Semantics of OpenMath and MathML3

Michael Kohlhase and Florian Rabe

Abstract. Even though OpenMath has been around for more than 10 years, there is still confusion about the "semantics of OpenMath". As the recent MathML3 recommendation semantically bases Content MathML on OpenMath Objects, this question becomes more pressing.

One source of confusions about OpenMath semantics is that it is given on two levels: a very weak algebraic semantics for expression trees, which is extended by considering mathematical properties in content dictionaries that interpret the meaning of (constant) symbols. While this two-leveled way to interpret objects is well-understood in logic, it has not been spelt out rigorously for OpenMath.

We present two denotational semantics for OpenMath: a construction-oriented semantics that achieves full coverage of all legal OpenMath expressions at the cost of great conceptual complexity, and a symbol-oriented one for a subset of OpenMath expressions. This subset is given by a variant of the OpenMath 2 role system, which – we claim – does not exclude any representations of meaningful mathematical objects.

1. Introduction

MathML [ABC+03] and OpenMath [BCC+04] are standards for the representation and communication of mathematical objects. Even though they have been around for more than 10 years, there is still confusion about the "semantics of OpenMath". As the recent MathML3 recommendation [ABC+10] semantically bases Content MathML on OpenMath Objects, this question becomes more pressing.

1.1. OpenMath and MathML

MathML comes in two parts: *presentation* MathML, which provides XML-based layout primitives for the traditional two-dimensional notation of mathematical formulae and *content* MathML, which focuses on encoding the meaning of objects rather than visual representations to allow the free exchange of mathematical objects between software systems and human beings. OpenMath has the same goals as content MathML, but was developed by a different community with slightly different intuitions. Both representation formats represent mathematical objects as expression trees. Content MathML tries to cover all of school and engineering mathematics (the "K-14" fragment) in a representation format intuitive to mathematicians, and OpenMath concentrates on an extensible framework built on a minimal structural core language with a well-defined extension mechanism. Where MathML supplies more than a dozen elements for special constructions, OpenMath only supplies concepts for function application (`OMA`), binding constructions (`OMBIND`), and attributions (`OMATTR`). Where MathML provides close to 100 elements for the K-14 fragment, OpenMath gets by with only an `OMS` element that identifies symbols by pointing to declarations in an open-ended set of Content Dictionaries.

An OpenMath Content Dictionary (CD) is a document that declares names (OpenMath "symbols") for basic mathematical concepts and objects. CDs act as the unique points of reference for OpenMath symbols (via `OMS` elements) and thus supply a notion of context that situates and disambiguates OpenMath expression trees. To maximize modularity and reuse, a CD typically contains a relatively small collection of definitions for closely related concepts. The OpenMath Society maintains a large set of public CDs [OMC], including CDs for all pre-defined symbols in MathML2. There is a process for contributing privately developed CDs to the OpenMath Society repository to facilitate discovery and reuse. OpenMath does not require CDs be publicly available, though in most situations the goals of semantic markup will be best served by referencing public CDs available to all user agents.

To avoid fragmentation and to smooth out interoperability obstacles, an effort has been made to align OpenMath and MathML semantically. To remedy the lack of regularity and specified meaning in MathML, content MathML was extended by concepts like binding structures and full semantic annotations from OpenMath and a structurally regular subset of the extended content MathML was identified that is isomorphic to OpenMath objects. This subset is called **strict content MathML** to contrast it to full content MathML that is seen to strike a more pragmatic balance between regularity and human readability. Full content MathML borrows the semantics from strict MathML by a mapping specified in the MathML3 specification [ABC+10] that defines the meaning of non-strict (**pragmatic**) MathML expressions in terms of strict MathML equivalents. Strict Content MathML in turn obtains its meaning by being an encoding of OpenMath Objects.

In this situation, the "meaning of OpenMath (Objects)" obtains a completely new significance. The aim of this paper is to clarify the status of semantics in OpenMath (and thus content MathML3). We observe a presentational gap between how mathematical objects and theories are conventionally given a meaning and the way OpenMath answers the question. This leads to misunderstandings about the meaning of OpenMath objects and its role in representing mathematical knowledge.

1.2. The Meaning of OpenMath

The OpenMath standard actually gives two answers to the question about the meaning of OpenMath expressions. The first one comes from the fact that OpenMath is intended as a communication standard between mathematical software systems: OpenMath envisions communication via *phrasebooks* ([AvLS98] or see [BCC+04, chapter 1]): Each mathematical software system S is equipped with an OpenMath phrasebook that converts OpenMath expressions from and to the internal representations of the system S. In this "system communication view", the meaning of OpenMath expressions is built into the phrasebooks that (purport to) understand the expression, and the meaning is whatever S (after conversion by the phrasebook) makes it to be. Clearly, this view of meaning is not very helpful, and taken in the radical simplicity we have formulated it here is not an adequate account. After all, the purpose of the OpenMath standard is to synchronize the system-specific representations of objects, so that communication between systems is meaning-preserving. To attain this goal, OpenMath does two things:

1. It defines the class of "OpenMath objects" which acts as the model for encodings of mathematical formulae. OpenMath objects are essentially labeled trees modulo α-conversion for binding structures and flattening for nested semantic annotations. The OpenMath standard considers OpenMath objects as primary citizens and views the "OpenMath XML encoding" as just an incidental design choice for an XML-based markup language. In fact OpenMath specifies another encoding: the "binary encoding" designed to be more space efficient at the cost of being less human-readable.
2. Rather than appealing to mathematical intuition, OpenMath stipulates that phrasebooks should be informed by (mathematical properties in) content dictionaries.

   It is the OpenMath Content Dictionaries which actually hold the meanings of the objects being transmitted. For example if application A is talking to application B, and sends, say, an equation involving multiplication of matrices, then A and B must agree on what a matrix is, and on what matrix multiplication is, and even on what constitutes an equation. All this information is held within some Content Dictionaries which both applications agree upon. [...] The primary use of Content Dictionaries is thought to be for designers of Phrasebooks, the programs which translate between the OpenMath mathematical object and the corresponding (often internal) structure of the particular application in question. [BCC+04, section 4.1]

Even if this is not spelt out in the OpenMath2 standard the *algebra $\mathcal{O}$ of* OpenMath *objects* can be interpreted as an (initial) model for encodings of mathematical formulae.

To the best of our knowledge, this "act of interpretation" has never been backed by a formal mathematical study; which is what prompted the work reported in this paper. As a consequence the "compliance chapter" in the OpenMath standard does not mention mathematical properties in CDs at all. While this can be seen as a failure of OpenMath to supply semantics ("OpenMath *is only syntax*"), we see it as an expression of the OpenMath representational philosophy expressed in

> *OpenMath objects do not specify any computational behavior, they merely represent mathematical expressions. Part of the OpenMath philosophy is to leave it to the application to decide what it does with an object once it has received it. OpenMath is not a query or programming language. Because of this, OpenMath does not prescribe a way of forcing "evaluation" or "simplification" of objects like* $2+3$ *or* $\sin(\pi)$. *Thus, the same object* $2+3$ *could be transformed to* 5 *by a computer algebra system, or displayed as* $2+3$ *by a typesetting tool.* [BCC+04, section 1.5]

Note that since $\mathcal{O}$ is initial, it is essentially unique and identifies (in the sense of "declares to be the same") fewer objects than any other model. As a consequence two mathematical objects must be identical if their OpenMath representations are, but not the other way around. In this sense the initial algebra semantics of OpenMath objects is intentionally weak to make the OpenMath format ontologically unconstrained and thus universally applicable. It basically represents the accepted design choice of representing objects as formulae. Any further (meaning-giving) properties of an object o are relegated to the content dictionaries referenced in o, where they can be specified formally (as "Formal Mathematical Properties" in `FMP` elements containing XML-encoded OpenMath objects) or informally (as "Commented Mathematical Properties" in `CMP` elements containing text). Thus the precision of OpenMath as a representation language can be adapted by supplying CDs to range from fully formal (by providing CDs based on some logical system) to fully informal (where CDs are essentially empty except for declaring symbols).

1.3. Overview of the Paper

When designing a formal language, we have to make a trade-off between expressivity and interpretability. The more flexible we make the language, the more content can be expressed in it; but also, the harder it gets to interpret the language. OpenMath systematically and intentionally errs on the side of expressivity imposing only minimal well-formedness constraints on syntactic objects in the form of a context-free grammar. Consequently, almost all well-formed OpenMath objects do not denote a mathematical object and must be interpreted using special values denoting undefinedness. This is very common in logic and not a problem in itself, but it precludes full-coverage, symbol-oriented semantics.

In this paper, we will develop both

1. a full-coverage, construction-oriented for OpenMath objects, which solves the problem of specifying denotations for OpenMath's feature of arbitrary binders and attributions, and
2. a symbol-oriented semantics for a restricted subset of OpenMath expressions that can serve as a compromise between the spirit of OpenMath and the elegance of a model-theoretical semantics.

The price for giving denotations for all OpenMath objects (under 1.) is that giving individual algebras is ludicrously complicated since they must, e.g., provide an interpretation for *any* object used as a binder, binding *any* number of bound variables, in which each variable carries *any* list of attribution keys, each attributing *any* value. This convoluted semantics is unavoidable and enforced by the generality of the OpenMath standard. But the question arises whether the benefit of arbitrary binders is worth the effort of such a semantics.

The price for semantic elegance (under 2.) is that we need to find a way to restrict the flexibility of OpenMath expressions without sacrificing (important) mathematical examples.

We will develop the full-coverage semantics for OpenMath in two steps. In Section 3, we present an initial algebra semantics of OpenMath objects, and then in Section 4 extend it to take mathematical properties in CDs into account. In Section 5, we discuss how role/type systems can be used to single out subsets that afford symbol-oriented semantics. We show the feasibility of such an approach by providing a role system that strengthens the one specified in the OpenMath 2 standard and exhibit a symbol-oriented semantics. Section 6 concludes the paper.

2. Preliminaries and Related Work

2.1. Model Theoretical Semantics

Model theoretical semantics of logical languages go back to [TV56, Rob50], an overview is given in [BF85]. They are often based on a **universe**, a set that contains the objects the language is designed to talk about (its domain of discourse). The denotation of an object of a formal language is usually defined via an **interpretation function** $[\![-]\!]$, an inductive, compositional function on the syntax. An **algebra** or (especially if propositions and truth values are involved) **model** is a universe together with an interpretation function.

