Monoid Theory in Alonzo: A Little Theories Formalization in Simple Type Theory

William M. Farmer
McMaster University
wmfarmer@mcmaster.ca

Dennis Y. Zvigelsky
McMaster University
yankovsd@mcmaster.ca

Abstract

Alonzo is a practice-oriented classical higher-order version of predicate logic that extends first-order logic and that admits undefined expressions. Named in honor of Alonzo Church, Alonzo is based on Church's type theory, Church's formulation of simple type theory. The *little theories method* is a method for formalizing mathematical knowledge as a *theory graph* consisting of *theories* as nodes and *theory morphisms* as directed edges. The development of a mathematical topic is done in the "little theory" in the theory graph that has the most convenient level of abstraction and the most convenient vocabulary, and then the definitions and theorems produced in the development are transported, as needed, to other theories via the theory morphisms in the theory graph.

The purpose of this paper is to illustrate how a body of mathematical knowledge can be formalized in Alonzo using the little theories method. This is done by formalizing *monoid theory* — the body of mathematical knowledge about monoids — in Alonzo. Instead of using the *standard approach to formal mathematics* in which mathematics is done with the help of a proof assistant and all details are formally proved and mechanically checked, we employ an *alternative approach* in which everything is done within a formal logic but proofs are not required to be fully formal. The standard approach focuses on *certification*, while this alternative approach focuses on *communication* and *accessibility*.

We would like to thank the referee for reviewing this paper and providing very helpful suggestions.

1 Introduction

Formal mathematics is mathematics done within a formal logic. *Formalization* is the act of expressing mathematical knowledge in a formal logic. One of the chief benefits of formal mathematics is that a body of mathematical knowledge can be formalized as a precise, rigorous, and highly organized structure. This structure records the logical relationships between mathematical concepts and facts, how these concepts and facts are expressed in a given theory, and how one theory is related to another. Since it is based on a formal logic, it can be developed and analyzed using software.

An attractive and powerful method for organizing mathematical knowledge is the *little theories method* [22]. A body of mathematical knowledge is represented in the form of a *theory graph* [38] consisting of *theories* as nodes and *theory morphisms* as directed edges. Each mathematical topic is developed in the "little theory" in the theory graph that has the most convenient level of abstraction and the most convenient vocabulary. Then the definitions and theorems produced in the development are transported, as needed, from this abstract theory to other, usually more concrete, theories in the graph via the theory morphisms in the graph.

The *standard approach to formal mathematics* focuses on *certification*: Mathematics is done with the help of a proof assistant and all details are formally proved and mechanically checked. We present in Section 2 an *alternative approach to formal mathematics*, first introduced in [21], that focuses on two other goals: *communication* and *accessibility*. The idea is that everything is done within a formal logic but proofs are not required to be fully formal and the entire development is optimized for communication and accessibility. We believe that formal mathematics can be made more useful, accessible, and natural to a wider range of mathematics practitioners — mathematicians, computing professionals, engineers, and scientists who use mathematics in their work — by pursuing this alternative approach.

The purpose of this paper is to illustrate how a body of mathematical knowledge can be formalized in Alonzo [20], a practice-oriented classical higher-order logic that extends first-order logic, using the little theories method and the alternative approach to formal mathematics. Named in honor of Alonzo Church, Alonzo is based on Church's type theory [8], Church's formulation of simple type theory [18], and is closely related to Peter Andrews' $\mathcal{Q}_0$ [1]; $\mathcal{Q}_0^{\mathrm{u}}$ [17], a version of $\mathcal{Q}_0$ with undefined expressions; and LUTINS [13, 14, 15], the logic of the IMPS proof assistant [23, 24]. Unlike traditional predicate logics, Alonzo admits partial functions and undefined expressions in accordance with the approach employed in mathematical practice that we call the *traditional approach to undefinedness* [16]. Since partial functions naturally arise from theory morphisms [15], the little theories method works best with a logic like Alonzo that supports partial functions.

Alonzo has a simple syntax with a *formal notation* for machines and a *compact notation* for humans that closely resembles the notation found in mathematical practice. The compact notation is defined by the extensive set of *notational definitions and conventions* given in [20]. Alonzo has two semantics, one for mathematics based on *standard models* and one for logic based on Henkin-style *general models* [32]. By virtue of its syntax and semantics, Alonzo is exceptionally well suited for expressing and reasoning about mathematical ideas and for specifying mathematical structures. A brief overview of the syntax and semantics of Alonzo is given in Section 3. See [20] for a full presentation of Alonzo.

We have chosen *monoid theory* — the concepts, properties, and facts about monoids — as a sample body of mathematical knowledge to formalize in Alonzo. A *monoid* is a mathematical structure consisting of a nonempty set, an associative binary function on the set, and a member of the set that is an identity element with respect to the function. Monoids are abundant in mathematics and computing. Single-object categories are monoids. Groups are monoids in which every element has an inverse. And several algebraic structures, such as rings, fields, Boolean algebras, and vector spaces, contain monoids as substructures.

Since a monoid is a significantly simpler algebraic structure than a group, monoid theory lacks the rich structure of group theory. We are formalizing monoid theory in Alonzo, instead of group theory, since it has just enough structure to adequately illustrate how a body of mathematical knowledge can be formalized in Alonzo. We will see that employing the little theories method in the formalization of monoid theory in Alonzo naturally leads to a robust theory graph.

Alonzo is equipped with a set of *mathematical knowledge modules* (*modules* for short) for constructing various kinds of mathematical knowledge units. For example, it has modules for constructing "theories" and "theory morphisms". A *language* (or *signature*) of Alonzo is a pair $L = (\mathcal{B}, \mathcal{C})$, where $\mathcal{B}$ is a finite set of base types and $\mathcal{C}$ is a set of constants, that specifies a set of expressions. A *theory* of Alonzo is a pair $T = (L, \Gamma)$ where L is a language called the *language of* T and Γ is a set of sentences of L called the *axioms of* T. And a *theory morphism* of Alonzo from a theory T_1 to a theory T_2 is a mapping of the expressions of T_1 to the expressions of T_2 such that (1) base types are mapped to types and closed quasitypes (expressions that denote sets of values), (2) constants are mapped to closed expressions of appropriate type, and (3) valid sentences are mapped to valid sentences.

Alonzo also has modules for constructing "developments" and "development morphisms". A *theory development* (or *development* for short) of Alonzo is a pair $D = (T, \Xi)$ where T is a theory and Ξ is a (possibly empty) sequence of definitions and theorems presented, respectively, as definition and theorem packages (see [20, Section 12.1]). T is called the *bottom theory* of D, and T', the extension of T ob-

tained by adding the definitions in Ξ to T, is called the *top theory* of D. We say that D is a *development of* T. A *development morphism* from a development D_1 to a development D_2 is a partial mapping from the expressions of D_1 to the expressions of D_2 that restricts to a theory morphism from the bottom theory of D_1 to the bottom theory of D_2 and that canonically extends to a theory morphism from the top theory of D_1 to the top theory of D_2 (see [20, Section 14.4.1]). Theories and theory morphisms are special cases of developments and development morphisms, respectively, since we identify a theory T with the trivial development $(T, [\,])$.

The modules for constructing developments and development morphisms provide the means to represent knowledge in the form of a *development graph*, a richer kind of theory graph, in which the nodes are developments and the directed edges are development morphisms. Alonzo includes modules for transporting definitions and theorems from one development to another via development morphisms. The design of Alonzo's module system is inspired by the IMPS implementation of the little theories method [22, 23, 24].

The formalization of monoid theory presented in this paper exemplifies an *alternative approach to formal mathematics*. We validate the definitions and theorems in a development using traditional (nonformal) mathematical proof. However, we extensively use the axioms, rules of inference, and metatheorems of $\mathfrak{A}$ — the formal proof system for Alonzo presented in [20] which is derived from Andrews' proof system for $\mathcal{Q}_0$ [1] — in these traditional proofs. The proofs are not included in the modules used to construct developments. Instead, they are given separately in Appendix A.

We produced the formalization of monoid theory with just a minimal amount of software support. We used the set of LaTeX macros and environments for Alonzo given in [19] plus a few macros created specifically for this paper. The macros are for presenting Alonzo types and expressions in both the formal and compact notations. The environments are for presenting Alonzo mathematical knowledge modules. The Alonzo modules are printed in brown color.

The overarching goal of this paper is to demonstrate that, using the little theories method and the alternative approach to formal mathematics, mathematical knowledge can be very effectively formalized in a version of simple type theory like Alonzo. Specifically, we want to show the following:

1. How the little theories method can be used to organize mathematical knowledge so that clarity is maximized and redundancy is minimized.

2. How formal libraries of mathematical knowledge that prioritize communication over certification can be built using the alternative approach to formal

mathematics with tools that are much simpler to learn and use than a proof assistant.

