs, they can be parameterized. In MBase we use techniques from the field of algebraic specification (see for instance (45)), where the structure of large-scale formalizations (of the intended meaning of programs) have been studied in detail. Concretely, we adopt the concept of a "development graph" put forward by Dieter Hutter (35), since this supplies a simple set of primitives for structured specifications and also supports management of theory change. Furthermore, it is logically equivalent to a large fragment of the emerging Casl standard (17) for algebraic specification (see (6)).

A development graph specifies the large-scale structure of a set of theories (i.e. sets of symbol declarations, their definitions, and axioms). It is a graph where the nodes are theories and the arcs are given by theory morphisms. The latter come in two categories: **import morphisms** and **inclusions**, both of which can be local and global. A set of import morphisms *define* (part of) a theory by specifying what material (symbols, definitions, axioms) is imported from existing theories. Since the material can be imported modulo a language morphism (i.e. it is translated before it is included into the new theory), this is a very powerful definition mechanism. We can for instance define a theory of rings given as a tuples $(R, +, 0, -, *, 1)$ by importing from a group $(M, \circ, e, i)$ via the morphism $\{M \mapsto R, \circ \mapsto +, e \mapsto 0, i \mapsto -\}$ and from a monoid $(M, \circ, e)$ via the $\{M \mapsto R^*, \circ \mapsto *, e \mapsto 1\}$, where R^* is R without 0 (as defined in the theory of monoids).

Inclusions are of a different nature: instead of defining a theory, they state structure information that can be inferred about a theory hierarchy. Like the import morphisms, inclusions are theory morphisms (the translations of all theorems of the source theory must be theorems of the target theory). Only that in contrast to the former, who have this property by definition, the inclusions have to be verified. Once they are established, they can be used to transport results and proofs from the source to the target theory, for instance, many algebraic domains like groups have a self-inclusion that is induced by the involution with the

inverse element. In many proofs, this inclusion can be used to transport proofs for symmetric cases instead of re-proving them. Moreover, the structure of the development graph can be used to support a "management of change" (see (35)). For instance it is often necessary during theory exploration and development to change definitions and axioms, invalidating proofs of theorems that use them. The theory structure can be used to specify the dependency relations and save valuable theorem proving time, the more (redundant) structure we have in a development graph, the more reusable and less brittle proofs become. To pinpoint the the contribution of individual axioms and definitions, the development graph divides morphisms and inclusions into global and local variants. The local versions only concern the axioms and definitions directly defined in the source theory, as a consequence, the global ones can be seen as transitive completions of the local ones. The user only specifies the global morphisms, while the system mainly works with the local decompositions that allow a more fine-grained analysis of the theory structure.

MBase provides data structures for the development graph and implements Hutter's "management of change"

Like the library systems of many practically used deduction systems, MBase views **abstract data types** as abbreviations for sets of definitions, axioms and theorems. For example, the abstract data type `Nat` that is specified by the constructor definitions for zero and the successor corresponds to the well-known Peano Axioms for the natural numbers. If we also specify the selector function "predecessor" for the successor function, then e.g. the corresponding commutation laws can be automatically generated. Again, we represent this by introducing data base objects for ADTs and group the corresponding definitions and using the `local-in` for grouping. Other definition mechanisms, such as those for e.g. the various classes of recursive functions can be handled in the same way.

3. Distributing MBase

In this section, we will extend the MBase data model presented above to support a distributed data model, and we will specify some of the management routines pertaining to distribution.

With the distribution MBase supports repositories from the archive server level, where large parts of formalized mathematics are kept centrally, to the personal level, where a researcher has a personal MBase to manage her mathematical theories under development. Inbetween there may be workgroup or institute servers, that support collaborative development of mathematical theories.

To get a feeling for the requirements of distributing MBase, let us take a look at a likely research communication scenario: We will first describe the communication pattern as it could have happened in the era when mathematics was done with pen and paper (around 2001), and then model it using distributed MBases (about 2005).

classical, see Fig. 3 Researcher R works on Theory T together with his col-

league R' at institute I. The theory T is a body of mathematics laid down in an article A published in journal J. Now, R extends theory T by a new definition D (say for a mathematical object O), proves a set P of theorems about O, and calls the resulting extended theory E. After that, R tells her colleague R' at I about D and P (say by circulating a memo in I), who gets interested and proves a set P' of useful properties of O. Together, R and R' put the theory E into final form F, and submit it to journal J. This accepts F and publishes it.

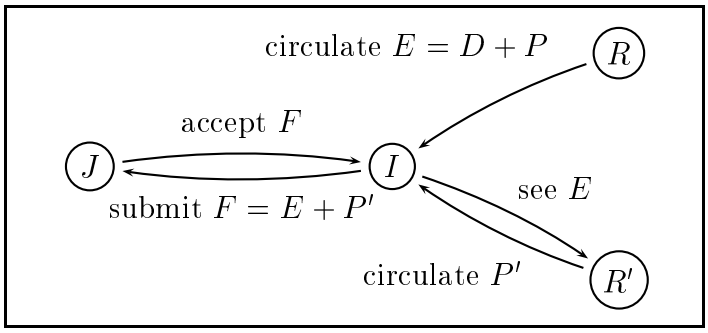

Figure 5: Classical Research Cooperation

with MBase**, see Fig. 3** In 2005, J and I have joined the MathWeb initiative, in particular, J has established an MBase server MJ for the journal J and has formalized (with the help of researchers from I) theory T, which now resides in the MBase server MJ. Furthermore, the institute has its own departmental MBase MI and the researchers R and R' have the personal MBases MR and MR'. Now, R develops the formalization FD of O, stores it in MR and formalizes the set P of theorems by formalizing them and formally proving them (yielding FP in MR). To do so, R may need to revise the initial version of D several times in order to be able to prove the desired theorems (reproving the already obtained results that depended on a previous version of D every time). This process will be supported by MBase based on techniques presented in (35), but this is outside of the scope of this article. Instead of sending around an internal note about D and P in I, R moves their formalizations FD and FP into the institute MBase server MI, from where R' can import them into his personal mbase MR'. Alternatively, R could leave FD and FP in MR and tell R' personally about them, allowing him to import them from MR into MR'; but this is a matter of institute policy, which we will not address in this article. On this basis R' formally proves FP', and adds it to theory FE, yielding FF the formal version of theory F. Then R and R' submit F to journal J, who evaluates it (possibly via his own personal MBase) and finally accepts F. To publish F on MJ, it requests FF from MI, which moves it there.

We believe that the latter (more complicated) picture is better than the sim-

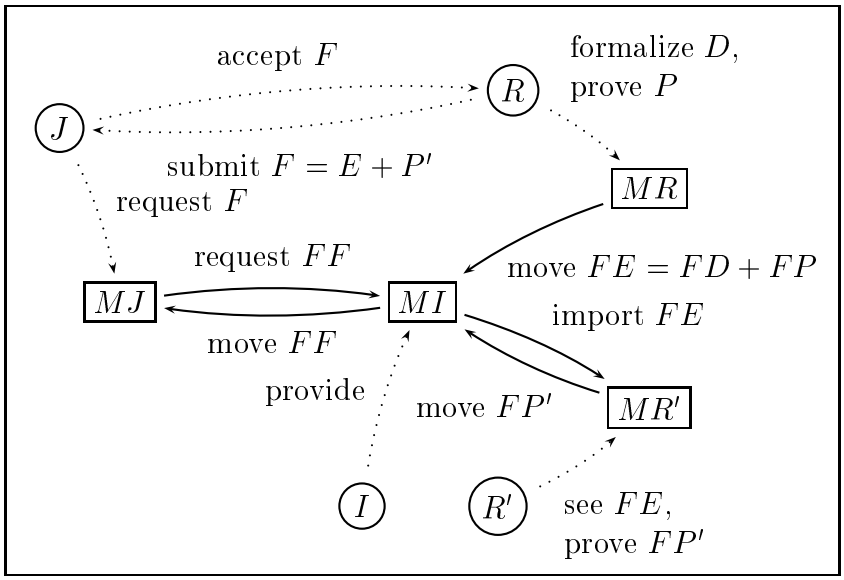

Figure 6: Research Cooperation with distributed MBases

ple pen-and paper method for managing, archiving and communicating mathematical theories, since the formalization gives more precision to mathematical arguments and the identification of mathematical concepts. In pen and paper mathematics intuitively clear and commonplace concepts like the natural numbers ($\mathbb{N}$) are often used without a precise definition, which can even result in mis-quotation or mis-application of theorems, since it is unclear whether zero is included in the set $\mathbb{N}$.

Many of the advantages that can be reaped from the MBase scenario for mathematics come from the hyperlinking possibility given by distribution and Internet-availability of MBase – most importantly by the unique referencing scheme – developed in this article.

There are other issues to be considered for this vision: For instance, mathematics communication is very document-centered (articles, books, technical reports), and there should be a way to map MBase contents to some form of documents. In (43) we develop an Xml-based meta-language OMDoc (this is an extension for the emerging OpenMath standard (15)) for annotating mathematical documents that also serves as a communication interface to MBase. As a consequence it will be possible e.g. to generate customized OMDoc documents from MBase, which can then be presented in one of the more standard presentation media (e.g. LaTeX, HtML, or MathMl).

3.1. The Distributed Data Base Model

For distributing MBase, we make four assumptions (we will relax the last two below):

A1 the distributed MBase processes can be reached via the Internet (by URL),

A2 they are essentially uniform; e.g. realized by the same program, or at least communicate by the same protocol (see (5) for one based on KQML).

A3 primary objects are realized only once in the network of MBases. With this we mean that there is one "defining" instance of each primary object. As a consequence, every primary MBase object has a unique description: a pair consisting of the URL of the MBase and the unique identifier of the object there.

A4 primary objects are never changed. This assumption is useful, since it makes caching and maintenance much simpler. It is reasonable, at least for published mathematics, since changing e.g. a definition or theorem that other mathematical objects depend on is desasterous for overall consistency.

Note that we cannot make a unique representation assumption similar to **A3** for relations between objects. For instance the definition D of the object O from the example above will probably contain symbols that reside in MI or MJ, therefore, the `depends-on` relation for D cannot be localized to MR. The solution here is to introduce **reference objects** into MR, that point to objects, say in MI or MJ.

Definition 3.1 (Reference Object): Reference objects are database objects that refer to primary objects located in remote MBases. Technically, they are pairs $(\mathcal{M}, \mathcal{I})$ that consist of the URL $\mathcal{M}$ of the remote MBase and the unique object identifier $\mathcal{I}$ there.

If $\mathcal{M}$ is the current MBase and $\mathcal{I}$ is the unique identifier of a reference object $(\mathcal{M}', \mathcal{I}')$ in $\mathcal{M}$ (i.e. instead of a primary object itself, $\mathcal{M}$ has a reference to an object $\mathcal{O}$ stored in the remote MBase $\mathcal{M}'$ under the unique identifier $\mathcal{I}'$), and $\mathcal{M}$ is queried for $\mathcal{I}$, then $\mathcal{M}$, can forward the query (e.g. using the KQML forward performative; cf. (24)) to $\mathcal{M}'$ as a query for $\mathcal{I}'$, to which $\mathcal{M}'$ would answer by sending $\mathcal{O}$ to the original querying agent. Of course there is no guarantee that $\mathcal{I}'$ points to a primary object in $\mathcal{M}'$, so that the process might be iterated. Therefore, $\mathcal{M}$ also tells the querying agent that it only has a reference object, so that it can – e.g. if it is also an MBase – update reference information.

3.2. Managing distributed MBases

Let us now look at the management of distributed MBases. In this article, we do *not* specify policies for managing MBase contents, but discuss the infrastructure and processes necessary to efficiently manage the distribution aspects of a distributed mathematical knowledge base.

One of the most basic procedures is that of moving data between MBases, e.g. of the theory FF from MI to MJ after the submission described in our scenario. This is realized by "moving" the primary objects and parts of the relations from MI to MJ.

Concretely, a primary object $\mathcal{O}$ (with unique identifier $\mathcal{I}$) is moved from $\mathcal{M}$

to $\mathcal{M}'$ by creating a new object $\mathcal{O}'$ (with identifier $\mathcal{I}'$) in $\mathcal{M}'$, and replacing $\mathcal{O}$ in $\mathcal{M}$ by a reference object $\mathcal{O}' = (\mathcal{M}', \mathcal{I}')$. Now, all tuples in relations that are anchored in $\mathcal{O}$, are moved to $\mathcal{M}'$ by deleting the tuple in $\mathcal{M}$ and augmenting the corresponding relation in $\mathcal{M}'$.

Definition 3.2 (Anchored): Every MBase relation defined in section 2 has an anchor feature. This is the first feature in the attribute value-maps (e.g. in Fig. 2.1) of the database records representing the relations. If f is the anchor feature of a relation R, then we say that R is **anchored** in f.

With assumption **A4**, we can use a very simple model for caching. Since primary objects never change, they can be cached, and cache-consistency is never a problem. To allow caching, we simply relax assumption **A3**, and permit *cached copies of primary objects* to exist in other MBases. We still insist on a variant of **A3**, i.e. that there is only one **defining instance** of a given primary object; all others are called **cached**.