Usually, almost all syntactic objects of the language are interpreted as **intra-universal** entities, i.e., as elements of the universe. Generally, **extra-universal** entities are not desirable because they complicate the semantics. Moreover, they can always be avoided by simply enlarging the universe to encompass them. Yet, there are two conditions under which they are useful. Firstly, there should only be few objects with extra-universal semantics. Usually, these objects are atomic, i.e., certain symbols; if they are composed objects, they should be subject to a comparatively simple language. Secondly, adding them to the universe would substantially complicate the universe. This usually means extending all operations that must be defined for all elements of the universe, which can be very inconvenient.

For instance, in first-order logic, the function and predicate symbols are interpreted extra-universally as functions and relations on the universe. Similarly, the truth values, i.e., the denotations of propositions are extra-universal. This is tolerated because there are only finitely many and only atomic objects with extra-universal semantics. Moreover, keeping them out of the universe is desirable because it permits universes without functions.

A contrasting example is the semantics of higher-order logic (a logic based on the simply typed λ-calculus) [Chu40]. There the universe contains truth values and (arbitrary orders) of functions, so that functions and predicates can be treated intra-universally (at the cost of having a much more complicated universe). Moreover, by using higher-order abstract syntax, even the quantifiers can be treated intra-universally as predicates on predicates, and the only remaining language-level constructions with an extra-universal interpretation are application and λ-abstraction.

The interpretation function is defined by induction on the syntax. Thus, at the very least, an algebra must provide one (intra- or extra-)universal entity for each symbol declared in the signature. If the interpretation of complex objects is defined generically in terms of the interpretations of their components (i.e., ultimately in terms of the interpretations of the symbols), we speak of a **symbol-oriented** semantics. The above semantics for first- and higher-order logic are examples, the former using extra-universal interpretations of the symbols.

Alternatively, algebras may additionally provide extra-universal operations that determine how the interpretation of complex objects is obtained from the interpretation of their components. In that case, we speak of a **construction-oriented** semantics. In this case, the universe is usually so rich that these operations are the only extra-universal entities needed. Consequently, the universe of the algebras is more complex, but once given, all symbols of the signature can easily be interpreted intra-universally.

For example, a construction-oriented semantics for higher-order logic can be given based on applicative structures, which consist of a universe together with an application operator. Thus, the semantics of application can vary with the particular algebra. In categorical logic, this is taken to the extreme by searching for classes of categories that provide just enough structure to interpret all the constructions. For example, for higher-order logic, this leads to algebras based on arbitrary cartesian closed categories [LS86].

2.2. Role and Type Systems

In a **classified** language, the syntactic objects are grouped into classes, algebras provide one universe for each syntactic class, and objects are interpreted as elements of the universe associated with their class. Often some objects belong to no class (usually called ill-formed), which are not interpreted at all. We call the opposite case of a semantics with a single universe **monolithic**.

Classified languages are most common in type theory [WR13, Chu40], where the objects are classified by their type and algebras provide a different universe for every type.

A classified syntax can be achieved using role systems or type systems. Both assume a syntactic class assigned to each symbol (the symbol's role or type) and then extend this assignment to all objects. Moreover, objects in certain syntactic positions are constrained according to their role or type so that many objects become ill-formed. We speak of a **role system** if the syntactic classes are extraneous to the formal language and of a **type system** if they are themselves syntactic objects.

In order to discuss role and type system for OpenMath, we distinguish positive and negative systems. A **negative** system defines some objects to be *ill-formed*, e.g., objects with a certain role/type may be forbidden in certain positions. A **positive** system, on the other hand, defines some objects to be *well-formed*, e.g., only objects with a certain role/type may be allowed in a certain position. A positive system is a necessary requirement for a symbol-oriented semantics: For example, if the possible binders are positively known, the algebra can derive their semantics from the symbols occurring in the binder.

Furthermore, we distinguish universal and limited system. A **universal** system can be applied to all OpenMath objects without loss of usability. A **limited** system applies to a subset of OpenMath objects or to a certain field of applications. In particular, any decidable type system limits attention to the decidable fragment of mathematics. For example, we might limit attention to the case where there is only one key (for type attributions) and only two objects permitted as binders (two symbols for λ and Π).

These distinctions lead to a trade-off between the desirable properties of being positive and universal. Usually, role systems are relatively simple, making this trade-off easier. Type systems are usually much more fine-grained than role systems, and a positive and universal type system for OpenMath is virtually impossible.

Role and Type Systems for OpenMath. In some languages, role/type systems can be strict enough to single out exactly the meaningful objects so that no undefined values are necessary in algebras. This is of course not possible for mathematics in general where meaningfulness is undecidable. Moreover, the design of OpenMath intentionally avoids a commitment to any particular type system. Still, both role and type systems have been given for OpenMath.

The OpenMath standard [BCC+04] already provides a role system. The possible roles are `constant`, `application`, `binder`, `key`. [1] The role system is negative: OM objects are ill-formed if their head does not have the respective role. In particular, no restriction is imposed for symbols without role or for composed objects. (Thus, the effect is marginal because the role restriction can be circumvented altogether by wrapping a symbol in a non-semantic attribution which the standard guarantees does not change the semantics.) Therefore, the system is universal: If no roles are assigned, all objects are well-formed.

In [RK09], we give a more fine-grained role system that uses arities for functions. The system is positive while being as universal as possible. For example, the default arity is that of a function with unlimited arity. Moreover, the binding objects are positively limited: The only permitted binders are symbols with that role and applications of such symbols to arguments.

Neither role system limits the bound variables of binding objects, which would be crucial for a symbol-oriented semantics.

In [Dav99], a simple type system is given. It is positive and intentionally limited to basic cases. It is essentially an extension of simple type theory with n-ary and associative-binary function arguments.

In [?], a stronger type system is given based on the calculus of constructions. It is also positive and limited – in particular, type checking is decidable – but much less limited than the above. It features all possible Π-types of the λ-cube as well as dependent product types. Attributions are limited to a single key for type attributions. Other binders than Π, λ, and Σ are permitted, but these are considered abbreviations of the corresponding functions defined in terms of higher-order abstract syntax.

A similar type system based on categorial types is sketched in [Str04]. It uses a special key for type attributions and represents binding using higher-order abstract syntax. Keys are treated like binary functions.

3. A Construction-Oriented Semantics for OpenMath

We will now define a an algebraic semantic semantics for OpenMath objects building on ideas from [BBK04]. The difference to the situation there (giving a semantics for the simply typed λ calculus with a type of Booleans) is that OpenMath allows n-ary function application (rather than binary) arbitrary binding symbols (rather than just λ-abstraction), and arbitrary attributions (rather than just simple types), but only assumes α-conversion (rather than $\alpha\beta\eta$ conversion).

[1]We do not cover the roles `attribution` and `error` here, which go beyond the fragment of OpenMath we consider; their treatment is analogous. Moreover, we write `key` instead of `semantic − attribution` for brevity.

3.1. Syntax

We start out by fixing an abstract syntax of “OM objects”, which we will relate to OpenMath objects in Section 3.3. We will call the objects specified in Definition 9 “*abstract* OM Objects” when we want to distinguish from the “*standard* OpenMath objects” defined in the OpenMath2 standard [BCC+04, section 2].

Definition 1 (Symbols and Variables). In all of the following, we will assume the existence of two disjoint, countably infinite sets: a set *Symbols* of **symbols** and a set *Variables* of **variables**. Furthermore, we assume a set $\mathit{Keys} \subseteq \mathit{Symbols}$ of **keys**.[2]

As usual in formal languages we are a little more careful about the symbols and variables we use in the construction of complex objects. The notions of vocabularies and contexts help us do this:

Definition 2 (OM Vocabulary). An OM **vocabulary** is a set of symbols. For every OM vocabulary T, we denote by $\mathit{Symbols}(T) := \mathit{Symbols} \cap T$ the **set of symbols of** T and by $\mathit{Keys}(T) := \mathit{Keys} \cap T$ the **set of keys of** T.

Definition 3 (OM Context). An OM **context** C is an n-tuple of variables which we will write as $\langle x_1, \ldots, x_n\rangle$. We will use $+$ for tuple concatenation and $\in$ for tuple membership.

Definition 4 (OM Objects). Let T be an OM vocabulary. The set $O(T,C)$ of **OM objects** over T in context C is the smallest set closed under the following operations

1. if $s \in \mathit{Symbols}(T) \setminus \mathit{Keys}(T)$, then $\mathbb{S}(s) \in O(T,C)$,
2. if $x \in C$, then $\mathbb{V}(x) \in O(T,C)$,
3. if $f, o_1, \ldots, o_n \in O(T,C)$ for $n > 0$, then $\mathbb{A}(f, o_1, \ldots, o_n) \in O(T,C)$,
4. if $b \in O(T,C)$, $X_1, \ldots, X_n \in \mathit{AttVar}(T,C)$ for $n \geq 0$, and $o \in O(T,C')$ where $C' = C + \langle \mathit{varname}(X_1), \ldots, \mathit{varname}(X_n)\rangle$, then $\mathbb{B}(b, [X_1, \ldots, X_n], o) \in O(T,C)$,
5. if $o \in O(T,C)$, $k \in \mathit{Keys}(T)$, and $v \in O(T,C)$, then $\mathbb{K}(o|k := v) \in O(T,C)$.

Here **attributed variables** are defined by: $o \in \mathit{AttVar}(T,C)$, iff $o = \mathbb{V}(x)$ for some $x \in C$ or $o = \mathbb{K}(o'|k := v) \in O(T,C)$ for some $o' \in \mathit{AttVar}(T,C)$. We call OM objects in the empty context **ground**. The name of an attributed variable is defined by $\mathit{varname}(\mathbb{K}(o'|k := v)) = \mathit{varname}(o')$ and $\mathit{varname}(\mathbb{V}(x)) = x$.

Note that in contrast to the OpenMath2 standard we only consider “unary” attributions that associate an object with a single key/value pair. This allows us to build the “flattening of attributions” into the abstract representation of OM Objects. We can regain the syntactic structure of OpenMath2 objects by introducing n-ary attributions as an abbreviation for nested attributions: $\mathbb{K}(o|k_1 := v_1, \ldots, k_n := v_n) = \mathbb{K}(\mathbb{K}(o|k_1 := v_1)|k_2 := v_2, \ldots, k_n := v_n)$ for $n \geq 2$. With this

[2]This is assumption strictly for convenience in theory development; actually the determination of a symbol being a key is made by ascribing the role “semantic-attribution” in an OpenMath content dictionary. When we align content dictionaries with vocabularies in Definition 14, we make sure that the CD roles are respected.

trick[3] we have fully covered the requirement of "attribution flattening equivalence" required in the OpenMath standard.