3. How Alonzo is exceptionally well suited for expressing and reasoning about mathematical ideas and for specifying mathematical structures in a direct and natural manner.

The paper is organized as follows. We present in Section 2 the alternative approach to formal mathematics and argue that this kind of approach can better serve the average mathematics practitioner than the standard approach. Section 3 gives a brief presentation of the syntax and semantics of Alonzo. Sections 4–11 present developments of theories of monoids, commutative monoids, transformation monoids, monoid actions, monoid homomorphisms, and monoids over real number arithmetic plus some supporting developments. These developments have been constructed to be illustrative; they are not intended to be complete in any sense. Sections 4–11 also present various development morphisms that are used to transport definitions and theorems from one development to another. Section 12 shows how our formalization of monoid theory can support a theory of strings. Related work is discussed in Section 13. The paper concludes in Section 14 with a summary and some final remarks. The definitions and theorems of the developments we have constructed are validated by traditional mathematical proofs presented in Appendix A. Appendix B contains some miscellaneous theorems needed for the proofs in Appendix A.

2 Alternative approach to formal mathematics

A *formal logic* (*logic* for short) is a *family of languages* such that:

1. The languages of the logic have a *common precise syntax*.

2. The languages of the logic have a *common precise semantics with a notion of logical consequence*.

3. There is a *sound formal proof system* for the logic in which proofs can be syntactically constructed.

Examples of formal logics for mathematics are the various versions of first-order logic, set theory, simple type theory, and dependent type theory.

There are five big benefits of formal mathematics, i.e., doing mathematics within a formal logic.

First, *mathematics can be done with greater rigor.* All mathematical ideas are expressed and reasoned about in a theory T of a formal logic. Mathematical concepts and statements are expressed as expressions and sentences of the language of T. All of these expressions and sentences have a precise, unambiguous meaning. The assumptions underlying the reasoning about the mathematical ideas are made explicit as axioms of the theory. The theorems of theory are precisely defined as the logical consequences of the axioms of the theory. And, finally, the theory is constructed so that it contains only the vocabulary and assumptions that are needed for the task at hand; irrelevant details are abstracted away.

Second, *conceptual errors can be systematically discovered.* In formal mathematics, all concepts and statements must be expressed in a language of a formal logic that has a precise semantics. The process of expressing mathematical ideas in a formal logic naturally leads to many conceptual errors being caught similarly to how type errors are caught in a modern programming language by type checking. Thus conceptual errors can be discovered systematically in formal mathematics in a way that is largely not possible in traditional mathematics. As a result, formal mathematics often yields a deeper understanding of the mathematics being explored than traditional mathematics.

Third, *mathematics can be done with software support.* Since the languages of a formal logic have a precise common syntax, the expressions and sentences of a language can be represented as data structures. The expressions and sentences can then be analyzed, manipulated, and processed via their representations as data structures. This, in turn, enables the study, discovery, communication, and certification of mathematics to be done with the aid of software. Since the languages also have a precise common semantics, there is a precise basis for verifying the correctness of this software.

Fourth, *results can be mechanically checked.* Formal proofs can be represented as data structures, and software can be used to check that one of these data structures represents an actual proof in the formal proof system of the logic. Software can also be used to help construct the formal proofs. Since the software needed to check the correctness of the formal proofs is often very simple and easily verified itself, it is possible to verify the correctness of the formal proofs with a very high level of assurance.

Fifth, *we can regard mathematical knowledge as a formal structure consisting of a network of interconnected theories.* A library of mathematical knowledge that represents this formal structure can be built by creating theories, defining new concepts, stating and proving theorems, and connecting one theory to another with theory morphisms that map the theorems of one theory to the theorems of another theory. The knowledge embodied in a structured library of this kind can be studied,

managed, searched, and presented using software.

The benefits of formal mathematics are huge. Greater rigor and discovering conceptual errors have been principal goals of mathematicians for thousands of years. Software support can greatly extend the reach and productivity of mathematics practitioners. Mechanically checked results can drive mathematics forward in areas where the ideas are poorly understood (often due to their novelty) or highly complex. And mathematical knowledge as a formal structure can enable the techniques and tools of mathematics and computing to be applied to mathematical knowledge itself.

The standard approach to formal mathematics, in which mathematics is done with the help of a proof assistant and all details are formally proved and mechanically checked, has three major strengths:

1. It achieves all five benefits of formal mathematics mentioned above.
2. All theorems are verified by machine-checked formal proofs. Thus there is a very high level of assurance that the results produced are correct.
3. There are several powerful proof assistants available, such as HOL [29], HOL Light [31], , Isabelle/HOL [48], Lean [10], Metamath/ZFC [39], Mizar [42], and Rocq (formerly Coq) [54], that support the approach.

It also has two important weaknesses:

1. It prioritizes certification over communication. For the average mathematics practitioner, communicating mathematical ideas is usually much more important than certifying mathematical results when the mathematics is well understood.
2. It is not accessible to the great majority of mathematics practitioners. Having to learn a strange logic and work with a complex proof assistant that utilizes unfamiliar ways of expressing and reasoning about mathematics is very often a bridge too far for the average mathematics practitioner.

We strongly believe, as an alternative to the standard approach, an approach to formal mathematics is needed that focuses on two goals, communication and accessibility, the weaknesses of the standard approach. To achieve these goals the alternative approach should satisfy the following requirements:

R1. *The underlying logic is fully formal and supports standard mathematical practice.* Supporting mathematical practice makes the logic easier to learn and use and makes formalization a more natural process.

R2. *Proofs can be traditional, formal, or a combination of the two.* This flexibility in how proofs are written enables proofs to be a vehicle for communication as well as certification.

R3. *There is support for organizing mathematical knowledge using the little theories method.* This enables mathematical knowledge to be formalized to maximize clarity and minimize redundancy.

R4. *There are several levels of supporting software.* The levels can range from just LaTeX support to a full proof assistant. The user can thus choose the level of software support they want to have and the level of investment in learning the software they want to make.

The alternative approach can achieve all five benefits of formal mathematics mentioned above, but it cannot achieve the same level of assurance as the standard approach that the results produced are correct. This is because the alternative approach prioritizes communication and accessibility over certification. Since most mathematics practitioners are usually more concerned about communication and accessibility than certification, the alternative approach is on average a better approach to formal mathematics than the standard approach. This is particularly true for applications that involve well-understood mathematics, the kind of mathematics that arises in mathematics education and routine applications. However, when the certification of results is the most important concern, the standard approach will often be a better choice than the alternative approach.

This paper employs an implementation of the alternative approach based on Alonzo that satisfies the first three requirements and partially satisfies the fourth requirement. Alonzo is a form of predicate logic, which is widely familiar to mathematics practitioners. Moreover, it supports the reasoning instruments that are most common in mathematical practice including functions, sets, tuples, and lists; mathematical structures; higher-order and restricted quantification; definite description; theories and theory morphisms; definitional and other kinds of conservative extensions; inductive sets; notational definitions and conventions, and undefined expressions. Thus Alonzo satisfies R1 as well or better than almost any other logic.

R2 is satisfied by our implementation of the alternative approach since proofs can be traditional or formal. Thus communication can be prioritized over certification in proofs when the mathematics is well understood. In this paper, all the proofs are traditional, but some make use of the axioms, rules of inference, and metatheorems of $\mathfrak{A}$, the proof system of Alonzo.

R3 is satisfied since Alonzo is equipped with a module system for organizing mathematical knowledge using the little theories method.

Our implementation of the alternative approach provides only the simplest level of software support: LaTeX macros for presenting Alonzo types and expressions and LaTeX environments for presenting Alonzo modules. Other levels of software support are possible; see the discussion in Chapter 16 of [20]. Alonzo has not been implemented in a proof assistant, but since it is closely related to LUTINS [13, 14, 15], the logic of the IMPS proof assistant [23, 24], it could be implemented in much the same way that LUTINS is implemented in IMPS. Thus R4 is only partially satisfied now, but it could be fully satisfied with the addition of more levels of software support.

The great majority of mathematics practitioners — including mathematicians — are much more interested in communicating mathematical ideas than in formally certifying mathematical results. Hence, the alternative approach — with support for standard mathematical practice, traditional proofs, the little theories method, and several levels of software — is likely to serve the needs of the average mathematics practitioner much better than the standard approach. This is especially true when the mathematical knowledge involved is well understood (such as monoid theory) and certification via traditional proof is adequate for the purpose at hand.

In summary, we believe that the alternative approach is not a replacement for the standard approach, but it would be more useful, accessible, and natural than the standard approach for the vast majority of mathematics practitioners.