We implement the caching scheme by augmenting the primary objects by a flag `cached` that marks a primary object as a **cache copy object** or as a **defining instance**, and the reference objects defined in 3.1 by a **cache reference feature** that points to (contains the unique identifier of) a cache copy object. We assume that the database maintenance algorithm, whenever it decides to make a cache copy of an object $\mathcal{O}$ (copying it from MBase $\mathcal{M}$), also copies from $\mathcal{M}$ all relation tuples anchored at $\mathcal{O}$ and augments the local relations with them. Now, the knowledge base algorithms can access cache objects just like defining instances: whenever they hit a reference object, they either access the cache copy object specified in the cache reference feature or (if that is empty) access the remote copy of the object. Cached objects can be removed without loss of information as long as the cache reference feature of the corresponding object is reset.

Sometimes there are situations where it is necessary to change a definition, e.g. if an error occurred in the formalization. We have assumed in **A4** that primary objects may not change, so the only way to repair the error is to create a new definition object in the knowledge base and only use that subsequently. This is possible and even feasible, since mathematical concepts in MBase are not primarily identified by their technical names but by their identifiers (which will be different by **A3**) even if the technical names coincide. We could even give the old object the status "obsolete" to warn anyone against using the old definition. Even if this is successful, it is in principle impossible to determine when it is possible to delete the old definition, since other MBases might still be referencing it.

A similar situation occurs when a primary object is moved from MBase $\mathcal{M}$ to $\mathcal{M}'$, and is not referenced in $\mathcal{M}$ anymore (this will frequently happen, if completed theories are moved to higher-level MBase servers, such as the archive server MJ in our scenario). Therefore an MBase $\mathcal{M}$ keeps a record of all the MBases referring to it: we call those MBases **dependent** on $\mathcal{M}$. When an MBase $\mathcal{M}'$ creates a reference object pointing to a primary object in $\mathcal{M}$, and it

is not already dependent on $\mathcal{M}$, then $\mathcal{M}'$ sends $\mathcal{M}$ a message introducing itself as a new dependent. This list of dependent MBases allows two optimizations:

1. Whenever $\mathcal{M}$ moves an object $\mathcal{O}$ to some MBase $\mathcal{M}''$, creating a reference object $(n, \mathcal{M}'', \mathcal{I}'')$, then it can send the new location of $\mathcal{O}$ to all dependent MBases, asking them to update their reference objects and thus shielding itself from future requests to $\mathcal{O}$.
2. If $\mathcal{M}$ itself does not reference an object $\mathcal{O}$, it can ask all its dependents whether they do. If not, $\mathcal{M}$ can delete $\mathcal{O}$.

In particular if an MBase $\mathcal{M}$ does not have dependents, then we are totally free to change, delete, or otherwise manipulate data, as long as internal consistency is guaranteed.

3.3. Managing Context with MBase

Conceptually, there are two kinds of MBases that differ in their policy towards data change, we call them **archive** and **scratch-pad** MBases.

1. An archive MBase is epitomized by the Journal MBase MJ in our scenario above, it archives unchanging mathematical knowledge and is referenced by many other MBases.
2. A scratch-pad MBase is epitomized by the personal MBases MR and MR', these do not have any dependents and are primarily used for theory development.

Since they have different purposes, they have will have different structures. For example, the amount of data contained in an archive server will in general be much larger, making sophisticated database support necessary, while scratch-pad databases will have to support theory revision algorithms like the "management of change" (35) alluded to in section 2.4, but the InstantDB database support currently implemented in MBase may be sufficient.

The two classes of MBases will have radically different policies towards deleting and changing data, one way to implement these is to disallow dependent MBases in scratch-pads.

In particular, the lightweight scratch-pad MBases can be used to emulate context server agents. Whenever a set of mathematical services needs a notion of shared context (as opposed to a private notion of state, e.g. in a constraint solver service), then they can request an MBase to store it, e.g. as a special theory. Whenever a participating service needs to access the context, it will just issue a knowledge base query or manipulation command.

This approach, where the context is stored externally to the participating mathematical services is more flexible (e.g. services can be called into, or leave the problem solving at arbitrary times) than a more classical approach, where context is stored and manipulated inside the services. Furthermore, it reduces context manipulation to knowledge base access and thus reduces implementation

complexity. Finally, knowledge base services could ultimately offer added-value services, such as proxying or pro-active lookup.

4. Logics, Morphisms and MBase Languages

The logical language supported by MBase is a polymorphically typed, sorted record λ-calculus modeled after the mathematical everyday language (often called "mathematical vernacular", e.g. (19)). It is a joint generalization of the ML-polymorphic λ-calculus with kinds as used in Isabelle and Hashimoto & Ohori's polymorphic record calculus (47). Records allow a clean formalization of mathematical structures, such as groups or fields, polymorphism is needed to reuse definitions and theorems in the knowledge base and ensure a modular structure of the theory. Finally the mechanism of "kinds" adds to the practical expressivity of the polymorphism and is used in many theorem proving systems (λ*Clam*, Isabelle,...). Finally, the MBase logic supplies the infrastructure for sorted λ-calculi (see section 5). Conceptually, sorts are unary predicates (corresponding to often-used sets in mathematics) that are treated specially in the inference procedures (sorted matching and unification). This added structure leads to a more concise representation and a more guided search. For clients that cannot manipulate sorts, types, records, or higher-order quantification, the mediators built into MBase can relativize these language features away, retaining the intended meaning.

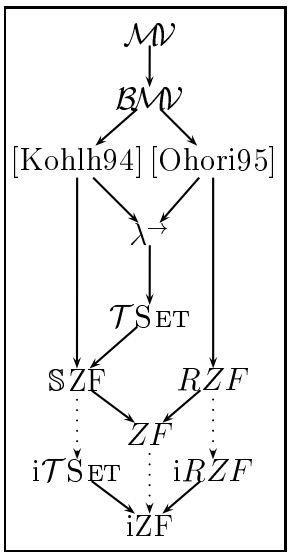

Figure 7: Hierarchy

We will use a variant of the theory interpretation approach proposed in (22) for **relativization** mappings, that can be used to transport meanings and proofs between logical formalisms. In fact, in the rest of the article, we will describe a whole hierarchy of representation languages (see Fig. 7), where relativizations can be used to arrive at various representation formalisms for mathematics, down to axiomatic (Zermelo-Fraenkel) set theory. Before we formally define the notion of relativization by the concept of **logical morphism** in the next section, let us discuss the consequences for the architecture of MBase.

The defining intuition for logic morphisms is that

> **Logic Morphisms Transport Proofs**: Let $\mathcal{F}:\mathcal{S}\longrightarrow\mathcal{S}'$ be a logic morphism and $\mathbf{A}$ an $\mathcal{S}$-theorem, then $\mathcal{F}(\mathbf{A})$ is an $\mathcal{S}'$-theorem.

This already suggests the logical structure of a mathematical knowledge base: Orthogonal to the usual theory hierarchy (induced by theory interpretation morphisms; we will not go into in this article, see (22)), there is a hierarchy of logical system induced by logic morphisms. In Fig. 7, we have specified some of the logical systems we will discuss in this article.

Mathematical knowledge can be specified in any of the logical systems; it can be queried and retrieved in any logical system that is downward accessible from this one. Furthermore, communication of mathematical software systems is possible by way of the "least common denominator logic". This may seem as a severe restriction of applicability of the approach, but it is not since the set of logical systems and morphisms in the hierarchy is not necessarily fixed:

- A new logical system can be incorporated by specifying a logic morphism to any of the existing systems.
- A new logic morphism can be added, if it is consistent with the information already present in the structure, i.e. if it is *redundant*.

Of course these hierarchy extensions generate proof obligations (determining the logic morphism property and redundancy), which will have to be supported in a system like MBase. We leave a discussion of this to another article.

The practical usefulness of a language hierarchy will depend very much on the existence of such redundant morphisms. In particular for the "least-common-denominator" problem between languages $\mathcal{L}$ and $\mathcal{L}'$ we can have two kinds of situations:

- If there is a good and well-understood way to translate formulae from language $\mathcal{L}$ to $\mathcal{L}'$, then we can implement this as a redundant logic morphism in MBase bypassing the need of an intermediate "communication logic". Moreover, making the logic morphism available in MBase will allow other users to use it.
- If there is no such translation, or if it is very domain-specific, then (of course) logic morphisms will not help (only further research into the semantic relation between the logics and possible translations will).

In the rest of this section, we will make the relativization approach concrete. We will first look at the elimination of sorts from sorted first-order logic by relativization. Based on the this guiding example, we we discuss the logical foundations and the relation to set-theoretic semantics in section 4.2. We will conclude this section by a discussion of the relativization of higher-order logic into first-order logic, in order to complete the lower half of the diagram in figure 7, before turning to the upper half in section 5.

4.1. Example: Relativizing Sorted First-Order Logic

In this section, we will consider relativization from sorted first-order logic to classical first-order logic. We will use the simply typed λ-calculus (32) as a meta-logical framework for representing the logical systems, since it gives us substitution, replacement and the treatment of bound variables for free. This is only a notational convenience and of no fundamental importance. In particular, this does not make any prerequisites on the part of the logical systems like first-order logic presented in this chapter. $\mathrm{FOL} = (\mathcal{L}^{\mathrm{FOL}}, ND(\mathrm{FOL}))$ is the

FOL	$= (\mathcal{L}^{\mathrm{FOL}}, ND(\mathrm{FOL}))$		
Signature	ι	Type	Individuals
	o	Type	Truth Values
	$\wedge$	$o \to o \to o$	Conjunction
	$\neg$	$o \to o$	Negation
	Π^ι	$(\iota \to o) \to o$	Universal Quantification
$\mathcal{L}^{\mathrm{FOL}}$	$=$ well-typed formulae of type o		
ND(FOL)	$\dfrac{\vdots}{\vdash^\iota_\Sigma \Pi^\iota \mathbf{B}}\,\Pi^\iota E$		$\dfrac{\vdots}{\vdash^\iota_\Sigma \mathbf{A}X}\,\Pi^\iota I$

Figure 8: First-Order Logic

logical system, where the logical part of the signature consists of the type constants o and ι (for truth values and individuals) and the term constants $\wedge, \neg$, and Π (see Fig. 4.1, all other connectives can be defined from $\neg$ and $\wedge$, by De Morgan rules, and quantification can be regained by treating $\forall X.\mathbf{A}$ as an abbreviation of $\Pi(\lambda X.\mathbf{A})$). The signature of first-order logic can contain further non-logical constants (called **parameters**) that model mathematical structures. In the following, all arguments and constructions will be parametric in the choice of parameters in the signature, and we will use the more precise FOL(Σ) for the instance of FOL that contains the parameters declared in the signature Σ.

$\mathcal{C} = ND(\mathrm{FOL})$ is the well-known calculus of natural deduction introduced by Gerhard Gentzen in (29). We will use $\vdash^\iota_\Sigma$ to abbreviate $\vdash_{ND(\mathrm{FOL})}$ (in Fig. 4.1, we have only depicted the quantifier rules, since they will be the only interesting ones for the discussion in this article).

The logical system $\mathbb{S}$FOL (see Fig. 4.1) is an extension of FOL, where the signature is extended by an order-sorted set $\mathbb{S}$ of sorts, a sorted quantifier Π^* and a set of constant- and subsort declarations (again, we will make use of higher-order abstract syntax here and write the traditional $\forall X_{\mathbb{A}}.\mathbf{B}$ as $\Pi^*\mathbb{A}(\lambda X.\mathbf{B})$.). The language $\mathcal{L}^{\mathbb{S}\mathrm{FOL}}$ is the set of well-sorted formulae, i.e. formulae, where for all applications $f(a)$ the argument a has a sort that is an argument sort of the function f. We specify this by the sort judgment $\Gamma \vdash^s_\Sigma \mathbf{A}::\mathbb{A}$ ($\mathbf{A}$ has sort $\mathbb{A}$ under the sort assumptions for the variables in $\mathbf{A}$ given in the variable context Γ). The subsort relation and the property of being well-sorted are given by the judgments $\vdash^s_\Sigma \mathbb{A} \leq \mathbb{B}$ and $\Gamma \vdash^s_\Sigma \mathbf{A}::\mathbb{A}$, which are proven by the subsorting and well-sortedness sub-calculi of $ND(\mathbb{S}\mathrm{FOL})$. We will use $\vdash^s_\Sigma$ for the propositional part of $ND(\mathbb{S}\mathrm{FOL})$.