Case 4 of Def. 9 reveals an underspecification in the OpenMath standard: The standard does not specify whether a bound variable may occur in the attributions of itself or of other variables bound by the same binder. Our definition requires $X_i \in \mathit{AttVar}(T, C)$, i.e., no variable may occur in any variable's attribution. While this a perfectly reasonable choice, others are possible. For example, $X_i \in \mathit{AttVar}(T, C + \langle \mathit{varname}(X_1), \ldots, \mathit{varname}(X_{i-1})\rangle)$ permits every variable to occur in attributions of later variables. That can be useful to merge nested binding objects all binding with the Π-binder of dependent type theory into a single binding object. $X_i \in \mathit{AttVar}(T, C')$ permits variables to occur in each other's attributions, which permits to represent mutually recursive let bindings.

In Case 4, one should also note that OpenMath permits bindings with 0 bound variables. These degenerate to unary functions.

Let us fortify our intuition with an example which will use throughout the paper; we focus on binding objects, since they are the most problematic case:

Example 1. The untyped universal quantification $\forall x.x = x$ is represented as $\mathbf{U} = \mathbb{B}(\mathbb{S}(\forall), [\mathbb{V}(x)], \boxed{x = x})$[4], where $\forall$ is a symbol. To show the interaction of attribution and binding, we use a typed identity function represented as a λ-abstraction: $\lambda x : \iota \to \iota.x$ is represented as $\mathbf{L} = \mathbb{B}(\mathbb{S}(\lambda), [\mathbb{K}(\mathbb{V}(x)|\tau := \boxed{\iota \to \iota})], \mathbb{V}(x))$. We have $\mathbf{U} \in O(\{\forall, =\}, \langle\rangle)$ and $\mathbf{L} \in O(T, \langle\rangle)$, where $T = \{\lambda, \tau, \iota, \to\}$ and $\mathit{Keys}(T) = \{\tau\}$.

The use of attributed variables in binders can lead to a somewhat awkward notations when accessing the keys and attributions present in abstract binding objects. Therefore, we use the auxiliary definition of binding signatures in the technical developments below. Intuitively, an OM binding object has binding signature σ if it binds $l(\sigma)$ variables where the i-th variable has $d^i(\sigma)$ attributions.

Definition 5 (Binding Signature). A **binding signature** σ consists of

- a natural number $l(\sigma)$ (the **length** of σ),
- natural numbers $d^1(\sigma)$, …, $d^n(\sigma)$ (the **depth of** σ **at** i).

We denote by $\overline{\sigma}$ the set of pairs $\langle i, j\rangle \in \mathbb{N}\times\mathbb{N}$ where $1 \leq i \leq l(\sigma)$ and $1 \leq j \leq d^i(\sigma)$.

Definition 6 (Abbreviated Binding Notation). If σ is a binding signature with length n, $b \in O(T, C)$, and $K : \overline{\sigma} \to \mathit{Keys}(T)$ and $V : \overline{\sigma} \to O(T, C)$, as well as $o \in O(T, C + \langle x_1, \ldots, x_n\rangle)$, then we write

$$\mathbb{B}(b\,[x_1, \ldots, x_n|K := V].o) \quad \text{for} \quad \mathbb{B}(b, [X_1, \ldots, X_n], o) \in O(T, C)$$

where $X_i = \mathbb{K}(\mathbb{V}(x_i)|K(i, 1) := V(i, 1), \ldots, K(i, d^i(\sigma)) := V(i, d^i(\sigma)))$.

[3]In fact we propose to follow this path in the next version of the OpenMath standard as it simplifies the presentation. Note that we are only talking about (standard) OpenMath objects, not their XML or binary encodings, where n-ary attributions make sense for notational convenience.

[4]Here and throughout the paper we will use boxed mathematical formulae to gloss OpenMath objects (encoded, abstract, or standard); we assume that this distinction is either meaningless or clear from the context. Here, $\boxed{x = x}$ stands for $\mathbb{A}(\mathbb{S}(=), \mathbb{V}(x), \mathbb{V}(x))$.

Example 2 (Continuing Example 1). In the abbreviated syntax $\forall x.x = x$ is represented as $\mathbf{U} := \mathbb{B}(\mathbb{S}(\forall)\,[x|\varnothing := \varnothing].\boxed{x = x})$ and $\lambda x : \iota \to \iota.x$ as $\mathbf{L} := \mathbb{B}(\mathbb{S}(\lambda)\,[x|K := V].\mathbb{V}(x))$, where in the latter case

- $l(\sigma) = 1$ and $d^1(\sigma) = 1$, and therefore $\overline{\sigma} = \{\langle 1, 1\rangle\}$
- $K = \{\langle 1, 1\rangle \mapsto \tau\}$ and $V = \{\langle 1, 1\rangle \mapsto \boxed{\iota \to \iota}\}$

Clearly, every OM object of the form $\mathbb{B}(b, [X_1, \dots, X_n], o)$ can be written uniquely as an expression of the form $\mathbb{B}(b\,[x_1, \dots, x_n|K := V].o)$, and we will use the latter notation in the future and abbreviate $\mathbb{B}(b\,[x_1, \dots, x_n|\varnothing := \varnothing].o)$ with $\mathbb{B}(b\,[x_1, \dots, x_n].o)$.

Definition 7 (Substitution). For $o \in O(T, \langle x_1, \dots, x_n\rangle)$, we denote by $\mathit{Subs}(o)$ the function that maps $\langle o_1, \dots, o_n\rangle$ to the object arising from o by substituting all free occurrences of x_i with o_i.

Because the definition of substitution application is straightforward and well-known, we omit it, and only mention one technical detail regarding the shadowing of bound variables: In the degenerate case of a binding $\mathbb{B}(b\,[x_1, \dots, x_n|K := V].o)$ with $x_i = x_j$ for some $i < j$, the OpenMath standard defines that x_i is shadowed by x_j, i.e., free occurrences of $x_i = x_j$ in o refer to x_j.

Definition 8 (α-Equality). Two objects are said to be **α-equal** iff they arise from one another by renaming bound variables. $\equiv_\alpha$ denotes the induced equivalence relation, and $[o]_\alpha$ denotes the equivalence class of o.

Finally, we define the head of an OM object as follows:

Definition 9 (OM Objects). The **head** of an OM object o is

- o if o is a symbol or variable,
- f if $o = \mathbb{A}(f, o_1, \dots, o_n)$,
- b if $o = \mathbb{B}(b, [X_1, \dots, X_n], o')$,
- k if $o = \mathbb{K}(o'|k := v)$.

3.2. Semantics

In the following, we will use use the notation $\Lambda x \in A.f(x)$ for the set-theoretical function defined by $\{\langle x, f(x)\rangle : x \in A\}$. A may be omitted if it is clear from the context. We also write B^A for the set of functions from A to B.

Definition 10 (OM Algebra). Let T be an OM vocabulary. An **OM algebra** A over T consists of

1. a set $U := U^A$ called the **universe of discourse**
2. a family of sets $R^A_n \subseteq U^{(U^n)}$ for $n \geq 1$; we also define $R^A_0 = U$,
3. an element $s^A \in U$ for every $s \in \mathit{Symbols}(T) \setminus \mathit{Keys}(T)$,
4. a family of mappings $@^A_n : U \times U^n \to U$ for $n \geq 1$,
5. a family of mappings $\beta^A_K : U \times U^{\overline{\sigma}} \times R^A_{l(\sigma)} \to U$ for every binding signature σ and mapping $K : \overline{\sigma} \to \mathit{Keys}(T)$,
6. a family of mappings $\alpha^A_k : U \times U \to U$ for every $k \in \mathit{Keys}(T)$.

s^A interprets the symbols intra-universally, and $@^A$, β^A, and α^A are extra-universal operations that yield a construction-oriented interpretation function. The sets R_n^A are special. Because OpenMath permits arbitrary expressions as binders, it is not possible to define the interpretation of every binder separately as is common in both first-order and higher-order settings. Instead, we need to model variable binding explicitly in the semantics. Syntactically, binders are operators that take terms with free variables as arguments. It is well-understood in higher-order logic and type theory that terms with n free variables can be modeled as n-ary functions on the universe. Thus, we interpret binders as operators taking functions as arguments. These come from the $R_{l(\sigma)}^A$ in the third argument of β operator (note that $l(\sigma) = n$ here). The functions from $\overline{\sigma}$ to U^A in the second argument are used for dealing with the keys of the attributed variables.

Since we can always write a binder like $\mathbb{B}(b\,[x].\mathbb{V}(x))$, the set R_1^A should at least contain the identity function. However, putting $R_n = U^{(U^n)}$ would be too big in general because only some of these functions actually arise from the interpretation of terms with free variables. Since the interpretation of these terms depends on A itself, we permit an arbitrary set R_n^A here and leave it to Def. 12 to sort out when an OM algebra is well-defined.

Definition 11 (Assignment). Let A be an OM Algebra over T, and let C be an OM context with n variables. An A-**assignment** φ for C is a tuple in $(U^A)^n$. We denote the assignment $\langle \varphi_1, \ldots, \varphi_n, u\rangle$ for $C + \langle x\rangle$ by φ, u.

Definition 12 (Interpretation). Let A be an OM Algebra over T, and let φ be an A-assignment for a context C. The **interpretation** $[\![o]\!]_\varphi^A$ of $o \in O(T, C)$ in A under φ is defined as follows:

1. $[\![\mathbb{S}(s)]\!]_\varphi^A = s^A$,
2. $[\![\mathbb{V}(x_i)]\!]_\varphi^A = \varphi_i$,
3. $[\![\mathbb{A}(f, o_1, \ldots, o_n)]\!]_\varphi^A = @_n^A([\![f]\!]_\varphi^A, \langle [\![o_1]\!]_\varphi^A, \ldots, [\![o_n]\!]_\varphi^A\rangle)$,
4. $[\![\mathbb{B}(b\,[x_1, \ldots, x_n|K := V].o)]\!]_\varphi^A = \beta_K^A([\![b]\!]_\varphi^A, \mathcal{V}, \mathcal{F})$ where
   (a) σ is the binding signature of the binding (which must have length n),
   (b) $\mathcal{V} = \Lambda p \in \overline{\sigma}.[\![V(p)]\!]_\varphi^A$,
   (c) $\mathcal{F} = \Lambda u \in (U^A)^n.[\![o]\!]_{\varphi,u_1,\ldots,u_n}^A$
5. $[\![\mathbb{K}(o|k := v)]\!]_\varphi^A = \alpha_k^A([\![o]\!]_\varphi^A, [\![v]\!]_\varphi^A)$.