3 Alonzo

Alonzo is fully presented in [20]. Due to space limitations, we cannot duplicate the entire presentation of Alonzo in this paper. Ideally, the reader should be familiar with the syntax and semantics of Alonzo presented in Chapters 4–7; the proof system for Alonzo presented in Chapter 8 and Appendices A–C; the tables of notational definitions found in Chapters 4, 6, 11, and 13; the notational conventions presented in Chapters 4 and 6; and the various kinds of (mathematical knowledge) modules of Alonzo presented in Chapters 9, 10, 12, and 14. However, we will give in this section a brief presentation of the syntax and semantics of Alonzo with most of the text taken from Chapters 4–6 of [20].

3.1 Syntax

The syntax of Alonzo consists of "types" that denote nonempty sets of values and "expressions" that either denote values (when they are defined) or denote nothing at all (when they are undefined). We present the syntax of Alonzo types and expressions with the compact notation, an "external" syntax intended for humans. The reader

is referred to [20] for the formal syntax, an "internal" syntax intended for machines. The compact notation for types and expressions is given below. Additional compact notation is introduced using *notational definitions* and *notational conventions*. A *notational definition* has the form

A stands for B,

where A and B are notations that present types or expressions; it defines A to be an alternate — and usually more compact, convenient, or standard — notation for presenting the type or expression that B presents. The meaning of A is the meaning of B. The notational definitions are given in tables with boxes surrounding the definitions, and the notational conventions are assigned names of the form "Notational Convention n".

Let $\mathcal{S}_{\mathsf{bt}}$, $\mathcal{S}_{\mathsf{var}}$, $\mathcal{S}_{\mathsf{con}}$ be fixed countably infinite sets of symbols that will serve as names of base types, variables, and constants, respectively. We assume that $\mathcal{S}_{\mathsf{bt}}$ contains the symbols $A, B, C \ldots, X, Y, Z$, etc., $\mathcal{S}_{\mathsf{var}}$ contains the symbols $a, b, c \ldots, x, y, z$, etc., and $\mathcal{S}_{\mathsf{con}}$ contains the symbols $A, B, C \ldots, X, Y, Z$, etc., numeric symbols, nonalphanumeric symbols, and words in lowercase sans sarif font.[1] We will employ the following syntactic variables for these symbols as well as types and expressions which are defined just below:

1. $\mathbf{a}, \mathbf{b}$, etc. range over $\mathcal{S}_{\mathsf{bt}}$.
2. $\mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{m}, \mathbf{n}, \mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z}$, etc. range over $\mathcal{S}_{\mathsf{var}}$.
3. $\mathbf{c}, \mathbf{d}$, etc. range over $\mathcal{S}_{\mathsf{con}}$.
4. $\alpha, \beta, \gamma, \delta$, etc. range over types.
5. $\mathbf{A}_\alpha, \mathbf{B}_\alpha, \mathbf{C}_\alpha, \ldots, \mathbf{X}_\alpha, \mathbf{Y}_\alpha, \mathbf{Z}_\alpha$, etc. range over expressions of type α.

A *type* of Alonzo is a string of symbols defined inductively by the following formation rules:

T1. *Type of truth values*: o is a type.

T2. *Base type*: $\mathbf{a}$ is a type.

T3. *Function type*: $(\alpha \to \beta)$ is a type.

T4. *Product type*: $(\alpha \times \beta)$ is a type.

[1] An expression like "u, v, w, etc." means the set of symbols that includes u, v, and w, and all possible annotated forms of u, v, and w such as u', v_1, and $\widetilde{w}$.

Let $\mathcal{T}$ denote the set of types of Alonzo. We assume $o \notin \mathcal{S}_{\mathsf{bt}}$.

When there is no loss of meaning, matching pairs of parentheses in the compact notation for types may be omitted (Notational Convention 1). We assume that function type formation associates to the right so that, e.g., a type of the form

$$(\alpha \to (\beta \to \gamma))$$

may be written more simply as $\alpha \to \beta \to \gamma$ (Notational Convention 2).

A type α denotes a nonempty set D_α of values. o denotes the set $D_o = \mathbb{B}$ of the Boolean (truth) values F and T. $(\alpha \to \beta)$ denotes some set $D_{\alpha\to\beta}$ of (partial and total) functions from D_α to D_β. $(\alpha \times \beta)$ denotes the Cartesian product $D_{\alpha\times\beta} = D_\alpha \times D_\beta$. We will use base types to denote the base domains of mathematical structures.

An *expression of type* α of Alonzo is a string of symbols defined inductively by the following formation rules:

E1. *Variable*: $(\mathbf{x} : \alpha)$ is an expression of type α.

E2. *Constant*: $\mathbf{c}_\alpha$ is an expression of type α.

E3. *Equality*: $(\mathbf{A}_\alpha = \mathbf{B}_\alpha)$ is an expression of type o.

E4. *Function application*: $(\mathbf{F}_{(\alpha\to\beta)}\, \mathbf{A}_\alpha)$ is an expression of type β.

E5. *Function abstraction*: $(\lambda\, \mathbf{x} : \alpha \,.\, \mathbf{B}_\beta)$ is an expression of type $(\alpha \to \beta)$.

E6. *Definite description*: $(\mathrm{I}\, \mathbf{x} : \alpha \,.\, \mathbf{A}_o)$ is an expression of type α where $\alpha \neq o$.

E7. *Ordered pair*: $(\mathbf{A}_\alpha, \mathbf{B}_\beta)$ is an expression of type $(\alpha \times \beta)$.

Let $\mathcal{E}$ denote the set of expressions of Alonzo. A *formula* is an expression of type o, and a *sentence* is a closed formula.

When there is no loss of meaning, matching pairs of parentheses in expressions may be omitted (Notational Convention 3). We assume that function application formation associates to the left so that, e.g., an expression of the form $((\mathbf{G}_{\alpha\to\beta\to\gamma}\, \mathbf{A}_\alpha)\, \mathbf{B}_\beta)$ may be written more simply as $\mathbf{G}_{\alpha\to\beta\to\gamma}\, \mathbf{A}_\alpha\, \mathbf{B}_\beta$ (Notational Convention 4). When the type α of a constant $\mathbf{c}_\alpha$ is known from the context of the constant, we will very often write the constant as simply $\mathbf{c}$ (Notational Convention 5). A variable $(\mathbf{x} : \alpha)$ occurring in the body $\mathbf{B}_\beta$ of $\lambda\, \mathbf{x} : \alpha \,.\, \mathbf{B}_\beta$ or in the body $\mathbf{A}_o$ of $\mathrm{I}\, \mathbf{x} : \alpha \,.\, \mathbf{A}_o$ may be written as just $\mathbf{x}$ if there is no resulting ambiguity (Notational Convention 6). So, for example, $\lambda\, \mathbf{x} : \alpha \,.\, (\mathbf{x} : \alpha)$ may be written more simply as $\lambda\, \mathbf{x} : \alpha \,.\, \mathbf{x}$. We will employ this convention for the other variable binders of Alonzo

introduced later by notational definitions (Notational Convention 7). A variable $(\mathbf{x} : \alpha)$ occurring in $\mathbf{B}_\beta$ may be written as just $\mathbf{x}$ if the type α is known from the context of the occurrence of $(\mathbf{x} : \alpha)$ in $\mathbf{B}_\beta$ (Notational Convention 8). For example, $\mathbf{A}_\alpha = (\mathbf{x} : \alpha)$ may be written as $\mathbf{A}_\alpha = \mathbf{x}$.

An expression of type α is always defined if $\alpha = o$ and may be either defined or undefined if $\alpha \neq o$. If defined, it denotes a value in D_α, the denotation of α. If undefined, it denotes nothing at all. We will use constants to denote the distinguished values of mathematical structures.

As previously defined, a *language* (or *signature*) of Alonzo is a pair $L = (\mathcal{B}, \mathcal{C})$ where $\mathcal{B}$ is a finite set of base types and $\mathcal{C}$ is a set of constants $\mathbf{c}_\alpha$ where each base type occurring in α is a member of $\mathcal{B}$. A type α is a *type of* L if all the base types occurring in α are members of $\mathcal{B}$, and an expression $\mathbf{A}_\alpha$ is an *expression of* L if all the base types occurring in $\mathbf{A}_\alpha$ are members of $\mathcal{B}$ and all the constants occurring in $\mathbf{A}_\alpha$ are members of $\mathcal{C}$. Let $\mathcal{T}(L) \subseteq \mathcal{T}$ denote the set of types of L and $\mathcal{E}(L) \subseteq \mathcal{E}$ denote the set of expressions of L. Notice that $\mathcal{B}$ and $\mathcal{C}$ may be empty, but $\mathcal{T}(L)$ and $\mathcal{E}(L)$ are always nonempty since $o \in \mathcal{T}(L)$.

3.2 Semantics

Let $L = (\mathcal{B}, \mathcal{C})$ be a language of Alonzo. We will now define the semantics of L.