The logical morphism $\mathcal{R}_\mathcal{S}$ from $\mathbb{S}$FOL to FOL interprets the sorts in $\mathcal{S}$ as unary predicates (parameters of type $\iota \to o$) in FOL. Note that with the definitions in

$\mathbb{S}$FOL	$= (\mathcal{L}^{\mathbb{S}\mathrm{FOL}}, ND(\mathbb{S}\mathrm{FOL})) = \mathrm{FOL}+$		
Signature Σ	$\mathbb{A}, \mathbb{B}, \ldots$	$\iota \to o$	Sorts $\widehat{=}$ Subsets of individuals
	$[\mathbb{A} \leq \mathbb{B}]$		Subsort declarations
	$[c::\mathbb{A}]$		Constant declarations
	Π^*	$(\iota \to o)^2 \to o$	Sorted Universal Quantification
$\mathcal{L}^{\mathbb{S}\mathrm{FOL}}$	Well-formed = well-typed		
Subsorting	$\dfrac{[\mathbb{A} \leq \mathbb{B}] \in \Sigma}{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{B}}$	$\dfrac{}{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{A}}$	$\dfrac{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{B} \quad \vdash^s_\Sigma \mathbb{B} \leq \mathbb{C}}{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{C}}$
Well-sorted	$\dfrac{[c::\mathbb{A}] \in \Sigma}{\Gamma \vdash^s_\Sigma c::\mathbb{A}}$	$\dfrac{}{\Gamma, [X::\mathbb{A}] \vdash^s_\Sigma X::\mathbb{A}}$	$\dfrac{\Gamma \vdash^s_\Sigma \mathbf{A}::\mathbb{B} \to \mathbb{C} \quad \Gamma \vdash^s_\Sigma \mathbf{B}::\mathbb{B}}{\Gamma \vdash^s_\Sigma \mathbf{AB}::\mathbb{C}}$
$ND(\mathbb{S}\mathrm{FOL})$	$\ldots$	$\dfrac{\Gamma \Vdash^\iota_\Sigma \Pi^*\mathbb{A}\mathbf{B} \quad \Gamma \Vdash^\iota_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \Vdash^\iota_\Sigma \mathbf{BA}}$	$\dfrac{\Gamma, [X::\mathbb{A}] \Vdash^\iota_\Sigma \mathbf{A}X}{\Gamma \Vdash^\iota_\Sigma \Pi^*\mathbb{A}\mathbf{A}}$

Figure 9: Sorted First-Order Logic

Fig. 4.1, the universal $\forall X_{\mathbb{A}}.\mathbf{A}$ in $\mathcal{L}^{\mathbb{S}\mathrm{FOL}}(\Sigma)$ is relativized to the $\mathrm{FOL}(\overline{\Sigma})$-formula $\forall X.\mathbb{A}(X) \Rightarrow \mathbf{A}$ (if $\mathbb{A}$ is a base sort). This is just the well-known relativization morphism for sorted first-order logics. Function sorts are relativized into first-order assertions about the domains and ranges of functions. The second part of Fig. 4.1 defines the **signature axioms** generated by a declaration in a sorted signature Σ. We will denote the set of all signature axioms by $\mathcal{R}_{\mathcal{S}}(\Sigma)$. Similarly, we can define the set $\mathcal{R}_{\mathcal{S}}(\Gamma)$ of **sort assumptions** generated by a sorted context Γ by setting $\mathcal{R}_{\mathcal{S}}([X::\mathbb{A}]) := \mathcal{R}_{\mathcal{S}}(\mathbb{A})(X)$ for a declaration $[X::\mathbb{A}]$, we will use $\mathcal{R}_{\mathcal{S}}(\Gamma, \Sigma)$ for $\mathcal{R}_{\mathcal{S}}(\Gamma) \cup \mathcal{R}_{\mathcal{S}}(\Sigma)$.

$\mathcal{R}^l_{\mathcal{S}}: \mathcal{L}^{\mathbb{S}\mathrm{FOL}} \longrightarrow \mathcal{L}^{\mathrm{FOL}}$	
Signature Σ	$\mathcal{R}_{\mathcal{S}}$-image
$a, f, g, \ldots$	$a, f, g, \ldots \in \overline{\Sigma}$
$\mathbb{A}, \mathbb{B}, \ldots$	$\mathcal{P}_{\mathbb{A}}, \mathcal{P}_{\mathbb{B}} \ldots \in \overline{\Sigma}$
$\mathbb{A} \to \mathbb{B}$	$\lambda F_{\iota\to\iota}.(\forall X.\mathcal{R}_{\mathcal{S}}(\mathbb{A})(X) \Rightarrow \mathcal{R}_{\mathcal{S}}(\mathbb{B})(FX))$
Π^*	$\lambda S_{\iota\to o}.\lambda T_{\iota\to o}.\forall X_\iota.SX \Rightarrow TX$
Signature Σ	Signature Axioms $\mathcal{R}_{\mathcal{S}}(\Sigma)$
$[\mathbb{A} \leq \mathbb{B}]$	$\mathcal{P}_{\mathbb{A}} \subseteq \mathcal{P}_{\mathbb{B}}$
$[c::\mathbb{A}]$	$\mathcal{R}^l_{\mathcal{S}}(\mathbb{A})(c)$

Figure 10: Formula Relativization from $\mathbb{S}$FOL to FOL: $\mathcal{R}^l_{\mathcal{S}}$

Their significance is that they encode all the information of the sorted signature in first-order logic, so that we have the following theorem:

Theorem 4.1 (Sort Relativization Theorem):
If $\Gamma \vdash^{\iota}_{\Sigma} \mathbf{A}$, *then* $\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathcal{R}_{\mathcal{S}}(\mathbf{A})$

The proof is a direct consequence of the definition of $\mathcal{R}^{l}_{\mathcal{S}}$, defined in Fig. 4.1: Let $\mathcal{D}\colon \Gamma \vdash^{s}_{\Sigma} \mathbf{A}$, then $\mathcal{D}_{\mathcal{S}}sorts^{ND(\mathbb{S}\mathrm{FOL})}\colon \mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathcal{R}_{\mathcal{S}}(\mathbf{A})$, since $\mathcal{R}^{l}_{\mathcal{S}}$ is a calculus morphism from $\mathbb{S}$FOL to FOL.

All the discussion so far has been purely syntactic, we will come to semantic questions in the next section.

$\dfrac{[\mathbb{A} \leq \mathbb{B}] \in \Sigma}{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{B}}$	$\dfrac{}{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{A}} \subseteq \mathcal{P}_{\mathbb{B}}}$ since $(\mathcal{P}_{\mathbb{A}} \subseteq \mathcal{P}_{\mathbb{B}}) \in \mathcal{T}_{\Sigma}$
$\dfrac{}{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{A}}$	$\dfrac{\dfrac{\dfrac{}{\mathcal{P}_{\mathbb{A}}X \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{A}}X}}{\vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{A}}X \Rightarrow \mathcal{P}_{\mathbb{A}}X}}{\vdash^{\iota}_{\Sigma} \forall X.\mathcal{P}_{\mathbb{A}}X \Rightarrow \mathcal{P}_{\mathbb{A}}X}$
$\dfrac{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{B} \quad \vdash^{s}_{\Sigma} \mathbb{B} \leq \mathbb{C}}{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{C}}$	transitivity of $\Rightarrow$
$\dfrac{[c::\mathbb{A}] \in \Sigma}{\Gamma \vdash^{s}_{\Sigma} c::\mathbb{A}}$	$\dfrac{}{\mathcal{R}_{\mathcal{S}}(\Sigma) \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{A}}c}$ since $\mathcal{P}_{\mathbb{A}}c \in \mathcal{R}_{\mathcal{S}}(\Sigma)$
$\dfrac{}{\Gamma, [X::\mathbb{A}] \vdash^{s}_{\Sigma} X::\mathbb{A}}$	$\dfrac{}{\mathcal{R}(\Gamma), \mathcal{P}_{\mathbb{A}}(X) \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{A}}X}$
$\dfrac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A}::\mathbb{B} \to \mathbb{C} \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{B}::\mathbb{B}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{AB}::\mathbb{C}}$	$\dfrac{\dfrac{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \forall X.\mathcal{P}_{\mathbb{B}}X \Rightarrow \mathcal{P}_{\mathbb{C}}(\mathcal{R}_{\mathcal{S}}(\mathbf{A}X))}{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{B}}(\mathcal{R}_{\mathcal{S}}(\mathbf{B})) \Rightarrow \mathcal{P}_{\mathbb{C}}(\mathcal{R}_{\mathcal{S}}(\mathbf{AB}))} \quad \mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{B}}(\mathcal{R}_{\mathcal{S}}(\mathbf{B}))}{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathcal{P}_{\mathbb{C}}(\mathcal{R}_{\mathcal{S}}(\mathbf{AB}))}$
$\dfrac{\Gamma \vdash^{s}_{\Sigma} \Pi^{*}\mathbb{A}\mathbf{B} \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{A}::\mathbb{A}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{BA}}$	$\dfrac{\dfrac{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \Pi(\lambda X.(\mathbb{A}X) \Rightarrow (\mathbf{B}X))}{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathbb{A}\mathbf{A} \Rightarrow \mathbf{BA}} \quad \mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathbb{A}\mathbf{A}}{\mathbf{BA}}$
$\dfrac{\Gamma, [X::\mathbb{A}] \vdash^{s}_{\Sigma} \mathbf{A}X}{\Gamma \vdash^{s}_{\Sigma} \Pi^{*}\mathbb{A}\mathbf{A}}$	$\dfrac{\dfrac{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma), \mathcal{P}_{\mathbb{A}}X \vdash^{\iota}_{\Sigma} \mathbf{A}X}{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \mathbb{A}X \Rightarrow \mathbf{A}X}}{\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \vdash^{\iota}_{\Sigma} \Pi(\lambda X.\mathbb{A}X \Rightarrow \mathbf{A}X)}$

Figure 11: Proof Relativization from $\mathbb{S}$FOL to FOL: $\mathcal{R}^{c}_{\mathcal{S}}$.

4.2. Logical Morphisms

The fundamental logical notions for relativizations like the ones discussed in the last section are logical systems and logic morphisms. For the purposes of this article, we will call a pair $\mathcal{S} = (\mathcal{L}, \mathcal{C})$ a **logical system**, if $\mathcal{L}$ is a **logical language** (set of well-formed formulae) and $\mathcal{C}$ is a **calculus** i.e. a set of inference rules defined in usual way as n-ary relations over well-formed formulae; formally $\mathcal{C} \subseteq \mathcal{L}^* = \bigcup_{i \in \mathbf{N}} \mathcal{L}^i$.

Given a logical system $\mathcal{S} = (\mathcal{L}, \mathcal{C})$, we define an $\mathcal{S}$-**derivation** $\mathcal{D}$ of an **assertion A** from a set $\mathcal{H}$ of **hypothesess** (written $\mathcal{D}: \mathcal{H} \vdash_{\mathcal{S}} \mathbf{A}$) as a tree $\mathcal{D}$ (or a directed acyclic graph), where the leaves of $\mathcal{D}$ are labeled with the formulae from $\mathcal{H}$ and the root is labeled with $\mathbf{A}$. Furthermore, all nodes of $\mathcal{D}$ are labeled by assertions $\mathbf{C}$ and inference rules $\mathcal{R} \in \mathcal{C}$, such that the for the labels $\mathbf{H}_1, \ldots, \mathbf{H}_n$ of the daughters of a node we have $\mathcal{R}(\mathbf{H}_1, \ldots, \mathbf{H}_n, \mathbf{C})$. Thus a calculus $\mathcal{C}$ defines a relation $\vdash_{\mathcal{S}}$ (of variable arity) on $\mathcal{L}$, which we will call the **derivation relation** of $\mathcal{S}$. We will use the terms like $\mathcal{S}$-**proof** (for a derivation of an assertion $\mathbf{A}$ from the empty set of hypotheses) and $\mathcal{S}$-**theorem** (for an assertion for which there is a $\mathcal{S}$-proof) in the usual way.

We say that a logical system $\mathcal{S} = (\mathcal{L}, \mathcal{C})$ is a **subsystem** of $\mathcal{S}' = (\mathcal{L}', \mathcal{C}')$, iff $\mathcal{L} \subseteq \mathcal{L}'$ and $\vdash_{\mathcal{S}} \subseteq \vdash_{\mathcal{S}'}$. We call $\mathcal{S}$ **equivalent** to $\mathcal{S}'$, iff $\mathcal{L} = \mathcal{L}'$ and $\vdash_{\mathcal{S}} = \vdash_{\mathcal{S}'}$, or equivalently, if they are subsystems of each other.

A calculus comes with a natural notion of composition of derivations: If $\mathcal{D}: \mathcal{H}, \mathbf{A} \vdash_{\mathcal{C}} \mathbf{B}$, and $\mathcal{E}: \mathcal{K} \vdash_{\mathcal{C}} \mathbf{A}$, then we obtain a $\mathcal{C}$-derivation from $\mathcal{D}$ and $\mathcal{E}$ (we denote it with $\mathcal{D} \oplus_{\mathbf{A}} \mathcal{E}$) by attaching $\mathcal{E}$ at the leaf $\mathbf{A}$ of $\mathcal{D}$; we have $\mathcal{D} \oplus_{\mathbf{A}} \mathcal{E}: \mathcal{H}, \mathcal{K} \vdash_{\mathcal{C}} \mathbf{B}$. Note that any calculus $\mathcal{C}$ can be augmented with combinations of the inference rules without changing the derivability relation (the logical systems are equivalent, which really interests us for our applications). We will therefore assume that calculi are minimal in the following sense: If $\mathcal{D}, \mathcal{E} \in \mathcal{C}$, then $\mathcal{D} \oplus_{\mathbf{A}} \mathcal{D} \notin \mathcal{C}$.

Let $\mathcal{S} = (\mathcal{L}, \mathcal{C})$ and $\mathcal{S}' = (\mathcal{L}', \mathcal{C}')$ be logical systems and $\mathsf{f}: \mathcal{L} \longrightarrow \mathcal{L}'$ a total function, then we call a total function g that maps $\mathcal{S}$-derivations to $\mathcal{S}'$-derivations a **calculus morphism** with respect to f, iff for any $\mathcal{S}$-derivation $\mathcal{D}: \mathcal{H} \vdash_{\mathcal{S}} \mathbf{A}$, we have $\mathsf{g}(\mathcal{D}): \mathsf{f}(\mathcal{H}) \vdash_{\mathcal{S}'} \mathsf{f}(\mathbf{A})$. A **logic morphism** $\mathcal{F}: \mathcal{S} \longrightarrow \mathcal{S}'$, is a pair $(\mathcal{F}^l, \mathcal{F}^c)$ of mappings, such that $\mathcal{F}$ is a calculus morphism with respect to $\mathcal{F}^l$. We call $\mathcal{F}$ a **logic homomorphism**, iff $\mathcal{F}^c(\mathcal{E} \oplus_{\mathbf{A}} \mathcal{D}) = \mathcal{F}^c(\mathcal{E}) \oplus_{\mathcal{F}^l(\mathbf{A})} \mathcal{F}^c(\mathcal{D})$. Note that a logic homomorphism is determined by its behavior on $\mathcal{C}$.