Whether the case for bindings is well-defined, depends on the sets R_n^A. We call A **well-defined** if $\Lambda u \in (U^A)^n.[\![o]\!]_{\varphi,u_1,\ldots,u_n}^A \in R_n^A$ for all C, n, $o \in O(T, C)$, and all assignments φ for C.

Example 3 (Continuing Example 2). To interpret **U** we use an OM Algebra A with

1. $U^A := \mathbb{N} \cup \{\mathsf{q}, \mathsf{e}, \mathsf{t}, \mathsf{f}, \bot\}$
2. $R_n^A = U^{(U^n)}$,
3. $\forall^A := \mathsf{q}$ and $=^A := \mathsf{e}$,

4. $@_2^A(\mathsf{e}, u, v) = \mathsf{t}$ if $u = v$; $@_2^A(\mathsf{e}, u, v) = \mathsf{f}$ if $u \neq v$; and $@_n^A(u, \langle u_1, \ldots, u_n\rangle) = \bot$ otherwise.
5. $\beta_\varnothing^A(\mathsf{q}, \varnothing, \mathcal{F}) = \mathsf{t}$ if $\mathcal{F}(u) = \mathsf{t}$ for all $u \in \mathbb{N}$; $\beta_\varnothing^A(\mathsf{q}, \varnothing, \mathcal{F}) = \mathsf{f}$ if $\mathcal{F}(u) = \mathsf{f}$ for some $u \in \mathbb{N}$; and $\beta_K^A(u, \langle x_1, \ldots, x_n\rangle, \mathcal{F}) = \bot$ otherwise.

Note that we only specify the parts of the algebra we actually need for our example, all others can be picked arbitrarily. If we want to evaluate $\boxed{\forall x.x = x}$ in A, recall that $\overline{\sigma} = \varnothing$ and thus $\mathcal{V} = \Lambda p \in \overline{\sigma}.[\![\varnothing(p)]\!]_\varnothing^A = \varnothing$, so we have

$$[\![\mathbf{U}]\!]_\varnothing^A = [\![\mathbb{B}(\mathbb{S}(\forall)\, [x].\boxed{x = x})]\!]_\varnothing^A = \beta_\varnothing^A(\mathsf{q}, \varnothing, \mathcal{F})$$

where $\mathcal{F} = \Lambda u \in U^A.[\![\boxed{x = x}]\!]_{(u)}^A$. So $[\![\mathbf{U}]\!]_\varnothing^A = \mathsf{t}$, iff $\mathcal{F}(u) = \mathsf{t}$ for all $u \in \mathbb{N}$. But observe that we have $\mathcal{F}(u) = [\![\mathbb{A}(=, \mathbb{V}(x), \mathbb{V}(x))]\!]_{(u)}^A = @_2^A(\mathsf{e}, \langle u, u\rangle) = \mathsf{t}$ by definition, and thus $[\![\mathbf{U}]\!]_\varnothing^A = \mathsf{t}$ as expected.

Extending A to an interpretation of the λ-binder is more complicated because we have to commit to a type theory.

Example 4 (Continuing Example 2). We extend U^A so that it contains all function sets that can be formed from the natural numbers, i.e., $\mathbb{N}^\mathbb{N}$ $\mathbb{N}^{(\mathbb{N}^\mathbb{N})}$, $(\mathbb{N}^\mathbb{N})^\mathbb{N}$ and so on, as well as the functions they contain. We call this set $\mathbb{N}^{**}$. For this to be useful, we should also extend our vocabulary with symbols ι and $\rightarrow$. We put

1. $U := \mathbb{N}^{**} \cup \{\mathsf{l}, \mathsf{p}, \bot\}$
2. $R_n^A = U^{(U^n)}$,
3. $\lambda^A = \mathsf{l}$, $\iota^A = \mathbb{N}$, and $\rightarrow^A = \mathsf{p}$, and
4. interpret $@_2^A(\mathsf{p}, \langle u, v\rangle)$ as the set of functions from v to u if u and v are sets and as $\bot$ otherwise. Furthermore, we put $@_1^A(f, \langle u\rangle) = f(u)$ whenever function application is defined. We put $@^A(f, \langle u_1, \ldots, u_n\rangle) = \bot$ otherwise.
5. Then for $\overline{\sigma} = \{\langle 1, 1\rangle\}$, $K = \{\langle 1, 1\rangle \mapsto \tau\}$, we can put $\beta_K^A(\mathsf{l}, \mathcal{V}, \mathcal{F})$ to be the function $\Lambda u \in \mathcal{V}(\langle 1, 1\rangle).\mathcal{F}(u)$. We put $\beta_L^A(u, \mathcal{V}, \mathcal{F}) = \bot$ in all other cases.
6. $\alpha_\tau^A(u, v) = u$.

Then we can interpret $\boxed{\lambda x : \iota \rightarrow \iota.x}$ as follows. We have $[\![\mathbf{L}]\!]_\varnothing^A = [\![\mathbb{B}(\mathbb{S}(\lambda)\, [x|K := V].\mathbb{V}(x))]\!]_\varnothing^A = \beta_K^A(\mathsf{l}, \mathcal{V}, \mathcal{F})$ where

- $\mathcal{V} = \Lambda p \in \{\langle 1, 1\rangle\}.[\![V(p)]\!]_\varnothing^A = \Lambda p \in \{\langle 1, 1\rangle\}.[\![\boxed{\iota \rightarrow \iota}]\!]_\varnothing^A = \{\langle 1, 1\rangle \mapsto \mathbb{N}^\mathbb{N}\}$,
- $\mathcal{F} = \Lambda u \in U.[\![\mathbb{V}(x)]\!]_{(u)}^A = \Lambda u \in U.u$

And thus, we evaluate $\beta_K^A(\mathsf{l}, \mathcal{V}, \mathcal{F})$ as the identity function on $\mathbb{N}^\mathbb{N}$ as expected.

A simple induction over the construction of OpenMath objects in Definition 9 using the respective clauses in Definition 12 gives us an OpenMath version of the well-known

Lemma 1 (Substitution Value Lemma). *If $o \in O(T, C + \langle x\rangle)$ and $o' \in O(T, C)$, then $[\![[x/o']o]\!]_\varphi^A = [\![o]\!]_{\varphi, [\![o']\!]_\varphi^A}^A$*

This in turn can be specialized in the usual way to obtain:

Corollary 1 (Soundness of α-Equality). *If $o \equiv_\alpha o'$ then $[\![o]\!]_\varphi^A = [\![o']\!]_\varphi^A$.*

So we have shown that OM algebras form a model class for OpenMath objects. We will now show that they characterize them up to isomorphism. For that we need to consider initial models, which will function as canonical representatives in this model class.

Definition 13 (Free OM Algebra). Let T be an OM vocabulary. Then the **free OM algebra** $I := I(T)$ **over** T is defined as follows.

1. $U^I = O(T, \varnothing)/_{\equiv_\alpha}$, i.e. the quotient set of the ground OpenMath objects modulo α-conversion.
2. R^I_n is the set of functions $\overline{Subs(o)}$ for $o \in O(T, \langle x_1, \dots, x_n \rangle)$, which are defined as follows: $\overline{Subs(o)}(\langle [o_1]_\alpha, \dots, [o_n]_\alpha \rangle) = [Subs(o)\langle o_1, \dots, o_n \rangle]_\alpha$.
3. $s^I = [\mathbb{S}(s)]_\alpha$,
4. $@^I_n([f]_\alpha, \langle [o_1]_\alpha, \dots, [o_n]_\alpha \rangle) = [\mathbb{A}(f, o_1, \dots, o_n)]_\alpha$,
5. for a binding signature σ: $\beta^I_K([b]_\alpha, \mathcal{V}, \mathcal{F}) = [\mathbb{B}(b\,[x_1, \dots, x_n | K := V].o)]_\alpha$ where
   - $V = \Lambda p \in \overline{\sigma}.v_p$ for some $v_p \in \mathcal{V}(p)$,
   - $o \in O(T, \langle x_1, \dots, x_n \rangle)$ is some object such that $\overline{Subs(o)} = \mathcal{F}$.
6. $\alpha^I_k([o]_\alpha, [v]_\alpha) = [\mathbb{K}(o | k := v)]_\alpha$.

Lemma 2. *$I(T)$ is well-defined.*

Proof. We need to show several well-definedness conditions.

R^A_n: $\overline{Subs(o)}$ is well-defined because substituting ground α-equivalent objects for the free variables of o yields α-equivalent objects.

$@^I_n$: If $f \equiv_\alpha f'$, $o_1 \equiv_\alpha o'_1, \dots, o_n \equiv_\alpha o'_n$, then $\mathbb{A}(f, o_1, \dots, o_n) \equiv_\alpha \mathbb{A}(f', o'_1, \dots, o'_n)$. This follows directly from the definition of α-equivalence.

β^I_K: If $b \equiv_\alpha b'$, $v_p \equiv_\alpha v'_p$ for all $p \in \overline{\sigma}$, then there exists an $o \in O(T, \langle x_1, \dots, x_n \rangle)$ such that $\overline{Subs(o)} = \mathcal{F}$, and for two such o, o' we have that

$$\mathbb{B}(b\,[x_1, \dots, x_n | K := \Lambda p.v_p].o) \equiv_\alpha \mathbb{B}(b'\,[x_1, \dots, x_n | K := \Lambda p.v'_p].o').$$

The existence follows from the definition of R^I_n. The α-equivalence holds because non-α-equivalent objects induce non-α-equivalent substitution functions.

α^I_k: If $o \equiv_\alpha o'$ and $v \equiv_\alpha v'$, then $\mathbb{K}(o | k := v) \equiv_\alpha \mathbb{K}(o' | k := v')$. This follows directly from the definition of α-equivalence.