A *frame* for L is a collection $\mathcal{D} = \{D_\alpha \mid \alpha \in \mathcal{T}(L)\}$ of nonempty domains (sets) of values such that:

F1. *Domain of truth values*: $D_o = \mathbb{B} = \{\text{F}, \text{T}\}$.

F2. *Predicate domain*: $D_{\alpha \to o}$ is a set of *some* total functions from D_α to D_o for $\alpha \in \mathcal{T}(L)$.

F3. *Function domain*: $D_{\alpha \to \beta}$ is a set of *some* partial and total functions from D_α to D_β for $\alpha, \beta \in \mathcal{T}(L)$ with $\beta \neq o$.

F4. *Product domain*: $D_{\alpha \times \beta} = D_\alpha \times D_\beta$ for $\alpha, \beta \in \mathcal{T}(L)$.

A predicate domain $D_{\alpha \to o}$ is *full* if it is the set of *all* total functions from D_α to D_o, and a function domain $D_{\alpha \to \beta}$ with $\beta \neq o$ is *full* if it is the set of *all* partial and total functions from D_α to D_β. The frame is *full* if $D_{\alpha \to \beta}$ is full for all $\alpha, \beta \in \mathcal{T}(L)$. Notice that the only restriction on a *base domain*, i.e., $D_{\mathbf{a}}$ for some $\mathbf{a} \in \mathcal{B}$, is that it is nonempty and that the frame is completely determined by its base domains when the frame is full. An *interpretation* of L is a pair $M = (\mathcal{D}, I)$ where $\mathcal{D} = \{D_\alpha \mid \alpha \in \mathcal{T}(L)\}$ is a frame for L and I is an *interpretation function* that maps each constant

in $\mathcal{C}$ of type α to an element of D_α. Notice that

$$(\{D_{\mathbf{a}} \mid \mathbf{a} \in \mathcal{B}\}, \{I(\mathbf{c}_\alpha) \mid \mathbf{c}_\alpha \in \mathcal{C}\})$$

is a mathematical structure. Hence an interpretation of a language *defines* (1) a mathematical structure and (2) a mapping of the base types and constants of the language to the base domains and distinguished values, respectively, of the mathematical structure.

Let $\mathcal{D} = \{D_\alpha \mid \alpha \in \mathcal{T}(L)\}$ be a frame for L. An *assignment into* $\mathcal{D}$ is a function φ whose domain is the set of variables of L such that $\varphi((\mathbf{x} : \alpha)) \in D_\alpha$ for each variable $(\mathbf{x} : \alpha)$ of L. Given an assignment φ, a variable $(\mathbf{x} : \alpha)$ of L, and $d \in D_\alpha$, let $\varphi[(\mathbf{x} : \alpha) \mapsto d]$ be the assignment ψ in $\mathcal{D}$ such that $\psi((\mathbf{x} : \alpha)) = d$ and $\psi((\mathbf{y} : \beta)) = \varphi((\mathbf{y} : \beta))$ for all variables $(\mathbf{y} : \beta)$ of L distinct from $(\mathbf{x} : \alpha)$. Given an interpretation M of L, let $\mathsf{assign}(M)$ be the set of assignments into the frame of M.

Let $\mathcal{D} = \{D_\alpha \mid \alpha \in \mathcal{T}(L)\}$ be a frame for L and $M = (\mathcal{D}, I)$ be an interpretation of L. M is a *general model* of L if there is a partial binary *valuation function* V^M such that, for all assignments $\varphi \in \mathsf{assign}(M)$ and expressions $\mathbf{C}_\gamma$ of L, (1) either $V^M_\varphi(\mathbf{C}_\gamma) \in D_\gamma$ or $V^M_\varphi(\mathbf{C}_\gamma)$ is undefined[2] and (2) each of the following conditions is satisfied:

V1. $V^M_\varphi((\mathbf{x} : \alpha)) = \varphi((\mathbf{x} : \alpha))$.

V2. $V^M_\varphi(\mathbf{c}_\alpha) = I(\mathbf{c}_\alpha)$.

V3. $V^M_\varphi(\mathbf{A}_\alpha = \mathbf{B}_\alpha) = \textsc{t}$ if $V^M_\varphi(\mathbf{A}_\alpha)$ is defined, $V^M_\varphi(\mathbf{B}_\alpha)$ is defined, and $V^M_\varphi(\mathbf{A}_\alpha) = V^M_\varphi(\mathbf{B}_\alpha)$. Otherwise, $V^M_\varphi(\mathbf{A}_\alpha = \mathbf{B}_\alpha) = \textsc{f}$.

V4. $V^M_\varphi(\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha) = V^M_\varphi(\mathbf{F}_{\alpha\to\beta})(V^M_\varphi(\mathbf{A}_\alpha))$ if $V^M_\varphi(\mathbf{F}_{\alpha\to\beta})$ is defined, $V^M_\varphi(\mathbf{A}_\alpha)$ is defined, and $V^M_\varphi(\mathbf{F}_{\alpha\to\beta})$ is defined at $V^M_\varphi(\mathbf{A}_\alpha)$. Otherwise, $V^M_\varphi(\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha) = \textsc{f}$ if $\beta = o$ and $V^M_\varphi(\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha)$ is undefined if $\beta \neq o$.

V5. $V^M_\varphi(\lambda\,\mathbf{x} : \alpha\ .\ \mathbf{B}_\beta)$ is the (partial or total) function $f \in D_{\alpha\to\beta}$ such that, for each $d \in D_\alpha$, $f(d) = V^M_{\varphi[(\mathbf{x}:\alpha)\mapsto d]}(\mathbf{B}_\beta)$ if $V^M_{\varphi[(\mathbf{x}:\alpha)\mapsto d]}(\mathbf{B}_\beta)$ is defined and $f(d)$ is undefined if $V^M_{\varphi[(\mathbf{x}:\alpha)\mapsto d]}(\mathbf{B}_\beta)$ is undefined.

V6. $V^M_\varphi(\mathrm{I}\,\mathbf{x} : \alpha\ .\ \mathbf{A}_o)$ is the $d \in D_\alpha$ such that $V^M_{\varphi[(\mathbf{x}:\alpha)\mapsto d]}(\mathbf{A}_o) = \textsc{t}$ if there is exactly one such d. Otherwise, $V^M_\varphi(\mathrm{I}\,\mathbf{x} : \alpha\ .\ \mathbf{A}_o)$ is undefined.

V7. $V^M_\varphi((\mathbf{A}_\alpha, \mathbf{B}_\beta)) = (V^M_\varphi(\mathbf{A}_\alpha), V^M_\varphi(\mathbf{B}_\beta))$ if $V^M_\varphi(\mathbf{A}_\alpha)$ and $V^M_\varphi(\mathbf{B}_\beta)$ are defined. Otherwise, $V^M_\varphi((\mathbf{A}_\alpha, \mathbf{B}_\beta))$ is undefined.

[2] We write $V^M_\varphi(\mathbf{C}_\gamma)$ instead of $V^M(\varphi, \mathbf{C}_\gamma)$.

V^M is unique when it exists. $V_\varphi^M(\mathbf{C}_\gamma)$ is called the *value of* $\mathbf{C}_\gamma$ *in* M *with respect to* φ when $V_\varphi^M(\mathbf{C}_\gamma)$ is defined. $\mathbf{C}_\gamma$ is said to have no value in M with respect to φ when $V_\varphi^M(\mathbf{C}_\gamma)$ is undefined.

An interpretation $M = (\mathcal{D}, I)$ of L is a *standard model* of L if $\mathcal{D}$ is full. Every standard model of L is a general model of L.

3.3 Additional compact notation

The compact notation for Alonzo types and expressions given above is extended in [20] with a variety of operators, binders, and abbreviations. Equipped with this additional compact notation, Alonzo becomes a practical logic in which mathematical ideas can be expressed naturally and succinctly. The compact notation that we need in this paper from Chapter 6 of [20] is presented in Tables 1–8. To make the notational definitions as readable as possible we have omitted matching parentheses in the right-hand side of the definitions when there is no loss of meaning and it is obvious where they should occur.

In Table 1, we present notation for the truth values and the standard Boolean operators. The notation $\wedge_{o\to o\to o}$ is an example of a *pseudoconstant*. It is not a real constant of Alonzo, but it stands for an expression $\mathbf{C}_\gamma$ that can be used just like a constant $\mathbf{c}_\gamma$. Unlike a normal constant, $\wedge_{o\to o\to o}$ and most other pseudoconstants can be employed in any language. Thus they serve as logical constants. The same symbols that are used to write constants are used to write pseudoconstants and parametric pseudoconstants (which are defined below) (Notational Convention 9).