In analogy to the Sort relativization theorem (4.1), we have to following meta-theorem.

Theorem 4.2 (General Relativization Theorem): *If $\mathcal{S} = (\mathcal{L}, \mathcal{C})$ and $\mathcal{S}' = (\mathcal{L}', \mathcal{C}')$ are logical systems, $\mathcal{R}: \mathcal{S} \longrightarrow \mathcal{S}'$ is a logic morphism, and $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, then $\mathcal{R}(\Sigma), \mathcal{R}(\mathcal{H}) \vdash_{\mathcal{C}'} \mathcal{R}(\mathbf{A})$.*

The existence of such theorems is the guiding intuition behind our setup of the landscape of representation languages in Fig. 7. Any theorem that is provable in

a higher representation language will be provable (and indeed the proof can be constructed by relativization) in the basic logics. Let us now investigate how we can build logical morphisms.

Let $\mathcal{S} = (\mathcal{L}, \mathcal{C})$ be a logical system and $\mathsf{f}: \mathcal{L}' \longrightarrow \mathcal{L}$ a total function, then f induces a calculus $\mathcal{C}^{\mathsf{f}}$ on $\mathcal{L}'$ by setting $\mathcal{H} \vdash_{\mathcal{C}^{\mathsf{f}}} \mathbf{A}$, iff $\mathsf{f}(\mathcal{H}) \vdash_{\mathcal{C}} \mathsf{f}(\mathbf{A})$. We call $\mathcal{S}^{\mathsf{f}} := (\mathcal{L}', \mathcal{C}_{\mathsf{f}})$ the logical system induced by f. Moreover, f induces a logical homomorphism $\mathcal{F}^{\mathsf{f}} =: \mathcal{S}^{\mathsf{f}} \to \mathcal{S}$, in the obvious way ($\mathcal{F}^{\mathsf{f}} = (\mathsf{f}, \mathsf{g})$, where g is the homomorphism on derivations induced by the translation f). Note that a function $\mathcal{F} := (\mathcal{F}^l, \mathcal{F}^c): \mathcal{S} \longrightarrow \mathcal{S}' = (\mathcal{L}', \mathcal{C}')$ is a logical morphism, iff $\vdash_{\mathcal{S}^{\mathsf{f}}} \subseteq \vdash_{\mathcal{S}'}$, or in other words, $\mathcal{S}^{\mathsf{f}}$ is a subsystem of $\mathcal{S}'$.

We will call a set $\mathcal{H}$ of logical systems together with a set of logical morphisms a **logical hierarchy**, if the set of logical morphisms is closed under composition (note that the composition of two logical morphisms is again one). The formal notions introduced so far are sufficient to introduce a methodology of maintaining logical hierarchies. We can start out with a logical system, (say FOL as in the last section), and introduce another logical system by inducing it from formula mapping. $ND(\mathbb{S}\text{FOL})$ is induced by $\mathcal{R}^l_{\mathcal{S}}$ in the following way: let us consider the case of universal instantiation we need an $ND(\mathbb{S}\text{FOL})$ rule that proves $\forall X_{\mathbb{A}}.\mathbf{A}$, so we look for a $ND(\text{FOL})$ proof of $\forall X.\mathcal{P}_{\mathbb{A}}X \Rightarrow \mathbf{A}$, we identify the smallest subtree, such that all of the leaves are in $\mathbf{Im}(\mathcal{R}^l_{\mathcal{S}})$, and arrive at the last but one in Fig. 4.1. If we proceed similarly with the other inference rules, we arrive at $ND(\mathbb{S}\text{FOL})$.

So we can see that we can introduce a new logical system with a logic morphism into a hierarchy by specifying the language (morphism) and inducing the calculus (this situation is similar to the case of import morphisms in the theory hierarchy in section 2.4). If we want to introduce a new logic morphism between existing logical systems, we have to be more careful, since the calculus in the source system is already fixed. In order to prove that the defining pair $\mathcal{F} = (\mathcal{F}^l, \mathcal{F}^c)$ of mappings is really a logic morphism we have to check that logical system induced by $\mathcal{F}$ is a subsystem of the original target system. Note that we even have to check these conditions for logical endomorphisms (logical morphisms from a logical system to itself), since we always have the identity morphism, to which a new logical morphism has to be compatible. Thus the case of adding a new (redundant) logical morphism to a hierarchy is similar to the case of the theory-inclusions discussed in section 2.4. We expect that we can develop a calculus for the "management of logical hierarchies" based on the Dieter Hutter's ideas for theory hierarchies, but we leave that to further work.

Let us now see how the ideas of linking logics relates to semantics. In this example, we take the semantics of first-order logic as given. It is just the classical Tarski-style semantics: A model is a pair $(\mathcal{D}_\iota, \mathcal{I})$, where $\mathcal{D}_\iota$ is an arbitrary set of individuals and $\mathcal{I}$ is a function that maps individual constants in Σ to members of $\mathcal{D}_\iota$, functions in Σ to functions/relations on $\mathcal{D}_\iota$ (of appropriate arities). Variables are evaluated by a variable assignment φ, so that the value function $\mathcal{I}_\varphi$ is just the homomorphism determined by $\mathcal{I}$ and φ. Note that this semantics is absolutely

consistent with our choice to take the simply typed λ-calculus as a meta-logic: the choice of the universe $\mathcal{D}_\iota$ determines the standard model $\mathcal{D} = \{\mathcal{D}_\alpha : \alpha \in \mathcal{T}\}$ if we take $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$. n-ary functions are then objects of type $\iota \to \ldots \to \iota \to \iota$, and predicates of type $\iota \to \ldots \to \iota \to o$. The semantics of universal quantification is regained by setting $\mathcal{I}(\Pi^\iota)$ to be the predicate that evaluates to $\mathtt{T}$, iff its argument is $\mathtt{T}$ on all inputs: $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathcal{I}_\varphi(\Pi^\alpha(\lambda X.\mathbf{A})) = \mathcal{I}(\Pi^\alpha)(\mathcal{I}_\varphi(\lambda X.\mathbf{A})) = \mathtt{T}$, iff $\mathcal{I}_\varphi(\lambda X.\mathbf{A})\mathsf{a} = \mathtt{T}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$. This is the case, iff $\mathcal{I}_{\varphi,[\mathsf{a}/X]}\mathbf{A} = \mathtt{T}$, by definition of the value of λ-terms in the simply typed λ-calculus.

The semantics of $\mathbb{S}$FOL is similar and well-known from the literature. Let us for the moment forget this and see whether we can *define* the semantics of the logical system $\mathbb{S}$FOL by $\mathcal{R}_\mathcal{S}$. For this we intuitively work the relativization mapping backwards.

We start out with the sorts. These are members of the signature, so they should be reflected directly in the structure of the model. Since they are relativized to unary predicates, a sort $\mathbb{A}$ must correspond to a subset $\mathcal{D}_\mathbb{A} = \{\mathsf{a} \in \mathcal{D}_\iota \mid \mathcal{I}(\mathbb{A}) = \mathtt{T}\} \subseteq \mathcal{D}_\iota$ of the universe $\mathcal{D}_\iota$. Now, the signature axioms tell us that if $[\mathbb{A} \leq \mathbb{B}] \in \Sigma$, then $\mathcal{D}_\mathbb{A} \subseteq \mathcal{D}_\mathbb{B}$ and if $[c::\mathbb{A}] \in \Sigma$, then $\mathcal{I}(c) \in \mathcal{D}_\mathbb{A}$. In particular, the signature axiom for functional sorts insists on the right input-output behavior of functions. For a variable context Γ, the context assumptions specify that the context is well-sorted.

Note that this is a (a posteriori) verification of the semantics of sorted logics from the literature. Also note that this account does not entail the fact that sorts are non-empty (a fact that is often assumed in sorted logics). We only know this if there is a constant declaration for each base sort in the signature.

We will say that the semantics we have constructed by looking at the relativization was **induced** by $\mathcal{R}_\mathcal{S}$ from the the semantics of FOL. Now, the relativization theorem gives us a conservative extension result: If $ND(\text{FOL})$ is sound for first-order semantics, then $ND(\mathbb{S}\text{FOL})$ is for the induced semantics. Furthermore, the logical system $\mathbb{S}$FOL is not more expressive than FOL.

In the special case of $\mathcal{R}:\mathbb{S}\text{FOL} \longrightarrow \text{FOL}$ we also have the converse result, ($\mathbb{S}$FOL and FOL are equally expressive), since there is a partial inverse $\mathcal{R}_{\mathbb{T}\mathsf{op}}$ to $\mathcal{R}_\mathcal{S}$ ($\mathcal{R}_\mathcal{S} \circ \mathcal{R}_{\mathbb{T}\mathsf{op}} = \text{Id}_\text{FOL}$), which embeds FOL as a **fragment** into $\mathbb{S}$FOL. $\mathcal{R}_{\mathbb{T}\mathsf{op}}(\Sigma)$ contains only one (base) sort $\mathbb{T}\mathsf{op}_\iota$ and one declaration $[c::\mathbb{T}\mathsf{op}_\alpha]$, for each constant of type α in Σ (here we use the convention that $\mathbb{T}\mathsf{op}_{\alpha\to\alpha} = \mathbb{T}\mathsf{op}_\alpha \to \mathbb{T}\mathsf{op}_\beta$). The language and calculus morphism are the identity. Clearly, the semantics induced from the semantics of $\mathbb{S}$FOL by $\mathcal{R}_{\mathbb{T}\mathsf{op}}$ is again the semantics of FOL.

4.3. Relativizing Type Theory into Set Theory

The goal of the next section will be to construct a hierarchy of representation languages culminating in a high-level logical system $\mathcal{MV}$ (see section 5) for formalizing mathematics. $\mathcal{BMV}$ is a joint generalization of Ohori's record λ-calculus (47) and the sorted λ-calculus from (41). $\mathcal{MV}$ extends $\mathcal{BMV}$ by specialized sort machinery to formalize mathematical structures like groups. Before

we undertake that, let us briefly complete the discussion of the lower half of Fig. 7.

The method of relativizations can be used to build up the simply typed λ-calculus ($\lambda^{\to}$) from axiomatic set theories like ZF (25), and we will spend the rest of this section exploring this possibility to ground the hierarchy of representation languages in set theories. Since the logical side of this is rather standard and well-understood (see e.g. (21)) and has been formalized in several deduction systems, e.g.in Otter (49) or Isabelle (48), we will only briefly sketch the process.

Axiomatic set theories like ZF only come with a basic type γ of "set" and with the logical relation constant $\in$ for element-hood. The axiomatic method is used to restrict set comprehension to get around paradoxical sets like Russell's set of all sets that do not contain themselves: the theories contain specific axioms for set comprehension; for instance there is an axiom stating that for any sets $\mathbb{A}$ and $\mathbb{B}$, the Cartesian product $\mathbb{A} \times \mathbb{B}$ is again a set. (Partial) functions are construed as univocal relations (a relation $\mathcal{F} \subset \mathbb{A} \times \mathbb{B}$ is a function, iff for all $(x, y), (x, z) \in \mathcal{F}$ we have $y = z$) and function application is represented as projection to the second argument ($\mathsf{f}[\mathsf{a}]$ is the (unique) b, such that $(\mathsf{a}, \mathsf{b}) \in \mathsf{f}$.)

We start out by relativizing the simply typed λ-calculus to **typed set theory** $\mathcal{T}$Set, i.e. a simply typed higher-order predicate logic HOL together with a formulation of the ZF axioms, interpreting sets as predicates and element-hood as predication (i.e. $\mathbf{A} \in \mathbf{S}$ stands for $\mathbf{S}(\mathbf{A})$). HOL is a variant of Andrews' system $\mathcal{Q}$ with comprehension axioms instead of β-conversion; the types make $\mathcal{T}$Set consistent (see (3) for a deduction-related introduction of higher-order logic and the simply typed model theory). Using the techniques from (21; 48), we use the selection axiom from ZF to construct a Λ-operator, i.e. a $\mathcal{T}$Set-formula that behaves like the λ-abstraction operator. Thus we can construct a language morphism from the simply typed λ-calculus to $\mathcal{T}$Set by mapping λ-abstractions in $\lambda^{\to}$ to HOL-formulae using Λ. The calculus morphism is constructed by mapping the β-axiom scheme of $\lambda^{\to}$ to the proof of the validity β in $\mathcal{T}$Set.

The next step is to relativize $\mathcal{T}$Set (higher-order logic) to sorted first-order logic. For this, we can either use a technique developed by Manfred Kerber (38) or we can directly use the definition of functions as univocal relations in ZF to build a logic morphism from $\mathcal{T}$Set to sorted first-order logic. Finally, the techniques detailed in section 4.1 get us to classical ZF. Note that we have to take care to relativize the ZF axioms in the source system to a form in which they are equivalent to the ZF axioms native to the target system.

4.4. Evaluation

The logic morphisms presented in this section can always be used to transform any proof in the source system into one of the target system (this is the reason for the definition of logic morphism used in this article), in other words, from a purely theoretical point of view, the expressive type-theoretic representation formalisms in the $\mathcal{MV}$ hierarchy can be viewed as being only syntactic sugar

to enhance legibility. However, from a practical point of view, the expressive formalisms allow for more efficient inference systems that allow the knowledge base system to give added value services, that would be impractical on the level of set theory.