□

Lemma 3. *Let T be an OM algebra. Then $[\![o]\!]^{I(T)} = [o]_\alpha$ for every $o \in O(T, \langle\rangle)$.*

Proof. This is proved by a straightforward induction on the structure of o. □

Lemma 4 ($I(T)$ is initial). *Let A be an OM algebra over T, and let $I := I(T)$. Then there is a unique mapping $h : U^I \to U^A$ satisfying $h([\![o]\!]^I) = [\![o]\!]^A$ for all $o \in O(T, \langle\rangle)$.*

Proof. h maps $[o]_\alpha \in U^I$ to $[\![o]\!]^A$. The needed property follows directly from Cor. 1 and Lem. 3. □

Corollary 2 (Completeness of α-Equality). *If $o, o' \in O(T, \langle\rangle)$ and $[\![o]\!]^A = [\![o']\!]^A$ for all OM algebras A, then $o \equiv_\alpha o'$.*

Proof. This follows from Lem. 3 by putting $A := I(T)$. □

In the classification of denotational semantics from Section 2.1 our semantics from Def. 10 is construction-oriented using the four functions s^A, $@_n^A$, β_K^A, and α_k^A for the four constructions: all of them come with the OM algebra, not the model class. As we have predicted in Section 2.1 the semantics is rather involved. In particular, giving individual OM algebras is ludicrously complicated due to the functions β_K^A: These must provide an interpretation for *any* object used as a binder, binding *any* number of bound variables, in which each variable carries *any* list of attribution keys, each attributing *any* value. This convoluted semantics is unavoidable and enforced by the generality of the OpenMath standard; for a more elegant semantics of a slightly restricted subset of OpenMath objects, see Section 5.

3.3. OpenMath Objects with Uninterpreted Symbols

The semantics discussed so far was based on the abstract notion of OM Vocabularies. To arrive at a semantics of OpenMath objects we need to relate this to OpenMath CDs.

The OpenMath2 standard introduces "abstract content dictionaries" to abstract from the concrete XML encoding of content dictionaries. According to [BCC+04, section 4.2], (abstract) CDs have a **CD name**, a **CD base URI**, and contain **symbol definitions**, which in turn consist (among others) of a **symbol name**, an optional **symbol role** (one of "binder", "attribution", "semantic-attribution", "application", "constant", and "error"), and a set of **mathematical properties**.

Definition 14 (OpenMath Symbols). We say that a CD C **declares** an OpenMath symbol $\langle n, c, u, r\rangle$, iff the CD base of C is u, the CD name of C is c, and C has a symbol definition with symbol name n and symbol role r (note that the role can be undefined as it is optional). We define the set *Symbols* to be the set of symbols declared by some OpenMath CD and the set *Keys* to be those with symbol role "semantic-attribution".

There are three differences between abstract OM Objects and standard OpenMath objects; all three are related to symbols and keys:

1. We do not take keys to be abstract OM objects by themselves (see clause 1 in Definition 9). We claim that that there are no mathematically meaningful situations where keys can appear except in attributions. This design decision should not be perceived as a serious impediment for our semantics, since keys can be added analogously to the treatment below at the cost of adding an additional case everywhere.
2. The OpenMath2 [BCC+04] "role system", poses some additional restrictions on where symbols can occur, but not enough to simplify our construction

of binding signatures. Therefore, we disregard it here and refer the reader to [RK09] for details and an extended role system proposal that would.

3. We do not consider attributions with symbols that are not in *Keys*, in particular symbols with roles "attribution" which are intended by the OpenMath2 standard for just this purpose. However the standard states

   > *This form of attribution may be ignored by an application, so should be used for information which does not change the meaning of the attributed OpenMath object.* [BCC⁺04, clause 2.1.4.*ii*]

   and therefore it is necessary to disregard these attributions in the construction of a semantics for OpenMath. In the mapping from standard OpenMath objects to abstract ones, we strip attributions with non-*Keys* symbols.

This allows us to define the meaning of an OpenMath object. As we are not taking mathematical properties in CDs into account, we will think of these symbols as uninterpreted, therefore we will call it the "algebraic meaning".

Definition 15 (Algebraic Meaning). Let o be an OpenMath object, then we call the set of symbols such that $\mathbb{S}(s)$ occurs in o the **OM vocabulary induced by** o.

If o is a ground OM Object, T its induced vocabulary, and A an OM algebra over T, then the **algebraic meaning of** o **in** A is $[\![o]\!]^A$ and the **algebraic meaning of** o is $[\![o]\!]^{I(T)}$.

Note that the algebraic meaning of an abstract OpenMath object is just an α-equivalence class of (standard) OpenMath objects.

As discussed in the introduction, the algebraic semantics only gives us a rather weak and syntactic concept of meaning of the OpenMath language. To understand the full meaning of OpenMath objects we need to take CDs into account, which we do in the next section.

4. OpenMath Models

If we want to understand mathematical properties in OpenMath content dictionaries, we need to have a notion of "truth" — after all, the properties are assumed to hold true. Furthermore, we need to take into account the mathematical properties themselves. In OpenMath there are two kinds of mathematical properties: "commented mathematical properties" (encoded as `CMP` elements which contain mathematical vernacular) and "formal mathematical properties" (encoded as `FMP` elements that contain XML encodings of OpenMath objects). We are going to concentrate on the latter in this paper since they provide more structure. This is no loss of generality, given the assumption in mathematical practice that any rigorously stated property can be fully formalized given enough resources. For the purposes of this paper we will just assume that we have access to an oracle that translates all commented mathematical properties into formal ones, which we handle with the methods presented in this section.

4.1. Theories and Satisfaction

As formal mathematical properties are expressed as OpenMath objects, we will need to build the required notions of "truth" and equality into an OM vocabulary. This is rather simple.

Definition 16 (OM Logic). An OM vocabulary L with distinguished symbols $\top$ and $=$ is called an **OM logic**.

In OpenMath CDs, (formal) mathematical properties are expressed as statements in some foundational logical system, thus the OM Objects representing them will in general contain symbols from the foundation and the CD itself. For instance, the `arith1` CD [CDa04] contains an `FMP` with the object $\boxed{\forall a, b.a + b = b + a}$ to express commutativity of addition. The symbols $\boxed{\forall}$ and $\boxed{=}$ are from the vocabulary of the foundational system and the symbol $\boxed{+}$ is from the CD itself.

We will treat OpenMath content dictionaries as logical theories, which are determined by their vocabularies and axioms, and model them using institutions (see [Rab08] for an introduction to both).

Definition 17 (Theory). Let L be an OM logic and T an OM vocabulary. An **OM theory** Θ for L is a pair $\langle T, Ax\rangle$ where $Ax \subseteq O(L \cup T, \langle\rangle)$.[5] We will denote Ax with $Axioms(\Theta)$ and use $O(\Theta, C) := O(L \cup T, C)$ and take an OM algebra over Θ to be an OM algebra over $L \cup T$.

Note that $\langle\varnothing, \varnothing\rangle$ is a theory for any OM logic L, we call $\langle\varnothing, \varnothing\rangle$ the **empty theory over** L.

In this setting we can define OM models as those algebras that respect equality and in which the axioms hold.

Definition 18 (Model). Let L be an OM logic and Θ be an OM theory for L. An OM algebra M over Θ is a **model** of Θ if

- for all C, $o, o' \in O(T, C)$, and φ, we have that $[\![\mathbb{A}(=, o, o')]\!]^M_\varphi = [\![\top]\!]^M$ iff $[\![o]\!]^M_\varphi = [\![o']\!]^M_\varphi$,
- for all $A \in Axioms(\Theta)$, we have that $[\![A]\!]^M = [\![\top]\!]^M$.

The **Model Class** $\mathcal{M}(C)$ **of** Θ is the set of OM Models of Θ.

This gives us the OM versions of the standard notions of satisfaction and semantic entailment.

Definition 19 (Satisfaction). Let L be an OM logic, Θ be an OM theory for L, $o \in O(\Theta, C)$, M an OM model of Θ, and φ an assignment for C into M. Then we say that M **satisfies** o **under** φ (which we denote as $M, \varphi \models o$), iff $[\![o]\!]^M_\varphi = [\![\top]\!]^M_\varphi$. We write $M \models o$ if $M, \varphi \models o$ holds for all assignments φ and say that o is *valid* in M.

[5]There are content dictionaries with formal mathematical properties that are not ground. We follow common mathematical practice and assume they are implicitly closed universally.

Definition 20 (Entailment). Let Θ be an OM theory and o a ground object. Then we say that Θ **entails** o ($\Theta \models o$), iff $M \models o$ for all $M \in \mathcal{M}(\Theta)$.

Example 5 (Continuing Example 3). We can extend the vocabulary $\{\forall, =\}$ into an OM logic $Q = \{\forall, =, \top\}$. Then the OM algebra A from 3 becomes a model for the empty theory over Q by putting $\top^A := \mathtt{t}$. Note that $\mathbf{U}$ is entailed by the empty theory over Q.

We will now turn to the initial semantics again, this time to build initial OM models.

Definition 21 (Congruence Relation). Let T be a OM vocabulary and A an OM algebra over T. A **congruence relation on** A is a family of equivalence relations on U^A and R^A_n all denoted by $\equiv$ such that (whenever applicable)[6]

1. if $u \equiv u'$ and $u_i \equiv u'_i$ for $i = 1, \ldots, n$, then
$$@^A_n(u, \langle u_1, \ldots, u_n\rangle) \equiv @^A_n(u', \langle u'_1, \ldots, u'_n\rangle),$$
2. for $l(\sigma) = n$, if $u \equiv u'$, $\mathcal{V}(p) \equiv \mathcal{V}'(p)$ for all $p \in \overline{\sigma}$, and $\mathcal{F} \equiv \mathcal{F}'$, then
$$\beta^A_K(u, \mathcal{V}, \mathcal{F}) \equiv \beta^A_K(u', \mathcal{V}', \mathcal{F}'),$$
3. if $u \equiv u'$ and $v \equiv v'$, then $\alpha_k(u, v) \equiv \alpha_k(u', v')$,
4. if $\mathcal{F} \equiv \mathcal{F}'$ and $u_i \equiv u'_i$ for $i = 1, \ldots, n$, then
$$\mathcal{F}(\langle u_1, \ldots, u_n\rangle) \equiv \mathcal{F}'(\langle u'_1, \ldots, u'_n\rangle).$$

Definition 22 (Quotient Algebra). Let T be an OM vocabulary, A an OM algebra over T, and $\equiv$ a congruence relation on A. Then the OM algebra $Q := A/\equiv$ over T is defined by:

1. $U^Q = U^A/\equiv$,
2. R^Q_n is the set of all functions of the form
$$f : (U^Q)^n \to U^Q,\ f([u_1]_\equiv, \ldots, [u_n]_\equiv) = [F(u_1, \ldots, u_n)]_\equiv$$
for some $F \in R^A_n$,
3. $@^Q_n$, β^Q_K, and α^Q_k are induced by their analogues in A.