In Table 2, we present notation for binary operators. We will occasionally use implicit notational definitions analogous to the notational definitions in Table 2 for the infix operators $<$, $>$, and $\geq$ corresponding to $\leq$ for other weak order operators such as $\subseteq$ and $\sqsubseteq$ (Notational Convention 10).

In Table 3, we present notation for the universal and existential quantifiers. We will usually write a sequence of universal quantifiers and a sequence of existential quantifiers in a more compact form with a single quantifier (Notational Convention 11). Thus, for example,

$$\forall\, \mathbf{x} : \alpha \,.\, \forall\, \mathbf{y} : \alpha \,.\, \forall\, \mathbf{z} : \beta \,.\, \mathbf{A}_o$$

will be written as

$$\forall\, \mathbf{x}, \mathbf{y} : \alpha,\, \mathbf{z} : \beta \,.\, \mathbf{A}_o.$$

We will also use this form with quasitypes (which are introduced below) (Notational Convention 12).

T_o	stands for	$(\lambda\, x : o \,.\, x) = (\lambda\, x : o \,.\, x)$.
F_o	stands for	$(\lambda\, x : o \,.\, T_o) = (\lambda\, x : o \,.\, x)$.
$\wedge_{o\to o\to o}$	stands for	$\lambda\, x : o \,.\, \lambda\, y : o \,.$ $(\lambda\, g : o \to o \to o \,.\, g\, T_o\, T_o) =$ $(\lambda\, g : o \to o \to o \,.\, g\, x\, y)$.
$(\mathbf{A}_o \wedge \mathbf{B}_o)$	stands for	$\wedge_{o\to o\to o}\, \mathbf{A}_o\, \mathbf{B}_o$.
$\Rightarrow_{o\to o\to o}$	stands for	$\lambda\, x : o \,.\, \lambda\, y : o \,.\, x = (x \wedge y)$.
$(\mathbf{A}_o \Rightarrow \mathbf{B}_o)$	stands for	$\Rightarrow_{o\to o\to o}\ \mathbf{A}_o\, \mathbf{B}_o$.
$\neg_{o\to o}$	stands for	$\lambda\, x : o \,.\, x = F_o$.
$(\neg \mathbf{A}_o)$	stands for	$\neg_{o\to o}\, \mathbf{A}_o$.
$\vee_{o\to o\to o}$	stands for	$\lambda\, x : o \,.\, \lambda\, y : o \,.\, \neg(\neg x \wedge \neg y)$.
$(\mathbf{A}_o \vee \mathbf{B}_o)$	stands for	$\vee_{o\to o\to o}\, \mathbf{A}_o\, \mathbf{B}_o$.

Table 1: Notational Definitions for Boolean Operators

$(\mathbf{A}_\alpha\ \mathbf{c}\ \mathbf{B}_\alpha)$	stands for	$\mathbf{c}_{\alpha\to\alpha\to\beta}\, \mathbf{A}_\alpha\, \mathbf{B}_\alpha$ or $\mathbf{c}_{(\alpha\times\alpha)\to\beta}\, (\mathbf{A}_\alpha, \mathbf{B}_\alpha)$.
$(\mathbf{A}_o \Leftrightarrow \mathbf{B}_o)$	stands for	$\mathbf{A}_o = \mathbf{B}_o$.
$(\mathbf{A}_\alpha \neq \mathbf{B}_\alpha)$	stands for	$\neg(\mathbf{A}_\alpha = \mathbf{B}_\alpha)$.
$(\mathbf{A}_\alpha < \mathbf{B}_\alpha)$	stands for	$(\leq_{\alpha\to\alpha\to o}\, \mathbf{A}_\alpha\, \mathbf{B}_\alpha) \wedge (\mathbf{A}_\alpha \neq \mathbf{B}_\alpha)$.
$(\mathbf{A}_\alpha > \mathbf{B}_\alpha)$	stands for	$\mathbf{B}_\alpha < \mathbf{A}_\alpha$.
$(\mathbf{A}_\alpha \geq \mathbf{B}_\alpha)$	stands for	$\mathbf{B}_\alpha \leq \mathbf{A}_\alpha$.
$(\mathbf{A}_\alpha = \mathbf{B}_\alpha = \mathbf{C}_\alpha)$	stands for	$(\mathbf{A}_\alpha = \mathbf{B}_\alpha) \wedge (\mathbf{B}_\alpha = \mathbf{C}_\alpha)$.
$(\mathbf{A}_\alpha\ \mathbf{c}\ \mathbf{B}_\alpha\ \mathbf{d}\ \mathbf{C}_\alpha)$	stands for	$(\mathbf{A}_\alpha\ \mathbf{c}\ \mathbf{B}_\alpha) \wedge (\mathbf{B}_\alpha\ \mathbf{d}\ \mathbf{C}_\alpha)$.

Table 2: Notational Definitions for Binary Operators

In Table 4, we present notation for expressions involving definedness. $\bot_o$ is a canonical "undefined" formula. $\bot_\alpha$ is a canonical undefined expression of type $\alpha \neq o$. $\Delta_{\alpha\to\beta}$ is the empty function of type $\alpha \to \beta$ (where $\beta \neq o$). $(\mathbf{A}_\alpha\!\downarrow)$ and $(\mathbf{A}_\alpha\!\uparrow)$ assert that the expression $\mathbf{A}_\alpha$ is defined and undefined, respectively. $(\mathbf{A}_\alpha \simeq \mathbf{B}_\alpha)$ asserts

$(\forall\, \mathbf{x} : \alpha \,.\, \mathbf{A}_o)$	stands for	$(\lambda\, x : \alpha \,.\, T_o) = (\lambda\, \mathbf{x} : \alpha \,.\, \mathbf{A}_o)$.
$(\exists\, \mathbf{x} : \alpha \,.\, \mathbf{A}_o)$	stands for	$\neg(\forall\, \mathbf{x} : \alpha \,.\, \neg \mathbf{A}_o)$.

Table 3: Notational Definitions for Quantifiers

$\bot_o$	stands for	F_o.
$\bot_\alpha$	stands for	$\mathrm{I}\, x : \alpha \,.\, x \neq x$ where $\alpha \neq o$.
$\Delta_{\alpha\to\beta}$	stands for	$\lambda\, x : \alpha \,.\, \bot_\beta$ where $\beta \neq o$.
$(\mathbf{A}_\alpha\!\downarrow)$	stands for	$\mathbf{A}_\alpha = \mathbf{A}_\alpha$.
$(\mathbf{A}_\alpha\!\uparrow)$	stands for	$\neg(\mathbf{A}_\alpha\!\downarrow)$.
$(\mathbf{A}_\alpha \simeq \mathbf{B}_\alpha)$	stands for	$(\mathbf{A}_\alpha\!\downarrow \vee \mathbf{B}_\alpha\!\downarrow) \Rightarrow \mathbf{A}_\alpha = \mathbf{B}_\alpha$.
$(\mathbf{A}_\alpha \not\simeq \mathbf{B}_\alpha)$	stands for	$\neg(\mathbf{A}_\alpha \simeq \mathbf{B}_\alpha)$.
$\mathsf{IF}(\mathbf{A}_o, \mathbf{B}_o, \mathbf{C}_o)$	stands for	$(\mathbf{A}_o \Rightarrow \mathbf{B}_o) \wedge (\neg\mathbf{A}_o \Rightarrow \mathbf{C}_o)$.
$\mathsf{IF}(\mathbf{A}_o, \mathbf{B}_\alpha, \mathbf{C}_\alpha)$	stands for	$\mathrm{I}\, x : \alpha \,.$ $(\mathbf{A}_o \Rightarrow x = \mathbf{B}_\alpha) \wedge (\neg\mathbf{A}_o \Rightarrow x = \mathbf{C}_\alpha)$ where $\alpha \neq o$.
$(\mathbf{A}_o \mapsto \mathbf{B}_\alpha \mid \mathbf{C}_\alpha)$	stands for	$\mathsf{IF}(\mathbf{A}_o, \mathbf{B}_\alpha, \mathbf{C}_\alpha)$.

Table 4: Notational Definitions for Definedness

that the expressions $\mathbf{A}_\alpha$ and $\mathbf{B}_\alpha$ are *quasi-equal*, i.e., they are both defined and equal or both undefined. And $(\mathbf{A}_o \mapsto \mathbf{B}_\alpha \mid \mathbf{C}_\alpha)$ is a conditional expression that denotes the value of $\mathbf{B}_\alpha$ if $\mathbf{A}_o$ holds and otherwise denotes the value of $\mathbf{C}_\alpha$.

The notation $\bot_\alpha$ is an example of a *parametric pseudoconstant*. It stands for an expression $\mathbf{C}_\alpha$ where α is a *parametric type* with the syntactic variable α serving as a parameter that can be freely replaced with any type. Thus $\bot_\alpha$ is polymorphic in the sense that it can be used with expressions of different types by simply replacing the syntactic variable α with the type that is needed. $\Delta_{\alpha\to\beta}$ is similarly a parametric pseudoconstant.