We believe that while axiomatic set theories address foundational issues of formalizing mathematics – in the old *representational tradition* of classical logic, where there is a quest for the minimal logical system that is expressive enough to encode all relevant problems – logical systems like the simply typed λ-calculus are more adequate to address *computational* needs of doing mathematics.

Orthogonal to the debate about set theory vs. type theory, there is a discussion, whether or not formalized mathematics should be constructive or not. We do not make any assertion about this, but note, that it is simple to extend the hierarchy of representation languages by providing a logical morphism to intuitionistic set theory that basically introduces oracles for the law of the excluded middle; see e.g. (33). In Fig. 7, we have marked the intuitionistic logical systems with an i and the oracle-morphisms with dotted lines.

In the next section, we will continue to develop higher-level representation formalisms for mathematics by the logical morphism method discussed in this section.

5. Mathematical Vernacular

In this section, we develop the basic concepts for a representation language $\mathcal{MV}$ for formalizing and reasoning about mathematics in MBase. Such a logic must be flexible, easy to use, and last but not least, it must support the rich, structured inference machinery mathematicians have at their disposal. In short, it should be modeled after the natural language of everyday mathematics, that is sometimes called "mathematical vernacular" (This term is taken from N.G. de Bruin in (19), where he proposes a different logical system with similar intentions).

In contrast to other authors, we contend that this language *can* be modeled in a formal language, and that the system $\mathcal{MV}$ is a good first approximation. We will develop the syntax and operational semantics of $\mathcal{MV}$, and show that it can be grounded in simpler logical systems (and ultimately in axiomatic set theory) by the technique of logical morphisms developed in section 4. This also gives us a way of relativizing all inference mechanisms, such as sort computation, sorted higher-order matching and unification into less expressive logics, where they can (if wanted) be verified.

Note that the relativizations give us a form of set-theoretic semantics (by mapping formulae to set theory), which can be shown to be equivalent to the standard Tarski-style semantics for $\lambda^{\rightarrow}$ (see (3) for typed Henkin models and (41) for a sorted version).

To get a better intuition about the language, we will develop $\mathcal{MV}$ in three steps. To introduce the basic setup of the language we start out with a language $\mathcal{BMV}$, which extends the simply typed λ-calculus by sorts and records in

section 5.1. Then we successively enhance the practical expressive power of the language by introducing label-selective application and abstractions, and dependent sorts as additional language constructs using semigroups as the motivating example. As we will see in section 5.5, this does not enhance the expressivity in principle, but (as we will see in the mathematical examples in section 5.4) it has practical advantages both for conciseness of representation and in enabling inference procedures.

5.1. $\mathcal{BMV}$ an Expressive Sorted Record-λ-Calculus

$\mathcal{T}$	$::= \iota \mid o \mid \mathcal{T} \to \mathcal{T}' \mid \{\!\{\ell^1\!:\!\mathcal{T}^1, \ldots, \ell^n\!:\!\mathcal{T}^n\}\!\}$	(Types: $\alpha, \beta, \ldots$)
$\mathcal{S}$	$::= \mathbb{T}\mathsf{op}_{\mathcal{T}} \mid \mathcal{S} \to \mathcal{S}' \mid \{\!\{\ell_1 :: \mathcal{S}^1, \ldots, \ell_n :: \mathcal{S}^n\}\!\} \mid \mathcal{S} \sqcap \mathcal{S}'$	(Sorts $\mathbb{A}, \mathbb{B}, \ldots$)
$\mathbf{M}$	$::= X \mid c \mid (\mathbf{M}\mathbf{N}) \mid \lambda X.\mathbf{M}$	(Terms $\mathbf{A}, \mathbf{B}, \ldots$)
	$\mid \{\!\{\ell^1 = \mathbf{M}_1, \ldots, \ell^n = \mathbf{M}_n\}\!\} \mid \mathbf{M}.\ell$	
	variables : X, Y, Z; constants : $c, \Pi^\alpha, \wedge, \neg$	
Σ	$::= \emptyset \mid \Sigma, [\mathbf{M} :: \mathcal{S}] \mid \Sigma, [\mathcal{S} \rhd \mathcal{T}] \mid \Sigma, [\mathcal{S} \leq \mathcal{S}']$	(Signature)
Γ	$::= \emptyset \mid \Gamma, [X :: \mathcal{S}]$	(Environment)

Figure 12: Syntax of $\mathcal{BMV}$

$\mathcal{BMV}$ is a sorted record-λ-calculus (see Fig. 5.1), i.e. an extension of the simply typed λ-calculus by records. We will use the type o for the truth values and the type ι for individuals. As a consequence terms and formulae can be distinguished by their type: the equivalents of (first-order) formulae are λ-terms of type o, whereas terms are λ-terms of type ι. We will call a type a **record type**, iff it is of the form $\{\!\{\ell_1\!:\alpha_1, \ldots, \ell_n\!:\alpha_n\}\!\}$, and we will use the standard record **selection** operator "." with the assumption that it is only applied to record types.

Furthermore, the type system is augmented with a typed sort system, that can be used to specify domains and ranges of functions and thus enables the system to compute most of the definedness preconditions that are ubiquitous in mathematics fully autonomously. From an abstract point of view, sorts enable us to constrain the set of models and restrict the inference procedures to this set of models. It is important for the soundness of the system that sorts are also typed (see Fig. 15 for an inference system that computes the type of a given sort).

The set of judgments (see Fig. 5.1) that are needed for the formal development of the calculus comprises the typing judgments for terms $\Gamma \vdash^{\iota}_{\Sigma} \mathbf{A}\!:\alpha$ and sorts $\vdash^{s}_{\Sigma} \mathbb{A} \rhd \alpha$, the sorting judgment ($\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A}$).

All of these judgments are relative to a set of global type/sort assumptions in the **signature** Σ and the judgments for terms (sorts do not contain variables) are also relative to a set of (local) type and sort assumptions Γ (the **context**) for the variables.

$\Gamma \vdash^{s}_{\Sigma} \mathbf{M}$	$\mathbf{M}$ is provable from assumptions Γ
$\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{B}$	Sort $\mathbb{A} \in \mathcal{S}$ is a subsort of $\mathbb{B} \in \mathcal{S}$
$\vdash^{\tau}_{\Sigma} \mathbb{A} :> \alpha$	Sort $\mathbb{A} \in \mathcal{S}$ has type $\alpha \in \mathcal{T}$ (at most one per sort $\mathbb{A}$)
$\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A}$	Term $\mathbf{A}$ has sort $\mathbb{A}$ assuming Γ and Σ
$\Gamma \vdash^{\tau}_{\Sigma} \mathbf{A} : \alpha$	Term $\mathbf{A}$ has type $\alpha \in \mathcal{T}_0$ assuming Γ and Σ

Figure 13: Judgments

$$\frac{[\mathbf{A} :: \mathbb{A}] \in \Sigma}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A}} \qquad \frac{[X :: \mathbb{A}] \in \Gamma}{\Gamma \vdash^{s}_{\Sigma} X :: \mathbb{A}} \qquad \frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} : \alpha}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{T}\mathsf{op}_{\alpha}}$$

$$\frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{C} \to \mathbb{A} \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{C} :: \mathbb{C}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A}\mathbf{C} :: \mathbb{A}} \qquad \frac{\Gamma, X :: \mathbb{B} \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A} \quad \vdash^{s}_{\Sigma} \mathbb{B} :> \beta}{\Gamma \vdash^{s}_{\Sigma} \lambda X_{\beta} . \mathbf{A} :: \mathbb{B} \to \mathbb{A}}$$

$$\frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A} \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{A} =_{\beta\eta} \mathbf{B}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{B} :: \mathbb{A}}$$

$$\frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A}.\ell :: \mathbb{A}.\ell} \qquad \frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A}_1 :: \mathbb{A}_1 \quad \ldots \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{A}_n :: \mathbb{A}_n}{\Gamma \vdash^{s}_{\Sigma} \{\!\{\ell_1 = \mathbf{A}_1, \ldots, \ell_n = \mathbf{A}_n\}\!\} :: \{\!\{\ell_1 :: \mathbb{A}_1, \ldots, \ell_n :: \mathbb{A}_n\}\!\}}$$

$$\frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A} \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{B}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A} \sqcap \mathbb{B}} \qquad \frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A} \sqcap \mathbb{B}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A}} \qquad \frac{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A} \sqcap \mathbb{B}}{\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{B}}$$

Figure 14: Well-sorted terms in $\mathcal{BMV}$

The most important judgment for well-formedness of $\mathcal{MV}$ expressions is the **term sorting** judgment (see Fig. 14), which classifies terms by their sorts. The first set of rules comes from the ordinary sorted λ-calculus (see (41) for an introduction), the second is an obvious adaptation of Ohori's rules for record typing (47), and the third set of rules is that for intersection sorts from (44). The most important rule in the sorted calculus is the first one in Fig. 14, the term declaration rule. In contrast to other systems it allows to declare and use sort information for term schemata like $[X_{\mathbb{R}} * X :: \mathbb{P}]$ (doubling a real number produces an positive real), $[(\lambda X . X), (\lambda X . Y_{\mathbb{R}}) :: \mathbb{M}]$ (the identity and the constant function are monomials), and even $[(\lambda F, G, X . FX * GX) :: \mathbb{M}^2 \to \mathbb{M}]$ (the set of monoids is closed under pointwise multiplication). Note that the latter give a full theory of monoid functions on the reals. The term typing judgment – which guarantees consistency and termination of $\beta\eta$-reduction – is defined in terms of

$$\frac{[\mathbb{A}\!:\!\alpha] \in \Sigma}{\vdash^{\iota}_{\Sigma} \mathbb{A}\!:\!\alpha} \qquad \frac{}{\vdash^{\iota}_{\Sigma} \mathsf{Top}_{\alpha}\!:\!\alpha} \qquad \frac{\vdash^{\iota}_{\Sigma} \mathbb{A}\!:\!\alpha \quad \vdash^{\iota}_{\Sigma} \mathbb{B}\!:\!\beta}{\vdash^{\iota}_{\Sigma} \mathbb{A} \to \mathbb{B}\!:\!\alpha \to \beta}$$

$$\frac{\vdash^{\iota}_{\Sigma} \mathbb{A}_1\!:\!\alpha_1 \quad \ldots \quad \vdash^{\iota}_{\Sigma} \mathbb{A}_n\!:\!\alpha_n}{\vdash^{\iota}_{\Sigma} \{\!\{\ell_1 :: \mathbb{A}_1, \ldots, \ell_n :: \mathbb{A}_n\}\!\}\!:\!\{\!\{\ell_1 : \alpha_1, \ldots, \ell_n : \alpha_n\}\!\}} \qquad \frac{\vdash^{\iota}_{\Sigma} \mathbb{A}\!:\!\{\!\{\ell : \alpha, \ldots\}\!\}}{\vdash^{\iota}_{\Sigma} \mathbb{A}.\ell\!:\!\alpha}$$

Figure 15: Sort Typing

$$\frac{[\mathbb{A} \leq \mathbb{B}] \in \Sigma}{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{B}} \qquad \frac{}{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{A}} \qquad \frac{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{B} \quad \vdash^{s}_{\Sigma} \mathbb{B} \leq \mathbb{C}}{\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{C}}$$

$$\frac{}{\vdash^{s}_{\Sigma} \mathbb{A} \sqcap \mathbb{B} \leq \mathbb{A}} \qquad \frac{}{\vdash^{s}_{\Sigma} \mathbb{A} \sqcap \mathbb{B} \leq \mathbb{B}}$$

$$\frac{\mathbb{A}' \leq \mathbb{A} \quad \mathbb{B} \leq \mathbb{B}'}{\mathbb{A} \to \mathbb{B} \leq \mathbb{A}' \to \mathbb{B}'} \qquad \frac{}{\mathsf{Top}_{\alpha \to \beta} \leq \mathsf{Top}_{\alpha} \to \mathsf{Top}_{\beta}}$$

Figure 16: Subsorting

it: if $\Gamma \vdash^{s}_{\Sigma} \mathbf{A} :: \mathbb{A}$ and $\vdash^{\iota}_{\Sigma} \mathbb{A}\!:\!\alpha$, then $\Gamma \vdash^{\iota}_{\Sigma} \mathbf{A}\!:\!\alpha$. Note that the typed system is just Ohori's record calculus (47), which is a conservative extension of the simply typed λ-calculus.

For such a construction, sorts must also be typed (see Fig. 15 for an inference system for the sort typing judgment). We will see in section 5.5 that this gives us a conservative extension of the simply typed λ-calculus. **Subsorting** is used in the signature to declare an intended subset relation between sorts. We do not have to declare all subsort relations in the signature, since some can be inferred by the inference system in Fig. 16. Note that we do not need a subsort judgment in a system like $\mathcal{BMV}$, since the notion of subsorting is in principle subsumed by the mechanism of term declarations (the rules in Fig. 16 are in fact admissible; see (41) for details). However it is good to include them explicitly in a system like $\mathcal{MV}$, intuitive usability and readability are important. With the methods from (41), we can check that $\mathcal{MV}$ is a well-defined system, e.g. if $\vdash^{s}_{\Sigma} \mathbb{A} \leq \mathbb{B}$, then there is a type $\alpha \in \mathcal{T}$, such that $\vdash^{\iota}_{\Sigma} \mathbb{A}\!:\!\alpha$ and $\vdash^{\iota}_{\Sigma} \mathbb{B}\!:\!\alpha$.