Lemma 5. *In the situation of Def. 22,*

- *Q is a well-defined OM algebra if A is,*
- *for all $o \in O(T, C)$ and all A-assignments $\varphi = (\varphi_1, \ldots, \varphi_n)$, it holds that $\left[\llbracket o\rrbracket^A_\varphi\right]_\equiv = \llbracket o\rrbracket^Q_{\varphi'}$ where φ' is the Q-assignment given by $\varphi'_i = [\varphi_i]_\equiv$ for $i = 1, \ldots, n$.*

Proof. We prove the lemma by induction on o and its context C. The first part of the lemma is proved in the induction step for binders.

- $\mathbb{S}(s)$: Trivial.
- $\mathbb{V}(x)$: Immediately from the relation between φ and φ'.

[6]Note that we do not have to consider a congruence on keys because equations between keys are not well-formed objects.

- $\mathbb{A}(f, o_1, \dots, o_n)$: Immediately from the definition of congruence.
- $\mathbb{K}(o|k := o')$: Immediately from the definition of congruence.
- $\mathbb{B}(b\,[x_1, \dots, x_n|K := V].o)$: If Q is well-defined, this follows immediately from the definition of congruence. To show well-definedness, we have to show that $f = \Lambda\langle v_1, \dots, v_n\rangle \in (U^Q)^n.[\![o]\!]^Q_{\varphi', v_1, \dots, v_n} \in R^Q_n$. Due to the well-definedness of A, we know that $F = \Lambda\langle u_1, \dots, u_n\rangle \in (U^A)^n.[\![o]\!]^A_{\varphi, u_1, \dots, u_n} \in R^A_n$. Thus, due to the definition of R^Q_n, we know that $f' : \langle [u_1]_\equiv, \dots, [u_n]_\equiv\rangle \mapsto [F(u_1, \dots, u_n)]_\equiv \in R^Q_n$. The induction hypothesis for o shows that $f = f'$.

□

Definition 23 (Induced Congruence). Let $\Theta = \langle T, Ax\rangle$ be an L-theory, then we define a congruence relation $\equiv_\Theta$ on $I(L \cup T)$ as follows:

$$[o]_\alpha \equiv_\Theta [o']_\alpha \qquad \text{iff} \qquad \Theta \models \mathbb{A}(=, o, o') \qquad \text{for } o, o' \in O(L \cup T, \langle\rangle)$$

and

$$\overline{Subs(o)} \equiv_\Theta \overline{Subs(o')} \qquad \text{iff} \qquad \Theta \models \mathbb{A}(=, o, o') \qquad \text{for } o, o' \in O(L \cup T, C).$$

We call $\equiv_\Theta$ the **congruence induced by** Θ.

Lemma 6. *Let $\Theta = \langle T, Ax\rangle$ be an L-theory, then $\equiv_\Theta$ is indeed a congruence relation.*

Proof. We need to show the properties of a congruence relation. All cases are straightforward. We prove the case of application as an example.

Assume $[o_i]_\alpha \equiv_\Theta [o'_i]_\alpha$ for $i = 0, \dots, n$. Then $\Theta \models \mathbb{A}(=, o_i, o'_i)$, and thus $[\![o_i]\!]^A = [\![o'_i]\!]^A$ in every Θ-model A. In that case, it follows that

$$@^A_n([\![o_0]\!]^A, \langle [\![o_1]\!]^A, \dots, [\![o_n]\!]^A\rangle) = @^A_n([\![o'_0]\!]^A, \langle [\![o'_1]\!]^A, \dots, [\![o'_n]\!]^A\rangle);$$

and therefore, $\Theta \models \mathbb{A}(=, \mathbb{A}(o_0, o_1, \dots, o_n), \mathbb{A}(o'_0, o'_1, \dots, o'_n))$; and therefore,

$$[\mathbb{A}(o_0, o_1, \dots, o_n)]_\alpha \equiv_\Theta [\mathbb{A}(o_0, o_1, \dots, o_n)]_\alpha.$$

Finally using Lem. 3 yields

$$@^{I(L\cup T)}_n([o]_\alpha, \langle [o_1]_\alpha, \dots, [o_n]_\alpha\rangle) \equiv_\Theta @^{I(L\cup T)}_n([o]_\alpha, \langle [o_1]_\alpha, \dots, [o_n]_\alpha\rangle)$$

as needed. □

As a consequence, the following construction is well-defined.

Definition 24 (Initial Model). Let $\Theta = \langle T, Ax\rangle$ be an L-theory, then $I(\Theta) := I(L \cup T)/\equiv_\Theta$ is called the **initial model for** Θ.

And that finally yields

Theorem 1 (Completeness). *For all $o \in O(\Theta, \langle\rangle)$ we have $I(\Theta) \models o$ iff $\Theta \models o$. In particular, $I(\Theta)$ is a Θ-model.*

Proof. We know $I(\Theta) \models o$ iff $[\![o]\!]^{I(\Theta)} = [\![\top]\!]^{I(\Theta)}$. Using Lem. 5, this is equivalent to $[[\![o]\!]^{I(T)}]_{\equiv_\Theta} = [[\![\top]\!]^{I(T)}]_{\equiv_\Theta}$ where T is the vocabulary of Θ. The latter is equivalent to $\Theta \models \mathbb{A}(=, o, \top)$ by Lem. 3 and Def. 23. And that is equivalent to $\Theta \models o$. □

From this result we directly obtain the main theorem of this section as a corollary:

Theorem 2 (Herbrand Theorem). *Every OM theory has a model that arises as a quotient of the free OM algebra.*

Note that our setup is a bit unusual in that we do not distinguish between consistent and inconsistent theories and obtain a model for every theory. This is because we only assume truth and equality in Def. 16 but not falsity and inequality. A more refined definition of logic could assume an additional symbol $\perp$ and require a model M to satisfy $[\![\top]\!]^M \neq [\![\perp]\!]^M$. Then there will be theories without models, and those theories are just the ones we intuitively think of as being inconsistent. The free OM-algebra over an inconsistent theory still exists, but its universe is a singleton. Thus, the above results would have to be restricted to consistent theories.

4.2. The Meaning of OpenMath CDs and Objects

Note that the definitions above are still abstract in the sense that they refer to OM vocabularies, and OM theories, and not OpenMath CDs. So as in section 3.3 we have to relate abstract OM objects to standard ones and in particular to answer the question: what is the theory of an OpenMath content dictionary? The OpenMath2 standard leaves this information under-defined, so we propose an interpretation that allows us to define an adequate notion of mathematical semantics[7].

Note that OpenMath CDs need not be self-contained, i.e. their `FMP`s can contain symbols that are neither introduced in the CD nor from the foundational system. Of course, these symbols (and thus the mathematical properties in CDs that introduce them) should have an effect on the meaning of the symbols described by the `FMP`, so they need to be taken into account; naturally this process must be iterated until fixed point has been reached.

Definition 25 (CD Import). Let C be an OpenMath content dictionary, then we say that C **imports** D, iff $C \neq D$ and some `FMP` element in C contains a symbol with CD D. We call a CD **basic**, iff it does not import other CDs.

In contrast to other module systems for mathematics (see [RK08, RK11] for an overview) OpenMath does not make make the "imports relation" explicit and in particular does not make any assumptions about the absence of cycles.

Definition 26 (Signature and Property set of a CD). The **signature** of a CD C is the set of symbols it declares in union with the signatures of all CDs imported by C. Similarly, the **property set** of a CD C is the set of OpenMath objects in `FMP` elements in C (these are called the **local properties** of C) in union with all the property sets of all CDs imported by C.

[7]Arguably the OpenMath standard cannot fix this fully, since it intends to support all mathematical software systems including such that are "semantics-independent" like mathematical editing systems.

With this, we can directly define the OM theory induced by a CD.

Definition 27 (Theory of a CD). We call the pair $\langle S, P\rangle$, where S is the signature of C and P is the property set of C the **OM theory of** C.

In essence, the OM theory of a content dictionary is the union of all symbol declarations and mathematical properties from all theories from which a symbol is used in the CD. There is no problem with the (implicit) imports being cyclic, since their morphisms (in the terminology of [RK11]) are the identity and we are constructing the (iterated) union. Note furthermore that OpenMath only supports literal CD names, and we can assume the set of CDs to be finite, therefore, the signature and axiom set of a CD are finite.

Note that in contrast to our definitions from section 4.1, the signature of a CD will already contain the OM logic, as OpenMath does not distinguish OM logics from other CDs. Following accepted mathematical practice we assume the logic to be first-order logic (with a choice operator) and a version of axiomatic set theory as a theory of first-order logic — we choose Zermelo-Fraenkel set theory with choice [Zer08, Fra22] since this is the best-known one. Note that any other foundation of mathematics would serve equally well for our purposes. For simplicity of presentation, we will assume the existence of two basic CDs for first-order logic (declaring connectives, quantifiers, equalities, and choice) and `ZFC` (declaring membership and axioms).

In OpenMath practice, commented mathematical properties seem to assume ZFC as a foundational system, whereas `FMP`s make due with less: they usually only use symbols from the CDS

- `logic1` [CDl04] supplies the symbol `true` — which we take as the distinguished symbol $\top$ — and the usual propositional connectives,
- and the symbol `eq` from CD `relation1` [CDr04] — which we take as $=$, together these two form a logic in the sense of Definition 16,
- `quant1` [CDq04] that supplies the first-order quantifiers.

Definition 27 allows us to define the meaning of a CD as a class of OM models.

Definition 28 (Model Class and Entailment for CDs). Let C be an OpenMath CD and Θ the OM theory of C, then the **Model Class of** C is $\mathcal{M}(\Theta)$ and $C \models o$, iff $\Theta \models o$.

Finally, Thm. 2 is exactly the bridging result between the OpenMath objects semantics postulated in the OpenMath2 standard (see Section 1.2) and the traditional foundations of mathematics (see section **??**). And with that we can finally define the meaning of OpenMath objects.

Definition 29 (The Meaning of an OpenMath Object). Let o be an OpenMath object, then we call the union of the theories of the CDs referenced in o the **Theory** of o. If $o \in O(T, \langle\rangle)$ is a ground OM Object, Θ its theory, and M an OM Model of Θ, then the **meaning of** o **in** M is $[\![o]\!]^M$.

5. A Symbol-Oriented Semantics for OpenMath

In this section, we will present a symbol-oriented semantics for a subset of OpenMath expressions defined by a novel role/type system that strengthens the one specified in the OpenMath 2 standard. We use a well-chosen trade-off in order to simplify the definition of algebras significantly while preserving the simplicity and flexibility of OpenMath.