The notational definitions of $\mathsf{IF}(\mathbf{A}_o, \mathbf{B}_o, \mathbf{C}_o)$ and $\mathsf{IF}(\mathbf{A}_o, \mathbf{B}_\alpha, \mathbf{C}_\alpha)$ (where $\alpha \neq o$) are *(parameterized) abbreviations* of the form

$$A(\mathbf{B}^1_{\alpha_1}, \ldots, \mathbf{B}^n_{\alpha_n}) \text{ stands for } C$$

where A is a name, $n \geq 0$, and the syntactic variables $\mathbf{B}^1_{\alpha_1}, \ldots, \mathbf{B}^1_{\alpha_1}$ appear in the expression C. A is written in uppercase sans sarif font to distinguish it from the name of a constant or pseudoconstant (Notational Convention 13). We will always assume that the bound variables introduced in C are chosen so that they are not free in $\mathbf{B}^1_{\alpha_1}, \ldots, \mathbf{B}^1_{\alpha_1}$(Notational Convention 14). For example, the bound variable $(x : \alpha)$ in the RHS of the notational definition of $\mathsf{IF}(\mathbf{A}_o, \mathbf{B}_\alpha, \mathbf{C}_\alpha)$ (where $\alpha \neq o$) in Table 4 is chosen so that it is not free in $\mathbf{A}_o$, $\mathbf{B}_\alpha$, or $\mathbf{C}_\alpha$.

Since we can identify a set $S \subseteq U$ with the predicate $p_S : U \to \mathbb{B}$ such that $a \in S$ iff $p_S(a)$, we will introduce a power set type of α, i.e., a type of the subsets of

$\{\alpha\}$	stands for	$\alpha \to o$.
$(\mathbf{A}_\alpha \in \mathbf{B}_{\{\alpha\}})$	stands for	$\mathbf{B}_{\{\alpha\}}\,\mathbf{A}_\alpha$.
$(\mathbf{A}_\alpha \notin \mathbf{B}_{\{\alpha\}})$	stands for	$\neg(\mathbf{A}_\alpha \in \mathbf{B}_{\{\alpha\}})$.
$\{\mathbf{x} : \alpha \mid \mathbf{A}_o\}$	stands for	$\lambda\,\mathbf{x} : \alpha\,.\,\mathbf{A}_o$.
$\emptyset_{\{\alpha\}}$	stands for	$\lambda\,x : \alpha\,.\,F_o$.
$\{\,\}_{\{\alpha\}}$	stands for	$\emptyset_{\{\alpha\}}$.
$U_{\{\alpha\}}$	stands for	$\lambda\,x : \alpha\,.\,T_o$.
n-α-SET	stands for	$\lambda\,x_1 : \alpha\,.\,\cdots\,.\,\lambda\,x_n : \alpha\,.\,\lambda\,x : \alpha\,.$ $x = x_1 \vee \cdots \vee x = x_n$ where $n \geq 1$.
$\{\mathbf{A}^1_\alpha, \ldots, \mathbf{A}^n_\alpha\}$	stands for	n-α-SET $\mathbf{A}^1_\alpha \cdots \mathbf{A}^n_\alpha$ where $n \geq 1$.
$\subseteq_{\{\alpha\}\to\{\alpha\}\to o}$	stands for	$\lambda\,a : \{\alpha\}\,.\,\lambda\,b : \{\alpha\}\,.$ $\forall\,x : \alpha\,.\,x \in a \Rightarrow x \in b$.
$\cup_{\{\alpha\}\to\{\alpha\}\to\{\alpha\}}$	stands for	$\lambda\,a : \{\alpha\}\,.\,\lambda\,b : \{\alpha\}\,.$ $\{x : \alpha \mid x \in a \vee x \in b\}$.
$\cap_{\{\alpha\}\to\{\alpha\}\to\{\alpha\}}$	stands for	$\lambda\,a : \{\alpha\}\,.\,\lambda\,b : \{\alpha\}\,.$ $\{x : \alpha \mid x \in a \wedge x \in b\}$.
$\overline{\cdot}_{\{\alpha\}\to\{\alpha\}}$	stands for	$\lambda\,a : \{\alpha\}\,.\,\{x : \alpha \mid x \notin a\}$.
$\overline{\mathbf{A}_{\{\alpha\}}}$	stands for	$\overline{\cdot}_{\{\alpha\}\to\{\alpha\}}\,\mathbf{A}_{\{\alpha\}}$.
$\setminus_{\{\alpha\}\to\{\alpha\}\to\{\alpha\}}$	stands for	$\lambda\,a : \{\alpha\}\,.\,\lambda\,b : \{\alpha\}\,.\,a \cap \overline{b}$.

Table 5: Notational Definitions for Sets

(α)	stands for	α.
$(\alpha_1 \times \cdots \times \alpha_n)$	stands for	$(\alpha_1 \times (\alpha_2 \times \cdots \times \alpha_n))$ where $n \geq 2$.
$(\mathbf{A}_\alpha)$	stands for	$\mathbf{A}_\alpha$.
$(\mathbf{A}^1_{\alpha_1}, \ldots, \mathbf{A}^n_{\alpha_n})$	stands for	$(\mathbf{A}^1_{\alpha_1}, (\mathbf{A}^2_{\alpha_1}, \ldots, \mathbf{A}^n_{\alpha_n}))$ where $n \geq 2$.
$\mathsf{fst}_{(\alpha\times\beta)\to\alpha}$	stands for	$\lambda\,p : \alpha \times \beta\,.\,\mathrm{I}\,x : \alpha\,.\,\exists\,y : \beta\,.\,p = (x, y)$.
$\mathsf{snd}_{(\alpha\times\beta)\to\beta}$	stands for	$\lambda\,p : \alpha \times \beta\,.\,\mathrm{I}\,y : \beta\,.\,\exists\,x : \alpha\,.\,p = (x, y)$.

Table 6: Notational Definitions for Tuples

α, as the type $\alpha \to o$ of predicates on α. The compact notation for $\alpha \to o$ is $\{\alpha\}$. We introduce this notation and compact notation for the common set operators in Table 5. $\emptyset_{\{\alpha\}}$ and $U_{\{\alpha\}}$ are parametric pseudoconstants that denote the empty set and the universal set, respectively, of the members in the domain of α.

We introduce notation for product types, tuples, and the accessors for ordered pairs in Table 6.

$\mathsf{id}_{\alpha\to\alpha}$	stands for	$\lambda x : \alpha \,.\, x.$
$\mathsf{dom}_{(\alpha\to\beta)\to\{\alpha\}}$	stands for	$\lambda f : \alpha \to \beta \,.$ $\{x : \alpha \mid (f\,x)\!\downarrow\}.$
$\mathsf{ran}_{(\alpha\to\beta)\to\{\beta\}}$	stands for	$\lambda f : \alpha \to \beta \,.$ $\{y : \beta \mid \exists\, x : \alpha \,.\, f\,x = y\}.$
$\mathsf{TOTAL}(\mathbf{F}_{\alpha\to\beta})$	stands for	$\forall\, x : \alpha \,.\, (\mathbf{F}_{\alpha\to\beta}\,x)\!\downarrow.$
$\vert_{(\alpha\to\beta)\to\{\alpha\}\to(\alpha\to\beta)}$	stands for	$\lambda f : \alpha \to \beta \,.\, \lambda s : \{\alpha\} \,.$ $\lambda x : \alpha \,.\, x \in s \mapsto f\,x \mid \bot_\beta.$
$(\mathbf{F}_{\alpha\to\beta}\vert_{\mathbf{A}_{\{\alpha\}}})$	stands for	$\vert_{(\alpha\to\beta)\to\{\alpha\}\to(\alpha\to\beta)}\,\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_{\{\alpha\}}.$

Table 7: Notational Definitions for Functions

Some convenient notation for functions is found in Table 7.

A *quasitype within type* $\alpha \in \mathcal{T}$ is any expression of type $\{\alpha\} = \alpha \to o$. A quasitype $\mathbf{Q}_{\{\alpha\}}$ denotes a subset of the domain denoted by α. Thus quasitypes represent subtypes and are useful for specifying subdomains of a domain. Unlike a type, a quasitype may denote an empty domain. Notice that an expression $\mathbf{A}_{\alpha\to o}$ is simultaneously an expression of type $\alpha \to o$, an expression of type of $\{\alpha\}$, and a quasitype within type α. So $\mathbf{A}_{\alpha\to o}$ (or $\mathbf{A}_{\{\alpha\}}$) can be used as a function, as a set, and like a type as shown below.