Now, we come to the $\mathcal{BMV}$ calculus for validity: a variant of Gentzen's calculus of natural deduction. We will use alphabetic renaming and permutation

$$\frac{\Gamma, X::\mathbb{B} \vdash^s_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash^s_\Sigma \mathbf{B}::\mathbb{B}}{\Gamma \Vdash^s_\Sigma (\lambda X_{\mathbb{B}}.\mathbf{A})\mathbf{B} \longrightarrow_\beta [\mathbf{B}/X]\mathbf{A}} \qquad \frac{\Gamma \vdash^\iota_\Sigma \mathbf{A}: \alpha \to \beta \quad X \notin \mathbf{Free}(\mathbf{A})}{\Gamma \Vdash^s_\Sigma (\lambda X.\mathbf{A}X) \longrightarrow_\eta \mathbf{A}}$$

$$\frac{}{\Gamma \Vdash^s_\Sigma \{\!\{\ell = \mathbf{A}, \ldots\}\!\}.\ell \longrightarrow_\rho \mathbf{A}} \qquad \frac{}{\Gamma \Vdash^s_\Sigma \{\!\{\ell_1 = \mathbf{A}.\ell_1, \ldots, \ell_n = \mathbf{A}.\ell_n\}\!\} \longrightarrow_\eta \mathbf{A}}$$

Figure 17: Operational Equality for $\mathcal{BMV}$.

$$\frac{\Gamma \Vdash^s_\Sigma \Pi^{\mathbb{A}}\mathbf{B} \quad \Gamma \vdash^s_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \Vdash^s_\Sigma \mathbf{B}\mathbf{A}} \qquad \frac{\Gamma, [X::\mathbb{A}] \Vdash^s_\Sigma \mathbf{A}X}{\Gamma \Vdash^s_\Sigma \Pi^{\mathbb{A}}\mathbf{A}} \qquad \frac{\Gamma \Vdash^s_\Sigma \mathbf{A} =_{\beta\eta\rho} \mathbf{B} \quad \Gamma \Vdash^s_\Sigma \mathbf{A}}{\Gamma \Vdash^s_\Sigma \mathbf{B}}$$

Figure 18: Natural Deduction for $\mathcal{BMV}$.

for records, record types and record sorts without reference. Furthermore, $\mathcal{MV}$ knows sorted variants of $\beta\eta$-reduction like the one in (41) and furthermore ρ-reduction for record constructors (see Fig. 18). Finally, we have the introduction and elimination rules for the sorted quantifier $\Pi^{\mathbb{A}}$. The $\Pi^{\mathbb{A}}$ ($\mathbb{A} \in \mathcal{S}$) are logical constants of sort $(\mathbb{A} \to \mathbb{T}\mathsf{op}_o) \to \mathbb{T}\mathsf{op}_o$ for $\mathcal{BMV}$, we use the usual higher-order abstract syntax, where $\forall X_{\mathbb{A}}.\mathbf{A}$ stands for $\Pi^{\mathbb{A}}(\lambda X.\mathbf{A})$.

5.2. Label-Selective Abstraction and Application

When formalizing larger bodies of mathematics or reusing already existing theories it often becomes problematic to remember argument order of functions. For this, programming languages like Common Lisp – where the situation is similar – have developed the so-called **keyword arguments**, i.e. a variant of function application and abstraction, where the mapping of arguments to formal parameters is not based on argument order, but on identification by so-called keywords. This idea has been formalized by Ait Kaci and Garrigue in the so-called *label-selective λ-calculus* (2), which extends the simply typed λ-calculus by label-selective application and abstraction.

In the following, we will briefly sketch how to extend $\mathcal{MV}$ analogously. Formally, we need an additional type schema: $\alpha \xrightarrow{\ell} \beta$, a corresponding sort schema $\mathbb{A} \xrightarrow{\ell} \mathbb{B}$ and two new term constructors: $[\mathbf{A}@_\ell\mathbf{B}]$ for **labeled application** and $\lambda^\ell X.\mathbf{A}$ for **labeled λ-abstraction**. We will reuse the record labels as selection labels, since they serve a similar purpose (ℓ and k correspond to the keywords

$$\frac{\Gamma \vdash^s_\Sigma \mathbf{A}::\mathbb{C} \xrightarrow{\ell} \mathbb{A} \quad \Gamma \vdash^s_\Sigma \mathbf{C}::\mathbb{C}}{\Gamma \vdash^s_\Sigma [\mathbf{A}@_\ell\mathbf{C}]::\mathbb{A}} \qquad \frac{\Gamma, X::\mathbb{B} \vdash^s_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \vdash^s_\Sigma \lambda^\ell X.\mathbf{A}::\mathbb{B} \xrightarrow{\ell} \mathbb{A}}$$

$$\frac{\Gamma, X::\mathbb{B} \vdash^s_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash^s_\Sigma \mathbf{B}::\mathbb{B}}{[(\lambda^\ell X.\mathbf{A})@_\ell\mathbf{B}] \longrightarrow_\beta [\mathbf{B}/X]\mathbf{A}}$$

$$\frac{X \notin \mathbf{Free}(\mathbf{B})}{[(\lambda^\ell X.\mathbf{A})@_k\mathbf{B}] \longrightarrow_\eta (\lambda^\ell X.[\mathbf{A}@_k\mathbf{B}])} \qquad \frac{\Gamma \vdash^i_\Sigma \mathbf{A}:\alpha \xrightarrow{\ell} \beta \quad X \notin \mathbf{Free}(\mathbf{A})}{(\lambda^\ell X.[\mathbf{A}@_\ell X]) \longrightarrow_\eta \mathbf{A}}$$

$$\frac{}{\mathbb{A}_1 \xrightarrow{\ell_1} \mathbb{A}_2 \xrightarrow{\ell_2} \mathbb{B} =_\rho \mathbb{A}_2 \xrightarrow{\ell_2} \mathbb{A}_1 \xrightarrow{\ell_1} \mathbb{B}}$$

$$\frac{}{(\lambda^\ell X.\lambda^k Y.\mathbf{A}) =_\rho (\lambda^k Y.\lambda^\ell X.\mathbf{A})} \qquad \frac{}{[\mathbf{A}@_\ell\mathbf{B}@_k\mathbf{C}] =_\rho [\mathbf{A}@_k\mathbf{C}@_\ell\mathbf{B}]}$$

Figure 19: A label-selective extension to $\mathcal{MV}$.

in Lisp). Finally, we will use the n-ary notation $[\mathbf{A}@_\ell\mathbf{B}@_k\mathbf{C}]$ as an abbreviation for $[[\mathbf{A}@_\ell\mathbf{B}]@_k\mathbf{C}]$.

The extensions to the respective inference systems can be found in Fig. 19. In particular, we consider labeled application/abstraction to be commutative (they are associative by construction, since types are left-associative). With this extension, to $\mathcal{MV}$ we can for instance have a constant div for integer division and express the term $5\mathsf{div}2$ as $[\mathsf{div}@_{\mathsf{dividend}}5@_{\mathsf{divisor}}2]$ or $[\mathsf{div}@_{\mathsf{divisor}}2@_{\mathsf{dividend}}5]$

5.3. Dependent (Record) Sorts

Label-selectivity gives us another advantage, we can extend it to a system with dependent sorts, if we allow terms and labels of type $\beta \to o$ to appear as base sorts locally. Conceptually, in $\mathcal{BMV}$, sorts are unary predicate *constants*, so the generalization is not as large as it seems at first. Let us look at the following formalization of associativity:

$$\mathsf{assoc} := \lambda^{\mathsf{Set}} S.\lambda^{\mathsf{Op}} F.\forall X_S Y_S Z_S.FX(FYZ) = F(FXY)Z \tag{1}$$

In $\mathcal{BMV}$, this would have the sort $\mathbb{T}\mathsf{op}_{\alpha\to o} \to (\mathbb{A} \to \mathbb{A} \to \mathbb{A}) \to \mathbb{T}\mathsf{op}_o$ for some a priori given sort $\mathbb{A}$. We would however to have $[\mathsf{assoc}@_{\mathsf{Set}}\mathbf{S}]$ (associativity on a given set $\mathbf{S}$) to have sort $(\mathbf{S} \to \mathbf{S} \to \mathbf{S}) \to \mathbb{T}\mathsf{op}_o$, i.e. to be a predicate on binary functions on $\mathbf{S}$.

If we extend the set of base sorts by variables of type $\alpha \to o$ (and of course make the sort typing judgment dependent on a context Γ, by extending all rules in Fig. 15 with contexts in the obvious way) and add the first two rules in Fig. 20 to $\mathcal{MV}$, then we obtain the following sort derivation, which gives us a sort that *does* show the dependence missing above.

$$\dfrac{\dfrac{\dfrac{\vdots}{[S{::}\mathbb{T}\mathsf{op}_{\alpha\to o}],[F{::}S\to S\to S]\vdash^s_\Sigma (\forall X_S Y_S Z_S.FX(FYZ)=F(FXY)Z){::}\mathbb{T}\mathsf{op}_o}}{[S{::}\mathbb{T}\mathsf{op}_{\alpha\to o}]\vdash^s_\Sigma \lambda^{\mathsf{Op}}F.\forall X_S Y_S Z_S.FX(FYZ)=F(FXY)Z{::}(S\to S\to S)\xrightarrow{\mathsf{Op}}\mathbb{T}\mathsf{op}_o}}{\vdash^s_\Sigma \mathsf{assoc}{::}\mathbb{T}\mathsf{op}_{\alpha\to o}\xrightarrow{\mathsf{Set}}(\mathsf{Set}^3)\xrightarrow{\mathsf{Op}}\mathbb{T}\mathsf{op}_o}$$

Here (and in the following) we use $\mathbb{A}^3$ as an abbreviation for the sort $\mathbb{A}\to\mathbb{A}\to\mathbb{A}$.

Unfortunately, the sorts discussed so far are not yet expressive enough for a direct representation of common mathematical structures such as semigroups. A semigroup is a pair $(S,\circ)$, were S is an arbitrary set and $\circ{:}\, S\times S\longrightarrow S$ is an associative binary function on S. Just as in the case of associativity discussed in section 5.3, we would like to represent S as a sort $\mathbb{S}$ and $\circ$ as a function of sort $\mathbb{S}\to\mathbb{S}\to\mathbb{S}$ in a record of type $\{\!\{\mathsf{Set}{:}\,\alpha\to o,\mathsf{Op}{:}\,\alpha\to\alpha\to\alpha\}\!\}$. However, in the system developed so far, we cannot express a record sort like

$$\mathbb{S}\mathsf{etop} := \{\!\{\mathsf{Set}{::}\mathbb{T}\mathsf{op}_{\alpha\to o},\mathsf{Op}{::}\mathsf{Set}\to\mathsf{Set}\to\mathsf{Set}\}\!\}$$

The second two rules in Fig. 20 extend $\mathcal{MV}$ by **very dependent record sorts**. This name is chosen to resemble Jason Hickey's "very dependent record types" (31) and serve the same purpose, even if the formalization on the level of sorts is much more unproblematic, since there are no consistency problems involved: Well-typedness is preserved at the level of (simple) record types.

To make the records dependent, we have to serialize the record construction rule from Fig. 14. Technically, we will (ab)use the context to store the necessary assumptions about the feature values and use the standard record **merge** operator $\otimes$ to write down the rules in Fig. 20.

Let $\mathbb{N}$ be the set of natural numbers and $+{::}\mathbb{N}^3$ the addition function on natural numbers, then we have the following sort derivation in $\mathcal{MV}$.

$$\dfrac{\dfrac{\dfrac{\vdash^s_\Sigma +{::}\mathbb{N}^3}{[\mathsf{Set}=\mathbb{N}]\vdash^s_\Sigma +{::}\mathsf{Set}^3}}{[\mathsf{Set}=\mathbb{N}]\vdash^s_\Sigma \{\!\{\mathsf{Op}=+\}\!\}{::}\{\!\{\mathsf{Op}{::}\mathsf{Set}^3\}\!\}\quad \vdash^s_\Sigma \mathbb{T}\mathsf{op}_{\iota\to o}\succ \iota\to o}}{\vdash^s_\Sigma \{\!\{\mathsf{Set}=\mathbb{N},\mathsf{Op}=+\}\!\}{::}\mathbb{S}\mathsf{etop}}$$

$$\frac{\Gamma,[X::\mathbb{B}] \models^s_\Sigma \mathbf{A}::\mathbb{A} \quad \vdash^\iota_\Sigma \mathbb{B}\rhd \beta\to o}{\Gamma \models^s_\Sigma \lambda^\ell X.\mathbf{A}::\mathbb{B}\xrightarrow{\ell}[\ell/X]\mathbb{A}} \qquad \frac{\Gamma \models^s_\Sigma \mathbf{A}::\mathbb{C}\xrightarrow{\ell}\mathbb{A} \quad \Gamma\models^s_\Sigma \mathbf{C}::\mathbb{C}}{\Gamma\models^s_\Sigma \mathbf{AC}::[\mathbf{C}/\ell]\mathbb{A}}$$

$$\frac{\Gamma,[\ell=\mathbf{A}]\models^s_\Sigma \mathbf{B}::\mathbb{B} \quad \Gamma\models^s_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma\vdash^\iota_\Sigma \mathbb{A}\rhd\alpha\to o}{\Gamma\models^s_\Sigma \{\!\{\ell=\mathbf{A}\}\!\}\otimes\mathbf{B}::\{\!\{\ell::\mathbb{A}\}\!\}\otimes\mathbb{B}} \qquad \frac{\Gamma\models^s_\Sigma [\mathbf{B}/\ell]\mathbf{A}::[\mathbf{B}/\ell]\mathbb{A}}{\Gamma,[\ell=\mathbf{B}]\models^s_\Sigma \mathbf{A}::\mathbb{A}}$$