Considering our construction-oriented semantics, we observe that from the four constructions of OpenMath (symbols, application, binding, and attribution) two (symbols and attributions) can be easily rephrased as a symbol-oriented semantics: Then s^A and α_k^A become the semantic entities assigned to a symbol $s \in \mathit{Symbols}(T) \setminus \mathit{Keys}(T)$ or $k \in \mathit{Keys}(T)$, respectively. Note that in the case of attributions, this is only possible because the OpenMath standard requires the head of an attribution to be a symbol.

For the heads of applications and bindings, on the other hand, OpenMath permits arbitrary complex objects. Consequently, the semantics must account for any object of the universe acting as a function or binder. Therefore, a separation between extra-universal functions/binders and intra-universal arguments/scopes is not possible.

For OpenMath applications, this cannot be remedied. Type systems have been used successfully for languages with function application, but these approaches only work because the syntactic form of functions is very restricted, in the easiest case all functions can be effectively normalized to a λ-abstraction. For a general purpose mathematical language like OpenMath, this is not the case: Functions can be obtained in a large variety of ways, and it is generally undecidable whether an object denotes a function.

The binding construction is an intermediate case. On the one hand, OpenMath permits arbitrary bindings. On the other hand, this freedom is rarely exploited: almost all binders used in practice are symbols, and each such symbol expects a specific list of attributions to the bound variables.

Therefore, our role system focuses on restricting the permitted binding objects. This leads to a symbol-oriented semantics with a single extra-universal application operator at a comparatively small price.

5.1. A Role System

We will now define a positive role system that provides a reasonable trade-off between being universal and being strict to prepare for a symbol-oriented semantics. Most characteristically, our role system generalizes the concept of functions with arities to binders.

The basic definitions are as follows:

Definition 30 (Arities). Consider a vocabulary T. An **application-arity** is a finite sequence of either $\_$ or $\_^+$, the latter occurring at most once. A **binding-arity** is a finite sequence of either K or K^+ for $K \subseteq \mathit{Keys}(T)$, the latter occurring at most once.

We will write sequences using the notation $[e_1, \ldots, e_n]$ and $E@e$ for the sequence E extended with the element e.

Definition 31 (Roles). A **role** is either `obj`, `key`, or a pair (a, b) of an application-arity a and a binding-arity b. A **roled vocabulary** (T, r) is vocabulary T together with a function r assigning a role to each symbol.

Intuitively, objects with role `obj` are the "usual" objects, i.e., values (including all functions) in the universe of discourse. These exclude the keys, which have role `key`, and the binders, which have the complex roles (a, b). In the latter roles, $\_$ stands for an argument, and $\_+$ for a finite non-empty sequence of arguments. Similarly, K stands for an attributed variable whose attribution keys are exactly the ones in K, and K^+ stands for a finite non-empty sequence of such attributed variables. We use the following auxiliary definition to avoid occurrences of $\_^+$ and K^+ sequences when possible:

Definition 32. Given an application or a binding-arity s of length m, and a natural number $n \geq m$, we define s^n as follows:

- if s does not contain $\_^+$ or K^+, then $s^n = s$,
- if s contains $\_+$ or K^+, respectively, then s^n arises from s by replacing $\_^+$ or K^+ with $n - m + 1$ repetitions of $\_$ or K, respectively.

Then we can make the meaning of binding arities more precise by:

Definition 33. An attributed variable X **matches** the set of keys $\{k_1, \ldots, k_n\}$ if it is of the form

$$X = \mathbb{K}(x|k_1 := v_1, \ldots, k_n := v_n)$$

for some ordering $k_1, \ldots, k_n$. In that case, we define

$$X(k_i) = v_i.$$

A sequence $X_1, \ldots, X_m$ of attributed variables **matches** the binding-arity b if $b^m = [b_1, \ldots, b_m]$ and each X_i matches b_i.

Example 6. Binders binding a single variable with a `type` attribution can be declared with the role $([], [\{\texttt{type}\}])$. More complex binders arise by combining application-arity and binding-arity such as in $([\_, \_], [\varnothing])$ for the definite integral $\int_a^b$, which takes two arguments and then binds one variable without attributions.

Then we can define the well-roled objects:

Definition 34. Assume a roled OM vocabulary (T, r). An object $o \in O(T, C)$ for some context C. We define the **well-roled** objects o and their roles $R(o)$. o is well-roled if one of the following holds:

1. $o = \mathbb{S}(s)$. In that case, $R(o) = r(s)$.
2. $o = \mathbb{V}(x)$ for $o \in C$. In that case, $R(o) =$ `obj`.
3. $o = \mathbb{A}(f, o_1, \ldots, o_n)$, $R(o_1) = \ldots = R(o_n) =$ `obj`, and
   (a) $R(f) =$ `obj`, in which case $R(o) =$ `obj`, or

   (b) $R(f) = (a, b)$ and the length of a is n or a contains $\_^+$, in which case $R(o) = ([], b)$.

4. $o = \mathbb{B}(o_1, [X_1, \ldots, X_n], o_2)$, $R(o_1) = ([], b)$, $X_1, \ldots, X_n$ matches b, all attribution values in all X_i have role $\texttt{obj}$, and $R(o_2) = \texttt{obj}$.
   In that case, $R(o) = \texttt{obj}$.
5. $o = \mathbb{K}(o'|k := v)$ and $R(k) = \texttt{key}$, and $R(o') = R(v) = \texttt{obj}$.
   In that case $R(o) = \texttt{obj}$.

Limitations. As expected, our role system is not quite universal. In the following, we discuss the limitations that our implicit or explicit assumptions impose when we restrict attention to well-roled objects. We assume that all unroled symbols are assigned the role `obj`.

Firstly, since our role system is positive, symbols with a role `key` or (a, b) may not occur in general OpenMath objects. The former may only occur as the head of an attribution. The latter may only occur as the head of a binding (if $a = []$) or as the head of an application that itself occurs as the head of a binding (if $a \neq []$). Conversely, only such objects may occur as the heads of attributions and bindings.

Secondly, since the keys of an attributed variable in a binding form a set, bound variables may not have two attributions with the same key, and the order of attributions does not matter. Moreover, the attributions on a bound variable must match the role of the binder.

For practical purposes, this means that all symbols intended to be used as binders or keys must be assigned a role and may then not be used for anything else. Moreover, for each binder, an a priori commitment is needed what attributions its bound variables will carry.

The latter limitation is the only one that may cause concern: It is conceivable that the same binder should be used with different binding-arities in different situations. For such purposes, a reasonable extension of our role system would be to permit optional keys, e.g., a binding-arity $\{\texttt{type}^?\}^+$ for arbitrarily many bound variables, which may carry a `type` attribution. Another possible generalization is to permit and then ignore attributions to a bound variable that are not required by the binder.

5.2. A Roled Semantics

For roled vocabularies, we can simplify the definition of OM algebras significantly. In particular, we can make it symbol-oriented. We will still use a monolithic universe, but we will interpret both binders and keys extra-universally.

First, we need some auxiliary definitions:

Definition 35. Given a set U, we define

- For a set $K = \{k_1, \ldots, k_n\}$ of keys, we call the elements of the set U^K K-**records** over U.

- For an application-arity a, the set U^a is given by

$$\begin{aligned} U^{[\_,\ldots,\_]} &= U \times \ldots \times U \\ U^{[\_,\ldots,\_,\_^+,\_,\ldots,\_]} &= U \times \ldots \times U \times \bigcup_{i=1}^{\infty} U^i \times U \times \ldots \times U \end{aligned}$$

- For a binding-arity b, the set U^b is given by

$$\begin{aligned} U^{[K_1,\ldots,K_n]} &= U^{K_1} \times \ldots \times U^{K_n} \\ U^{[K_1,\ldots,K_m,K^+,L_1,\ldots,L_n]} &= U^{K_1} \times \ldots \times U^{K_m} \times \bigcup_{i=1}^{\infty} U^K \times U^{L_1} \times \ldots \times U^{L_n} \end{aligned}$$

- Given an element $r \in U^b$, then $|r|$ is the number of records in r.

These definitions are used to capture the semantics of binders with binding-arity b. If $b = [K_1, \ldots, K_n]$ does not contain an occurrence of K^+, the binder expects n bound variables. Then the set U^b contains tuples $r = \langle r_1, \ldots, r_n \rangle$ such that r_i is a K_i-record providing an attributed value $r_i(k)$ for each key $k \in K_i$. In these cases, we always have $|r| = n$. If b contains K^+, then U^b contains tuples of arbitrary length corresponding to an arbitrary number of bound variables. Therefore, we use $|r|$ to obtain the actual number of records in a tuple r. Finally, pairs (r, u) where $r \in U^b$ and $f : U^{|r|} \to U$ provide a record of attributions and a function f on U in as many arguments as there are attributed variables.

Definition 36 (Applicative Structure). An **applicative structure** is a pair $(U, @)$ where U is a set and $@$ is a family of mappings $@_n : U \times U^n \to U$ for $n \geq 1$.

We still have to use the general application operator $@$ from Def. 10. This is necessary because we have to permit arbitrary objects with functional behavior. But apart from that, we can finally give a symbol-oriented semantics.

Given a universe U, the general structure of the semantics is as follows:

Syntactic role	Semantic domain
`obj`	U
`key`	$U \times U \to U$
(a, b)	$U^a \to (\{(r,u) \mid r \in U^b,\ f : U^{\lvert r \rvert} \to U\} \to U)$

In particular, binders and keys are interpreted in certain extra-universal domains.

Definition 37 (Roled Algebra). Let (T, r) be a roled OM vocabulary. A **roled OM algebra** A over (T, r) consists of

- an applicative structure $(U, @)$,
- for every $s \in T$ an element s^A of the domain corresponding to $r(s)$.