In Table 8, we introduce various notations for using quasitypes in place of types. Quasitypes can be used to restrict the range of a variable bound by a binder. For example, $(\lambda\, x : \mathbf{Q}_{\{\alpha\}} \,.\, \mathbf{B}_\beta)$ denotes the function denoted by $\lambda\, x : \alpha \,.\, \mathbf{B}_\beta$ weakly restricted to the domain denoted by $\mathbf{Q}_{\{\alpha\}}$. Quasitypes can also be used to sharpen definedness statements. For example, $(\mathbf{A}_\alpha \downarrow \mathbf{Q}_{\{\alpha\}})$, read as $\mathbf{A}_\alpha$ *is defined in* $\mathbf{Q}_{\{\alpha\}}$, asserts that the value of $\mathbf{A}_\alpha$ is defined and is a member of the set denoted by $\mathbf{Q}_{\{\alpha\}}$. $(\mathbf{Q}_{\{\alpha\}} \to \mathbf{R}_{\{\beta\}})$ is a quasitype within $\alpha \to \beta$ that denotes the function space from the denotation of $\mathbf{Q}_{\{\alpha\}}$ to the denotation of $\mathbf{R}_{\{\beta\}}$, and $(\mathbf{Q}_{\{\alpha\}} \times \mathbf{R}_{\{\beta\}})$ is a quasitype within $\alpha \times \beta$ that denotes the Cartesian product of the denotation of $\mathbf{Q}_{\{\alpha\}}$ and the denotation of $\mathbf{R}_{\{\beta\}}$.

4 Monoids

A *monoid* is a mathematical structure $(m, \cdot, e)$ where m is a nonempty set of values, $\cdot : (m \times m) \to m$ is an associative function, and $e \in m$ is an identity element with respect to $\cdot$. Mathematics and computing are replete with examples of monoids such as $(\mathbb{N}, +, 0)$, $(\mathbb{N}, *, 1)$, and $(\Sigma^*, ++, \epsilon)$ where Σ^* is the set of strings over an alphabet

$(\lambda\,\mathbf{x} : \mathbf{Q}_{\{\alpha\}}\,.\,\mathbf{B}_\beta)$	stands for	$\lambda\,\mathbf{x} : \alpha\,.\,(\mathbf{x} \in \mathbf{Q}_{\{\alpha\}} \mapsto \mathbf{B}_\beta \mid \bot_\beta)$.
$(\forall\,\mathbf{x} : \mathbf{Q}_{\{\alpha\}}\,.\,\mathbf{B}_o)$	stands for	$\forall\,\mathbf{x} : \alpha\,.\,(\mathbf{x} \in \mathbf{Q}_{\{\alpha\}} \Rightarrow \mathbf{B}_o)$.
$(\exists\,\mathbf{x} : \mathbf{Q}_{\{\alpha\}}\,.\,\mathbf{B}_o)$	stands for	$\exists\,\mathbf{x} : \alpha\,.\,(\mathbf{x} \in \mathbf{Q}_{\{\alpha\}} \wedge \mathbf{B}_o)$.
$(\mathrm{I}\,\mathbf{x} : \mathbf{Q}_{\{\alpha\}}\,.\,\mathbf{B}_o)$	stands for	$\mathrm{I}\,\mathbf{x} : \alpha\,.\,(\mathbf{x} \in \mathbf{Q}_{\{\alpha\}} \wedge \mathbf{B}_o)$.
$(\mathbf{A}_\alpha \downarrow \mathbf{Q}_{\{\alpha\}})$	stands for	$\mathbf{A}_\alpha\!\downarrow \wedge\, \mathbf{A}_\alpha \in \mathbf{Q}_{\{\alpha\}}$.
$(\mathbf{A}_\alpha \uparrow \mathbf{Q}_{\{\alpha\}})$	stands for	$\neg(\mathbf{A}_\alpha \downarrow \mathbf{Q}_{\{\alpha\}})$.
$\rightarrow_{\{\alpha\}\to\{\beta\}\to\{\alpha\to\beta\}}$	stands for	$\lambda\,s : \{\alpha\}\,.\,\lambda\,t : \{\beta\}\,.$ $\{f : \alpha \to \beta \mid \forall\,x : \alpha\,.$ $(f\,x)\!\downarrow \Rightarrow (x \in s \wedge f\,x \in t)\}$ where $\beta \neq o$.
$\times_{\{\alpha\}\to\{\beta\}\to\{\alpha\times\beta\}}$	stands for	$\lambda\,s : \{\alpha\}\,.\,\lambda\,t : \{\beta\}\,.$ $\{p : \alpha \times \beta \mid$ $\mathsf{fst}_{(\alpha\times\beta)\to\alpha}\,p \in s \wedge$ $\mathsf{snd}_{(\alpha\times\beta)\to\beta}\,p \in t\}$
$(\mathbf{Q}_{\{\alpha\}} \to o)$	stands for	$\{s : \{\alpha\} \mid s \subseteq \mathbf{Q}_{\{\alpha\}}\}$.
$\mathcal{P}(\mathbf{Q}_{\{\alpha\}})$	stands for	$\mathbf{Q}_{\{\alpha\}} \to o$.
$(\mathbf{Q}_{\{\alpha\}} \to \mathbf{R}_{\{\beta\}})$	stands for	$\rightarrow_{\{\alpha\}\to\{\beta\}\to\{\alpha\to\beta\}}\ \mathbf{Q}_{\{\alpha\}}\,\mathbf{R}_{\{\beta\}}$ where $\beta \neq o$.
$(\alpha \to \mathbf{R}_{\{\beta\}})$	stands for	$U_{\{\alpha\}} \to \mathbf{R}_{\{\beta\}}$ where $\beta \neq o$.
$(\mathbf{Q}_{\{\alpha\}} \to \beta)$	stands for	$\mathbf{Q}_{\{\alpha\}} \to U_{\{\beta\}}$ where $\beta \neq o$.
$(\mathbf{Q}_{\{\alpha\}} \times \mathbf{R}_{\{\beta\}})$	stands for	$\times_{\{\alpha\}\to\{\beta\}\to\{\alpha\times\beta\}}\,\mathbf{Q}_{\{\alpha\}}\,\mathbf{R}_{\{\beta\}}$.
$(\alpha \times \mathbf{R}_{\{\beta\}})$	stands for	$U_{\{\alpha\}} \times \mathbf{R}_{\{\beta\}}$.
$(\mathbf{Q}_{\{\alpha\}} \times \beta)$	stands for	$\mathbf{Q}_{\{\alpha\}} \times U_{\{\beta\}}$.
$\mathsf{TOTAL\text{-}ON}(\mathbf{F}_{\alpha\to\beta}, \mathbf{Q}_{\{\alpha\}}, \mathbf{R}_{\{\beta\}})$	stands for	$\forall\,x : \mathbf{Q}_{\{\alpha\}}\,.\,(\mathbf{F}_{\alpha\to\beta}\,x)\!\downarrow \mathbf{R}_{\{\beta\}}$.

Table 8: Notational Definitions for Quasitypes

Σ, $+\!+$ is string concatenation, and ϵ is the empty string.

Table 9 defines some parametric pseudoconstants that we will need for monoids, and Table 10 defines several useful abbreviations for monoids.

Let $T = (L, \Gamma)$ be a theory[3] of Alonzo. Consider a tuple

$$(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$$

where (1) ζ_α is either a type α of L or a closed quasitype $\mathbf{Q}_{\{\alpha\}}$ of L and (2) $\mathbf{F}_{(\alpha\times\alpha)\to\alpha}$ and $\mathbf{E}_\alpha$ are closed expressions of L. Let $\mathbf{X}_o$ be the sentence

$$\mathsf{MONOID}(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha),$$

where MONOID is the abbreviation introduced by the notational definition given in Table 10 and $\mathbf{M}_{\{\alpha\}}$ is $U_{\{\alpha\}}$ if ζ_α is α and is $\mathbf{Q}_{\{\alpha\}}$ otherwise. If $T \models \mathbf{X}_o$, then $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ denotes a monoid $(m, \cdot, e)$ in T. Stated more precisely,

[3] A *theory* of Alonzo and related notions are presented in Chapter 9 of [20].