Figure 20: Extending $\mathcal{MV}$ by dependent sorts

$$\frac{}{\models^s_\Sigma \{\mathbb{A}|\mathbf{P}\}\leq\mathbb{A}} \qquad \frac{\Gamma\models^s_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma\not\models^s_\Sigma \mathbf{PA}}{\Gamma\models^s_\Sigma \mathbf{A}::\{\mathbb{A}|\mathbf{P}\}}$$

$$\frac{\vdash^\iota_\Sigma \mathbf{P}:\alpha\to o \quad \vdash^\iota_\Sigma \mathbb{A}\rhd\alpha}{\vdash^\iota_\Sigma \{\mathbb{A}|\mathbf{P}\}:\alpha} \qquad \frac{}{\Gamma,[X::\{\mathbb{A}|\mathbf{P}\}]\not\models^s_\Sigma \mathbf{P}X}$$

Figure 21: Augmenting $\mathcal{MV}$ by Selection Sorts

5.4. Selection Sorts and Semigroups

In this section, we will fortify our intuition about $\mathcal{MV}$ by considering an example from elementary algebra: semigroups. To be able to handle them naturally, we will need to upgrade the system by **selection sorts**. Concretely, we use a new sort constructor $\{\cdot|\cdot\}$ that yields a new (base) sort $\{\mathbb{A}|\mathbf{P}\}$ for a given sort $\mathbb{A}$ with $\vdash^\iota_\Sigma \mathbb{A}\rhd\alpha$ and a closed term $\mathbf{P}$ of type $\alpha\to o$. Intuitively, this sort corresponds to the set of all objects of sort $\mathbb{A}$, on which $\mathbf{P}$ holds. Consider for instance the set of continuous real functions, that we can model as the sort $\mathbb{C}:=\{(\mathbb{R}\to\mathbb{R})|(\lambda X.\forall\epsilon.\exists\delta.\ldots)\}$. Now we can represent the theorem that the sum of two continuous functions is again continuous by $\forall F_\mathbb{C} G_\mathbb{C}.\mathbb{C}(\lambda X_\mathbb{R}.+(FX)(GX))$ If we want to prove this lemma, we have to be able to expand the definitions of the sort $\mathbb{C}$, which explains the necessity of the last axiom in Fig. 21. Note that once we have proven this theorem, we can interpret it as a term declaration, add it to the signature, and directly use it for further sort computations.

But let us come back to the problem of modeling semigroups. We have seen in section 5.3 that we can represent the structure consisting of a set and an operation on this set by the sort $\mathbb{S}\mathsf{etop}=\{\!\{\mathsf{Set}::\mathbb{T}\mathsf{op}_{\alpha\to o},\mathsf{Op}::\mathsf{Set}^3\}\!\}$, thus we can represent the sort of all semigroups by

$$\mathbb{S}\mathsf{emigroup}:=\{\mathbb{S}\mathsf{etop}|\mathbf{A}\}\qquad\text{where}\qquad \mathbf{A}:=(\lambda X.[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})])$$

As we have seen above, we have $\vdash^s_\Sigma \mathsf{assoc}::\mathbb{A} := (\mathbb{T}\mathsf{op}_{\alpha\to o}) \xrightarrow{\mathsf{Set}} (\mathsf{Set}^3) \xrightarrow{\mathsf{Op}} \mathbb{T}\mathsf{op}_o$ and therefore we can show that $\mathbb{S}\mathsf{emigroup}$ is a well-typed sort.

$$\dfrac{\dfrac{\dfrac{\dfrac{[X::\mathbb{S}\mathsf{etop}] \vdash^s_\Sigma \mathsf{assoc}::\mathbb{A} \quad [X::\mathbb{S}\mathsf{etop}] \vdash^s_\Sigma X.\mathsf{Set}::\mathbb{T}\mathsf{op}_{\alpha\to o}}{[X::\mathbb{S}\mathsf{etop}] \vdash^s_\Sigma [\mathsf{assoc}@_{\mathsf{Set}}X.\mathsf{Set}]::(X.\mathsf{Set})^3 \xrightarrow{\mathsf{Op}} \mathbb{T}\mathsf{op}_o} \quad \dfrac{\vdots}{[X::\mathbb{S}\mathsf{etop}] \vdash^s_\Sigma X.\mathsf{Op}::(X.\mathsf{Set})^3}}{[X::\mathbb{S}\mathsf{etop}] \vdash^s_\Sigma [\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})]::\mathbb{T}\mathsf{op}_o}}{\vdash^s_\Sigma \lambda X.[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})]::\mathbb{S}\mathsf{etop} \to \mathbb{T}\mathsf{op}_o}}{\vdash^\iota_\Sigma \{\mathbb{S}\mathsf{etop}|\lambda X.[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})]\} \succ \{\!\{\mathsf{Set}\colon \alpha \to o, \mathsf{Op}\colon \alpha^3\}\!\}}$$

To see how we can use the selection sorts, let us now prove that the operation of a semigroup is associative on its set, i.e. we want to prove the formula $\Pi^{\mathbb{S}\mathsf{emigroup}}\mathbf{A}$, then – using $\mathbb{S}\mathsf{emigroup} = \{\mathbb{S}\mathsf{etop}|\mathbf{A}\}$ – we have $[X::\mathbb{S}\mathsf{emigroup}] \vdash^s_\Sigma \mathbf{A}X$ by the axiom in Fig. 20 and thus $\vdash^s_\Sigma \Pi^{\mathbb{S}\mathsf{emigroup}}\mathbf{A}$ by the sorted quantifier introduction rule from Fig. 18.

5.5. Relativization for extended $\mathcal{MV}$

In this section, we will present two relativization morphisms that show that the records and selection sorts in $\mathcal{MV}$ can be eliminated and that therefore $\mathcal{MV}$ is a conservative extension of the simply typed λ-calculus.

For constructing an elimination morphism for label-selective applications and abstractions, we will make use of the record calculus in $\mathcal{BMV}$. Intuitively, the translation works like this: maximal chains of labeled abstractions are represented as single abstractions over records with the same labels. Similarly, maximal chains of labeled applications as applications to single records. In our example involving integer division we would translate:

$$\begin{array}{lll} [[\mathsf{div}@_{\mathsf{dividend}}5]@_{\mathsf{divisor}}2] & \text{to} & \mathsf{div}\{\!\{\mathsf{dividend} = 5, \mathsf{divisor} = 2\}\!\} \\ \mathsf{div} =_\eta \lambda^{\mathsf{dividend}} X.\lambda^{\mathsf{divisor}} Y.\mathbf{A} & \text{to} & \lambda Z_{\{\!\{\mathsf{dividend}:\iota,\mathsf{divisor}:\iota\}\!\}}.\mathbf{A}' \end{array}$$

The language morphism $\overline{\cdot}$ is given by the three equations below.

$$\begin{array}{rcl} \overline{\mathbb{A}_1 \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_{n-1}} \mathbb{A}_n \xrightarrow{\ell_n} \mathbb{B}} & = & \{\!\{\ell_1::\overline{\mathbb{A}_1}\}\!\} \otimes [\mathbb{A}_1/\ell_1](\{\!\{\ell_2::\overline{\mathbb{A}_2}\}\!\} \otimes [\mathbb{A}_2/\ell_2](\ldots \\ & & \otimes [\mathbb{A}_{n-1}/\ell_{n-1}]\{\!\{\ell_n::\overline{\mathbb{A}_n}\}\!\})\ldots) \to \overline{\mathbb{B}} \\ \overline{\lambda^{\ell_1}X_1.\ldots\lambda^{\ell_n}X_n.\mathbf{A}} & = & \lambda Z.\overline{[Z.\ell_1/X_1],\ldots,[Z.\ell_n/X_n]\mathbf{A}} \\ \overline{[\mathbf{B}@_{\ell_1}\mathbf{C}_1@\ldots@_{\ell_n}\mathbf{C}_n]} & = & \overline{\mathbf{A}}\{\!\{\ell_1 = \overline{\mathbf{C}_1},\ldots,\ell_n = \overline{\mathbf{C}_n}\}\!\} \end{array}$$

Here, $\mathbf{A}$ and $\mathbb{B}$ must be of base type and $\mathbf{B}$ may not be an application, so that we always transform maximal sequences of arguments and bound variables in one step. Note that this is not a restriction of generality, since we can always η-expand. This translation uses the fact that entries in a record do not have a

$$\frac{\Gamma \vdash^{\iota}_{\Sigma} \Pi^*\mathbb{A}(\lambda X.\mathbf{P}X \Rightarrow \mathbf{B}X) \quad \Gamma \vdash^{s}_{\Sigma} \mathbf{A}::\mathbb{A}}{\dfrac{\Gamma \vdash^{\iota}_{\Sigma} \mathbf{PA} \Rightarrow \mathbf{BA} \qquad \Gamma \vdash^{\iota}_{\Sigma} \mathbf{PA}}{\Gamma \vdash^{\iota}_{\Sigma} \mathbf{BA}}} \qquad \frac{\dfrac{\dfrac{[\Gamma, [X::\mathbb{A}] \vdash^{\iota}_{\Sigma} \mathbf{P}X]^1}{\Gamma, [X::\mathbb{A}] \vdash^{\iota}_{\Sigma} \mathbf{A}X}}{\Gamma, [X::\mathbb{A}] \vdash^{\iota}_{\Sigma} \mathbf{P}X \Rightarrow \mathbf{A}X}\,1}{\dfrac{\Gamma \vdash^{\iota}_{\Sigma} (\lambda Y.\mathbf{P}Y \Rightarrow \mathbf{A}Y)X}{\Gamma \vdash^{\iota}_{\Sigma} \Pi^*\mathbb{A}(\lambda X.\mathbf{P}X \Rightarrow \mathbf{A}X)}}$$

Figure 22: Relativizing Selection Sorts (from the Calculus Morphism)

fixed order to obtain the order-independence of labeled abstraction:

$$\begin{aligned}
\overline{[\mathbf{B}@_{\ell_1}\mathbf{C}_1@_{\ell_2}\mathbf{C}_2]} &= \overline{\mathbf{B}}\{\!\{\ell_1 = \overline{\mathbf{A}_1}, \ell_2 = \overline{\mathbf{A}_2}\}\!\} \\
&= \overline{\mathbf{B}}\{\!\{\ell_2 = \overline{\mathbf{A}_2}, \ell_1 = \overline{\mathbf{A}_1}\}\!\} = \overline{[\mathbf{B}@_{\ell_2}\mathbf{C}_2@_{\ell_1}\mathbf{C}_1]} \\
\overline{\lambda^{\ell_1}X_1.\lambda^{\ell_2}X_2.\mathbf{A}} &= \lambda Z.\overline{[Z.\ell_1/X_1], [Z.\ell_2/X_2]\mathbf{A}} \\
&= \lambda Z.\overline{[Z.\ell_2/X_2], [Z.\ell_1/X_1]\mathbf{A}} = \overline{\lambda^{\ell_2}X_2.\lambda^{\ell_1}X_1.\mathbf{A}}
\end{aligned}$$

The argumentation for the types and sorts is analogous to the case for applications. A tedious but simple calculation with ND proofs shows that this language morphism can be extended to a calculus morphism. In particular, the inference rules in Fig. 19 turn into trivial ND proofs about records and their sets of labels.

We will not go in to details for relativizing away record sorts and types. This can be achieved by using one of two standard techniques. By introducing a new type ρ for record objects and modeling all record labels as partial functions from records to values. Thus a record $\mathcal{BMV}$ of type $\{\!\{\ell\!:\!\alpha, k\!:\!\beta\}\!\}$ would receive type ρ and we would extend the signature by functions $f_\ell\!:\!\rho \to \alpha$ and $f_k\!:\!\rho \to \beta$. Alternatively, one can fix an ordering on record labels and map records to n-tuples.

The logic morphism for eliminating selection sorts uses the fact that we can define selection sorts by relativization using $\mathcal{P}_{\{\mathbb{A}|\mathbf{P}\}} := \lambda X.\mathcal{P}_{\mathbb{A}}X \wedge \mathbf{P}X$, just like we did for ordinary sort relativization in section 4.1. Thus the language morphism relativizes all occurrences of formulae of the form $\Pi^{\{\mathbb{A}|\mathbf{P}\}}\mathbf{A}$ to $\Pi^{\mathbb{A}}(\lambda X.\mathbf{P}X \Rightarrow \mathbf{A})$. The calculus morphism is given in Fig. 22, it shows the relativization rules for $\Pi^{\{\mathbb{A}|\mathbf{P}\}}$ elimination and introduction. Finally, the relativization of the axiom in Fig. 20 is a trivial. Note that this allows any proof $\Gamma, [X::\{\mathbb{A}|\mathbf{P}\}] \vdash^{s}_{\Sigma} \mathbf{A}$ to be transformed into one of the form for $\Gamma, [X::\mathbb{A}] \vdash^{\iota}_{\Sigma} \mathbf{A}$ under the assumption of $\Gamma, [X::\mathbb{A}] \vdash^{\iota}_{\Sigma} \mathbf{P}X$. This justifies the implication introduction step in the transformation of the second derivation in Fig. 22 (we only have to make sure that the first rule is eliminated first during the transformation).