Definition 38 (Interpretation). Let A be a roled OM Algebra over (T, r), and let φ be an A-assignment for a context C. The **interpretation** $[\![o]\!]^A_\varphi$ of a well-roled $o \in O(T, C)$ in A under φ is an element of the domain corresponding to $R(o)$. It is defined by induction on o according to Def. 34:

1. $[\![\mathbb{S}(s)]\!]^A_\varphi = s^A$ (for any role $r(s)$),

2. $[\![\mathbb{V}(x_i)]\!]^A_\varphi = \varphi_i$,
3. (a) if $R(f) = \texttt{obj}$, then $[\![\mathbb{A}(f, o_1, \ldots, o_n)]\!]^A_\varphi = @^A_n([\![f]\!]^A_\varphi, \langle[\![o_1]\!]^A_\varphi, \ldots, [\![o_n]\!]^A_\varphi\rangle)$,
   (b) if $R(f) = (a, b)$, then $[\![\mathbb{A}(f, o_1, \ldots, o_n)]\!]^A_\varphi = f^A(\langle[\![o_1]\!]^A_\varphi, \ldots, [\![o_n]\!]^A_\varphi\rangle)$,
4. if $R(o_1) = ([], b)$ and $b^n = [K_1, \ldots, K_n]$, then $[\![\mathbb{B}(o_1, [X_1, \ldots, X_n], o_2)]\!]^A_\varphi = [\![o_1]\!]^A_\varphi(\langle V_1, \ldots, V_n\rangle, F)$ where
   - $V_i \in U^{K_i}$ with $V_i(k) = [\![X_i(k)]\!]^A_\varphi$ for $i = 1, \ldots, n$ and $k \in K_i$,
   - $F = \Lambda u \in U^n.[\![o_2]\!]^A_{\varphi, u_1, \ldots, u_n}$
5. $[\![\mathbb{K}(o|k := v)]\!]^A_\varphi = k^A([\![o]\!]^A_\varphi, [\![v]\!]^A_\varphi)$.

Finally, we have to show that roled algebras are indeed a special case of Def. 10:

Theorem 3. *Given a roled vocabulary (T, r), every roled algebra A over (T, r) induces an algebra B over T such that for all objects $o \in O(T, C)$ with $R(o) = \texttt{obj}$ and all assignments φ for C and A, we have $[\![o]\!]^A_\varphi = [\![o]\!]^B_\varphi$.*

Proof. Let U be the universe of A. The universe V of B is given by the closure of $U \cup \mathcal{P}(Keys(T))$ under the formation of n-ary tuples and functions. We put $R^B_n = U^{(V^n)}$.

Then we define the necessary construction-oriented interpretation in B in the straightforward way:

- s^B is given by s^A for $s \in Symbols(T)$,
- $@^B_n(f, u)$ is given
  - by $@^A_n(f, u)$ if $f \in U$, $u \in U^n$,
  - otherwise, by $f(u)$ if $f \in V \setminus U$ and $f(u) \in V$ is defined,
  - otherwise, by $\perp$,
- $\beta^B_K(u, v, w)$ is given
  - by $u(r, w)$ if $u(r, w) \in V$ is defined and where $r = (r_1, \ldots, r_n)$ is obtained from $v : \overline{\sigma} \to U$ and $K : \overline{\sigma} \to Keys(T)$ by $r_i(K(i, j)) = v(i, j)$,
  - otherwise, by $\perp$,
- $\alpha^B_k(u, v)$ is given by $k^A(u, v)$ for $k \in Keys(T)$.

It is straightforward to show the two interpretation functions agree for well-roled arguments. □

Here the construction of the universe V shows once again how much simpler algebras becomes if a role system is used.

6. Conclusion

In this paper we have tried to rectify common misunderstandings about the meaning of OpenMath (and thus MathML3) expressions. We have shown that the free algebra of OpenMath objects forms an initial algebra for "formulae with uninterpreted symbols" which is syntactic in nature as all initial algebras are. Indeed for OpenMath and content MathML expressions that do not contain symbols

— and are thus unrestricted by content dictionaries — this is the best meaning we can hope for: OpenMath cannot impose more restrictions than α-equivalence and flattening of attributions without losing coverage. This is captured by the the algebraic semantics of OpenMath expressions in Section 3.

But the meaning of an OpenMath object comes mainly from the mathematical properties in the content dictionaries of its symbols. In section 4 we have been able to show that this can be grafted onto the algebraic semantics by interpreting OpenMath CDs as logical theories over a foundational system like first-order logic with ZFC as an axiomatic set theory.

Note that our semantic analysis has only taken into account symbol names, roles, and mathematical properties. The former two are relevant for the OM vocabularies and the latter for the OM theories that give OpenMath symbols their meaning. In particular, we did not look at descriptions (for symbols or whole CDs) or examples. The status of these CD parts is left unspecified by the OpenMath2 standard, and usage in actual CDs is non-uniform. Symbol descriptions reach from appealing to the folklore — e.g. "*This symbol represents the Boolean value true.*" [CDl04] to specific literature references e.g. "*See CRC Standard Mathematical Tables and Formulae, editor: Dan Zwillinger, CRC Press Inc., 1996, (7.7.11) section 7.7.1.*" [CDs04]. Arguably both forms "mean" something to the human reader, and especially the latter should surely contribute to the theory. The case of examples in CDs is similarly unclear: if they were uninformative to the human reader, nobody would put them in. But again practice in published CDs is no help: examples are often statements — and thus in principle mathematical properties — about (mathematical objects constructed by) the symbols they illustrate, and — if they are — they tend to be valid, but it would be incautious to assume this to be generally the case or even a normative part of the CD. The next version of the OpenMath standard could of course clarify these issues at the cost of making it more heavyweight and thus arguably less useful. We propose to use the OMDoc format [Koh06] that already addresses these issues for specifying content dictionaries instead if the additional functionality is desired.

A final objection often brought up against the "semantics of OpenMath" is that the standard CDs maintained by the OpenMath society are very weak, and (even with the methods presented here) do not give a clear and unambiguous meaning for K-14 mathematics. Indeed this criticism is formally justified, but misses the main point of the OpenMath philosophy, namely that the set of CDs is open-ended, and that we can build CDs to suit all our communication and representation needs. In particular it is possible (and in fact rather simple) to build a CD `NatArith` for natural numbers and arithmetic by encoding the Peano Axioms and recursive equations for the arithmetical operators in OpenMath objects so that that its theory $\Theta = \Theta(\texttt{NatArith})$ determines the class of Θ-models up to isomorphism (and all are isomorphic to $\mathbb{N}$). To see this just use the standard proof with our notion of OM models from section 4.

The OpenMath society (and the W3C Math Working Group for that matter) view the weakness of the standard OpenMath/MathML CD group as a feature and not a bug. These CDs contain fewer mathematical properties to allow them to describe larger model classes. For instance the CD `arith1` [CDa04] (somewhat) corresponds to the class of (Abelian) semigroups. And that is a good thing, since it is intended to capture the informal usage in K-14: in many situations we need the flexibility offered by the OpenMath/MathML CDs so that we do not over-specify the meaning. We would probably not want to scare elementary school children who are struggling with long division with the Peano Axioms or teenagers in high school with the subtle differences between Riemann and Lebesque integration.

We end this treatise on the "meaning of OpenMath and MathML" with the observation that it is possible to specify the meaning of mathematical objects and formulae at many levels of flexibility and rigorousness and extend the invitation to our readers to do just that: to contribute content dictionaries to the community of mathematicians (by way of the OpenMath society CD site [OMC]).

References

[ABC$^+$03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3C recommendation, World Wide Web Consortium (W3C), 2003.

[ABC$^+$10] Ron Ausbrooks, Stephen Buswell, David Carlisle, Giorgi Chavchanidze, Stéphane Dalmas, Stan Devitt, Angel Diaz, Sam Dooley, Roger Hunter, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Paul Libbrecht, Bruce Miller, Robert Miner, Murray Sargent, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 3.0. W3C Recommendation, World Wide Web Consortium (W3C), 2010.

[AvLS98] John Abbott, Andre van Leeuwen, and Andreas Strotmann. Openmath: Communicating mathematical information between co-operating agents in a knowledge network. *Journal of Intelligent Systems*, 8, 1998.

[BBK04] Christoph Benzmüller, Chad Brown, and Michael Kohlhase. Higher order semantics and extensionality. *Journal of Symbolic Logic*, 69:1027–1088, 2004.

[BCC$^+$04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaëtano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The OpenMath Society, 2004.

[BF85] J. Barwise and S. Feferman, editors. *Model-Theoretic Logics*. Springer-Verlag, 1985.

[CDa04] `arith1`. Openmath content dictionary, 2004.

[CDl04] `logic1`. Openmath content dictionary, 2004.

[CDq04] `quant1`. Openmath content dictionary, 2004.

[CDr04] `relation1`. Openmath content dictionary, 2004.

[CDs04] `sdata1`. Openmath content dictionary, 2004.

[Chu40] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

[Dav99] James H. Davenport. A small OpenMath type system. Technical report, The OpenMath Esprit Project, 1999.

[Fra22] Adolf Abraham Fraenkel. Der Begriff "definit" und die Unabhängigkeit des Auswahlsaxioms. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse*, 1922. reprinted as [Fra67].

[Fra67] Adolf Abraham Fraenkel. The notion of "definite" and the independence of the axiom of choice. Source books in the history of the sciences series, pages 284–289. Harvard Univ. Press, Cambridge, MA, 3rd printing, 1997 edition, 1967.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, August 2006.

[LS86] J. Lambek and P. Scott. *Introduction to Higher-Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1986.

[OMC] OpenMath content dictionaries. web page at `http://www.openmath.org/cd/`. seen June2008.

[Rab08] Florian Rabe. *Representing Logics and Logic Translations*. PhD thesis, Jacobs University Bremen, 2008.

[RK08] Florian Rabe and Michael Kohlhase. An exchange format for modular knowledge. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev, and S. Schulz, editors, *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics*, number 418 in CEUR Workshop Proceedings, pages 50–68, Aachen, 2008.

[RK09] Florian Rabe and Michael Kohlhase. A better role system for OpenMath. In James H. Davenport, editor, *22nd OpenMath Workshop*, July 2009.

[RK11] Florian Rabe and Michael Kohlhase. A scalable module system. Manuscript, submitted to Information & Computation, 2011.

[Rob50] A. Robinson. On the application of symbolic logic to algebra. In *Proceedings of the International Congress of Mathematicians*, pages 686–694. American Mathematical Society, 1950.

[Str04] Andreas Strotmann. The categorial type of openmath objects. In Andrea Asperti, Grzegorz Bancerek, and Andrej Trybulec, editors, *Mathematical Knowledge Management, MKM'04*, number 3119 in LNAI, pages 378–392. Springer Verlag, 2004.

[TV56] A. Tarski and R. Vaught. Arithmetical extensions of relational systems. *Compositio Mathematica*, 13:81–102, 1956.

[WR13] A. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1913.

[Zer08] E. Zermelo. Untersuchungen ber die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65:261–281, 1908. English title: Investigations in the foundations of set theory I.

Michael Kohlhase
Computer Science, Jacobs University Bremen, Germany
`http://kwarc.info/kohlhase`

Florian Rabe
Computer Science, Jacobs University Bremen, Germany
`http://kwarc.info/frabe`