6. Conclusion

We have described the data model of the MBase system, a web-based, distributed mathematical knowledge base (it is realized as a mathematical service in MathWeb) that offers the infrastructure for a universal repository of formalized mathematics. We have explained how the distribution of MBase supports repositories from the archive server level, where large parts of formalized mathematics are kept centrally, to the personal (scratch-pad) level, where a researcher has a personal MBase to manage her mathematical theories under development. In between there may be workgroup or institute servers, that support collaborative development of mathematical theories.

We have presented a methodology for building a hierarchy of representation languages for a mathematical knowledge base. We have shown that using logic morphisms allows us to define high-level language features, such as dependent sorts in a step-by-step manner from lower (and more standard) ones, and ultimately from axiomatic set theory. The intended meaning of the more expressive logical systems is induced via the logical morphisms from the simpler logical systems, thus a knowledge base that is built up using the method proposed in this article is truly grounded in set theory. An implementation of the logic morphisms in a knowledge base system, such as the MBase system under development in Saarbrücken, will give constructive evidence to the old belief of working mathematicians that all of mathematics can be relativized (and thus grounded) in set theory.

We have instantiated this methodology by sketching the development of a sorted λ-calculus that we claim is well-suited for formalizing mathematical practice. It is an extension of the sorted λ-calculus from (41) by dependent function–, record–, and selection sorts. We have sketched the relativizations needed to integrate it into the MBase system. The advantage of a sorted formulation over a classical type-theoretic one (e.g. LF dependent type discipline or Jason Hickeys "very dependent record types" (31)) is that consistency is a consequence of the relativization, since the sorts are typed. This makes all objects simply typed, and hence important meta-theorems like strong normalization of the built-in reductions are relatively easy to prove.

The next step will be to develop inference procedures like higher-order matching that are needed for answering high-level queries in MBase. We conjecture that this will be possible by adapting the methods (in particular, the structure theorem) from (41). In fact, one key motivation to extend known representation languages for mathematics by the additional structure developed in this article was to use the additional structure for inference purposes. We conjecture that the availability of such inference procedures will decide on the usefulness, and thus ultimately on the success of a mathematical knowledge base system.

The MBase service can be used as an ontology server giving a semantics for system integration and furthermore, the formal representation of knowledge

elements allows semantics-based retrieval of distributed mathematical facts. Possible queries to MBase could be glossed as follows:

1. *For a formula* $\mathbf{A}$, *give me all knowledge elements* $\mathbf{B}$, *which are instances of* $\mathbf{A}$; This kind of queries allows searching for all instances of a given schema. This is particularly valuable if the formalism allows function and predicate variables. For instance a schema $\mathbf{A} = \forall X, Y.F(X + Y = Y + X)$ allows to search for knowledge elements that use/assert the commutativity of addition using the variable F to return the context.
2. *Give me all theorems/simplifiers that are applicable to a formula* $\mathbf{C}$. In this query, matching has to be augmented by quantifier elimination. It is interesting to obtain a set of possible forward inferences in a concrete situation.
3. *Classify the mathematical structure given by the set* $\mathcal{S}$ *of axioms.* This kind of query could be issued, in order to retrieve the mathematical knowledge about a concrete mathematical structure (which may turn out to be a well-known one like a ring in disguise). A possible follow-up query could be one whether there are "interesting" specializations of the structure that would allow for stronger results.

These queries crucially depend on the notion of matching employed. The more expressive (higher up in the taxonomy in Fig. 7) the representation formalism is, the more powerful the matching algorithms can become (e.g. higher-order matching in $\Lambda^{\rightarrow}$).

It will be necessary to augment the known matching algorithms to make them aware of the logic morphisms: If we are only looking for formulae, building in the language morphisms will be sufficient; if we want to be able to search for proofs of a certain form, it will also be necessary to extend matching to proofs and also to build in calculus morphisms. This will generate interesting research questions that we will address in due course, but not in this article.

Finally, there are many kinds of data mining applications that could be run on a larger collection of formal mathematical knowledge. For instance it would be interesting to search for similarity of mathematical structures. Also to search for possible logic morphisms between theories that may be reused later to transport proofs.

References

[1] A. Adams, H. Gottliebsen, S. Linton, and U. Martin. VSDITLU: a Verifiable Symbolic Definite Integral Table Look-up. In Ganzinger (28), pages 112–126.

[2] Hassan Aït-Kaci and Jacques Garrigue. Label-selective lambda-calculus: Syntax and confluence. In *Proceedings of the 13th International Conference*

on Foundations of Software Technology and Theoretical Computer Science, volume 761 of *LNCS*, Bombay, India, 1993.

[3] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[4] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem-proving system for classical type theory. *Journal of Automated Reasoning*, 16:321–353, 1996.

[5] Alessandro Armando, Michael Kohlhase, and Silvio Ranise. Communication protocols for mathematical services based on KQML and OMRS. In Manfred Kerber and Michael Kohlhase, editors, *CALCULEMUS-2000, Systems for Integrated Computation and Deduction*, St. Andrews, Scotland, 2000. AKPeters. in press.

[6] Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy and D. Bert, editors, *Proceedings Workshop on Algebraic Development Techniques, WADT-99*. Springer, LNCS 1827, 2000.

[7] C. Ballarin, K. Homann, and J. Calmet. Theorems and algorithms: An interface between isabelle and maple. In *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'95)*, pages 150–157. ACM Press, 1995.

[8] A. Bauer, E. Clarke, and X. Zhao. Analytica — an Experiment in Combining Theorem Proving and Symbolic Computation. *Journal of Automated Reasoning*, 21(3):295–325, 1998.

[9] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In William McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, number 1249 in LNAI, pages 252–255, Townsville, Australia, 1997. Springer Verlag.

[10] Christoph Benzmüller, Matthew Bishop, and Volker Sorge. Integrating TPS and ΩMEGA. *Journal of Universal Computer Science*, 5(2), 1999.

[11] Matthew Bishop and Peter B. Andrews. Selectively instantiating definitions. In Kirchner and Kirchner (40), pages 365–380.

[12] R. Boulton, K. Slind, A. Bundy, and M. Gordon. An interface between CLAM and HOL. In Jim Grundy and Malcolm Newey, editors, *Theorem Proving in Higher Order Logics: Emerging Trends*, Technical Report CS-98-08, pages 87–104, The Australian National University Canberra, 1998.

[13] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML). W3C Recommendation TR-XML, World Wide Web Consortium, December 1997. Available at `http://www.w3.org/TR/PR-xml.html`.

[14] Jacques Calmet and Karsten Homann. Towards a mathematics software bus. *Theoretical Computer Science*, 107, 1997.

[15] Olga Caprotti and Arjeh M. Cohen. Draft of the Open Math standard. The Open Math Society, `http://www.nag.co.uk/projects/OpenMath/omstd/`, 1998.

[16] Lassaad Cheikhrouhou and Jörg Siekmann. Planning diagonalization proofs. In Fausto Giunchiglia, editor, *Artificial Intelligence: Methodology, Systems and Applications*, number 1480 in LNAI, pages 167–180, Sozopol, Bulgaria, 1998. Springer Verlag.

[17] Language Design Task Group CoFI. Casl — the CoFI algebraic specification language — summary, version 1.0. Technical report, `http://www.brics.dk/Projects/CoFI`, 1998.

[18] Arjeh Cohen, Hans Cuypers, and Hans Sterk. *Algebra Interactive!* Springer Verlag, 1999. Interactive Book on CD.

[19] N. G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In R. P Nederpelt, J. H. Geuvers, and R. C. de Vrijer, editors, *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, pages 865 – 935. Elsevier, 1994.

[20] Stephen Deach. Extensible stylesheet language (xsl) specification. W3c working draft, W3C, 1999. Available at `http://www.w3.org/TR/WD-xsl`.

[21] H.D. Ebbinghaus. *Einführung in die Mengenlehre.* Wissenschaftliche Buchgesellschaft, 1977.

[22] William M. Farmer. Theory interpretation in simple type theory. In *HOA'93, an International Workshop on Higher-order Algebra, Logic and Term Rewriting*, volume 816 of *LNCS*, Amsterdam, The Netherlands, 1993. Springer Verlag.

[23] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, October 1993.

[24] T. Finin and R. Fritzson. KQML — a language and protocol for knowledge and information exchange. In *Proceedings of the 13th Intl. Distributed Artificial Intelligence Workshop*, pages 127–136, Seattle, WA, USA, 1994.

[25] Adolf Abraham Fraenkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, 86:230–237, 1922.

[26] Andreas Franke, Stephan M. Hess, Christoph G. Jung, Michael Kohlhase, and Volker Sorge. Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, 5:156–187, 1999.

[27] Andreas Franke and Michael Kohlhase. System description: MathWeb, an agent-based communication layer for distributed automated theorem proving. In Ganzinger (28), pages 217–221.

[28] Harald Ganzinger, editor. *Proceedings of the 16th Conference on Automated Deduction*, number 1632 in LNAI. Springer Verlag, 1999.

[29] Gerhard Gentzen. Untersuchungen über das logische Schließen I & II. *Mathematische Zeitschrift*, 39:176–210, 572–595, 1935.

[ILF] The ILF Group. The ILF mathematical library. Internet page at `http://www-irm.mathematik.hu-berlin.de/~ilf/mathlib.html`.

[30] J. Harrison and L. Théry. A Skeptic's Approach to Combining HOL and Maple. *Journal of Automated Reasoning*, 21(3):279–294, 1998.

[31] Jason J. Hickey. Formal objects in type theory using very dependent types. In *Foundations of of Object Oriented Languages 3*, 1996.

[32] J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, 1986.

[33] Douglas Howe. Semantic foundations for embedding hol in nuprl. In Martin Wirsing and Maurice Nivat, editors, *Algebraic Methodolgy and Software Technology*, volume 1101 of *LNCS*, pages 85–101. Springer Verlag, 1996.

[34] Xiaorong Huang and Armin Fiedler. Presenting machine-found proofs. In McRobbie and Slaney (46), pages 221–225.

[35] Dieter Hutter. Reasoning about theories. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1999.

[36] Dieter Hutter and Claus Sengler. INKA - The Next Generation. In McRobbie and Slaney (46), pages 288–292.

[IMPS] The imps online theory library. Internet interface at `ftp://math.harvard.edu/imps/imps_html/theory-library.html`.

[37] Patrick Ion and Robert Miner. Mathematical Markup Language (MathML) 1.0 specification. W3C Recommendation REC-MathML-19980407, World Wide Web Consortium, April 1998. Available at `http://www.w3.org/TR/REC-MathML/`.

[IsabelleKB] The isabelle online theory library. Internet interface at `http://www4.informatik.tu-muenchen.de/~isabelle/library-Isabelle98-1`.

[38] Manfred Kerber. How to prove higher order theorems in first order logic. In John Mylopoulos and Ray Reiter, editors, *Proceedings IJCAI'91*, pages 137–142, 1991. Morgan Kaufmann.

[39] Manfred Kerber, Michael Kohlhase, and Volker Sorge. Integrating computer algebra into proof planning. *Journal of Automated Reasoning*, 21(3):327–355, 1998.

[40] Claude Kirchner and Hélène Kirchner, editors. *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in LNAI. Springer Verlag, 1998.

[41] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. PhD thesis, Universität des Saarlandes, 1994.

[42] Michael Kohlhase. OMDoc: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *Proceedings AISC'2000*, 2000. forthcoming.

[43] Michael Kohlhase. OMDoc: Towards an OPENMATH representation of mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/omdoc`.

[44] Michael Kohlhase and Frank Pfenning. Unification in a λ-calculus with intersection types. In Dale Miller, editor, *Proceedings of the International Logic Programming Sympsion. ILPS'93*, pages 488–505. MIT Press, 1993.

[45] J. Loeckx, H.-D. Ehrig, and M Wolf. *Specification of Abstract Data Types*. Teubner, Chichester;New York;Brisbane, 1996. ISBN 3-519-02115-3.

[46] M.A. McRobbie and J.K. Slaney, editors. *Proceedings of the 13th Conference on Automated Deduction*, number 1104 in LNAI, New Brunswick, NJ, USA, 1996. Springer Verlag.

[47] Atsushi Ohori. A polymorphic record calculus and its compilation. *ACM Transactions on Programming Languages and Systems*, 17(6):844–895, 1995.

[48] Lawrence C. Paulson. Set theory for verification: I. from foundations to functions. *Journal of Automated Reasoning*, 11:353–389, 1993.

[PVS] Pvs libraries. `http://pvs.csl.sri.com/libraries.html`.

[49] Art Quaife. Automated deduction in von Neumann-Bernays-Gödel set theory. *Journal of Automated Reasoning*, 8(1):91–148, 1992.

[50] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.0 Specification. W3C Recommendation REC-html40, World Wide Web Consortium, April 1998. Available at `http://www.w3.org/TR/PR-xml.html`.

[51] Julian D.C. Richardson, Alan Smaill, and Ian M. Green. System description: Proof planning in higher-order logic with *λclam*. In Kirchner and Kirchner (40).

[52] Jörg Siekmann, Christoph Benzmüller, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Martin Pollet, Volker Sorge, Carsten Ullrich, and Jürgen Zimmer. Adaptive course generation and presentation. In P. Brusilovski, editor, *Proceedings of ITS-2000 workshop on Adaptive and Intelligent Web-Based Education Systems*, Montreal, 2000.

[53] G. Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343. Springer-Verlag, Berlin, 1995.