$\mathsf{set\text{-}op}_{((\alpha\times\beta)\to\gamma)\to((\{\alpha\}\times\{\beta\})\to\{\gamma\})}$ stands for $\lambda\, f : (\alpha \times \beta) \to \gamma \;.\; \lambda\, p : \{\alpha\} \times \{\beta\} \;.$ $\{z : \gamma \mid \exists\, x : \mathsf{fst}\, p,\, y : \mathsf{snd}\, p \;.\; z = f\,(x, y)\}.$
$\circ_{((\alpha\to\beta)\times(\beta\to\gamma))\to(\alpha\to\gamma)}$ stands for $\lambda\, p : (\alpha \to \beta) \times (\beta \to \gamma) \;.\; \lambda\, x : \alpha \;.\; (\mathsf{snd}\, p)\,((\mathsf{fst}\, p)\, x).$
$(\mathbf{F}_{\alpha\to\beta} \circ \mathbf{G}_{\beta\to\gamma})$ stands for $\circ_{((\alpha\to\beta)\times(\beta\to\gamma))\to(\alpha\to\gamma)}\,(\mathbf{F}_{\alpha\to\beta}, \mathbf{G}_{\beta\to\gamma}).$
$\bullet_{((\alpha\to\beta)\times\alpha)\to\beta}$ stands for $\lambda\, p : (\alpha \to \beta) \times \alpha \;.\; (\mathsf{fst}\, p)\,(\mathsf{snd}\, p).$

Table 9: Notational Definitions for Monoids: Pseudoconstants

if $T \models \mathbf{X}_o$, then, for all general models M of T and all assignments $\varphi \in \mathsf{assign}(M)$, $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ denotes the monoid

$$(V_\varphi^M(\mathbf{M}_{\{\alpha\}}), V_\varphi^M(\mathbf{F}_{(\alpha\times\alpha)\to\alpha}), V_\varphi^M(\mathbf{E}_\alpha)).$$

Thus we can show that $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ denotes a monoid in T by proving $T \models \mathbf{X}_o$. However, we may need general definitions and theorems about monoids to prove properties in T about the monoid denoted by $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$. It would be extremely inefficient to state these definitions and prove these theorems in T since instances of these same definitions and theorems could easily be needed for other triples in T, as well as in other theories, that denote monoids.

Instead of developing part of a monoid theory in T, we should apply the little theories method and develop a "little theory" T_{mon} of monoids, separate from T, that has the most convenient level of abstraction and the most convenient vocabulary for talking about monoids. The general definitions and theorems of monoids can then be introduced in a development[4] D_{mon} of T_{mon} in a universal abstract form. When these

[4] A *development* of Alonzo and related notions are presented in Chapter 12 of [20].

$\mathsf{MONOID}(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ stands for $\mathbf{M}_{\{\alpha\}}\!\downarrow \wedge$ $\mathbf{M}_{\{\alpha\}} \neq \emptyset_{\{\alpha\}} \wedge$ $\mathbf{F}_{(\alpha\times\alpha)\to\alpha} \downarrow (\mathbf{M}_{\{\alpha\}} \times \mathbf{M}_{\{\alpha\}}) \to \mathbf{M}_{\{\alpha\}} \wedge$ $\mathbf{E}_\alpha \downarrow \mathbf{M}_{\{\alpha\}} \wedge$ $\forall\, x, y, z : \mathbf{M}_{\{\alpha\}}\ .$ $\quad \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(y, z)) = \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(\mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, y), z) \wedge$ $\forall\, x : \mathbf{M}_{\{\alpha\}}\ .\ \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(\mathbf{E}_\alpha, x) = \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, \mathbf{E}_\alpha) = x.$
$\mathsf{COM\text{-}MONOID}(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ stands for $\mathsf{MONOID}(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha) \wedge$ $\forall\, x, y : \mathbf{M}_{\{\alpha\}}\ .\ \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, y) = \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(y, x)$
$\mathsf{MON\text{-}ACTION}(\mathbf{M}_{\{\alpha\}}, \mathbf{S}_{\{\beta\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha, \mathbf{G}_{(\alpha\times\beta)\to\beta})$ stands for $\mathsf{MONOID}(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha) \wedge$ $\mathbf{S}_{\{\beta\}}\!\downarrow \wedge$ $\mathbf{S}_{\{\beta\}} \neq \emptyset_{\{\beta\}} \wedge$ $\mathbf{G}_{(\alpha\times\beta)\to\beta} \downarrow (\mathbf{M}_{\{\alpha\}} \times \mathbf{S}_{\{\beta\}}) \to \mathbf{S}_{\{\beta\}} \wedge$ $\forall\, x, y : \mathbf{M}_{\{\alpha\}},\ s : \mathbf{S}_{\{\beta\}}\ .$ $\quad \mathbf{G}_{(\alpha\times\beta)\to\beta}\,(x, \mathbf{G}_{(\alpha\times\beta)\to\beta}\,(y, s)) = \mathbf{G}_{(\alpha\times\beta)\to\beta}\,(\mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, y), s) \wedge$ $\forall\, s : \mathbf{S}_{\{\beta\}}\ .\ \mathbf{G}_{(\alpha\times\beta)\to\beta}\,(\mathbf{E}_\alpha, s) = s.$
$\mathsf{MON\text{-}HOMOM}(\mathbf{M}^1_{\{\alpha\}}, \mathbf{M}^2_{\{\beta\}}, \mathbf{F}^1_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}^1_\alpha, \mathbf{F}^2_{(\beta\times\beta)\to\beta}, \mathbf{E}^2_\beta, \mathbf{H}_{\alpha\to\beta})$ stands for $\mathsf{MONOID}(\mathbf{M}^1_{\{\alpha\}}, \mathbf{F}^1_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}^1_\alpha) \wedge$ $\mathsf{MONOID}(\mathbf{M}^2_{\{\beta\}}, \mathbf{F}^2_{(\beta\times\beta)\to\beta}, \mathbf{E}^2_\beta) \wedge$ $\mathbf{H}_{\alpha\to\beta} \downarrow \mathbf{M}^1_{\{\alpha\}} \to \mathbf{M}^2_{\{\beta\}} \wedge$ $\forall\, x, y : \mathbf{M}^1_{\{\alpha\}}\ .\ \mathbf{H}_{\alpha\to\beta}\,(\mathbf{F}^1_{(\alpha\times\alpha)\to\alpha}\,(x, y)) = \mathbf{F}^2_{(\beta\times\beta)\to\beta}\,(\mathbf{H}_{\alpha\to\beta}\,x, \mathbf{H}_{\alpha\to\beta}\,y) \wedge$ $\mathbf{H}_{\alpha\to\beta}\,\mathbf{E}^1_\alpha = \mathbf{E}^2_\beta$

Table 10: Notational Definitions for Monoids: Abbreviations

definitions and theorems are needed in a development D, a development morphism[5] from D_{mon} to D can be created and then used to transport the abstract definitions and theorems in D_{mon} to concrete instances of them in D. The validity of these concrete definitions and theorems in D is guaranteed by the fact that the abstract definitions and theorems are valid in the top theory of D_{mon} and the development morphism used to transport them preserves validity.

We can verify that $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ denotes a monoid in T by simply constructing an appropriate theory morphism Φ from T_{mon} to T. As a bonus, we can use Φ to transport the abstract definitions and theorems in D_{mon} to concrete instances of them in a development of T whenever they are needed. Moreover, we do not have to explicitly prove that a particular property of $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$, such as $\mathbf{X}_o$, that holds by virtue of $(\zeta_\alpha, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$ denoting a monoid is valid in T; instead, we only need to show that there is an abstract theorem of T_{mon} that Φ transports to this property.

The following theory definition module defines a suitably abstract theory of monoids named MON:

Theory Definition 4.1 (Monoids)**.**

Name: MON.

Base types: M.

Constants: $\cdot_{(M\times M)\to M}$, e_M.

Axioms:

1. $\forall x, y, z : M \,.\, x \cdot (y \cdot z) = (x \cdot y) \cdot z$ ($\cdot$ is associative).
2. $\forall x : M \,.\, \mathsf{e} \cdot x = x \cdot \mathsf{e} = x$ (e is an identity element with respect to $\cdot$).

Notice that we have employed several notational definitions and conventions in the axioms — including dropping the types of the constants — for the sake of brevity. This theory specifies the set of monoids exactly: The base type M, like all types, denotes a nonempty set m; the constant $\cdot_{(M\times M)\to M}$ denotes a function $\cdot : (m\times m) \to m$ that is associative; and the constant e_M denotes a member e of m that is an identity element with respect to $\cdot$.

The following development definition module defines a development, named MON-1, of the theory MON:

[5] A *theory morphism* and a *development morphism* of Alonzo are presented in Sections 14.3 and 14.4, respectively, of [20].

Development Definition 4.2 (Monoids 1)**.**

Name: MON-1.

Bottom theory: MON.

Definitions and theorems:

Thm1: $\mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$ (models of MON define monoids).

Thm2: $\mathsf{TOTAL}(\cdot_{(M\times M)\to M})$ ($\cdot$ is total).

Thm3: $\forall\, x : M \,.\, (\forall\, y : M \,.\, x \cdot y = y \cdot x = y) \Rightarrow x = \mathsf{e}$ (uniqueness of identity element).

Def1: $\mathsf{submonoid}_{\{M\}\to o} =$
$\lambda\, s : \{M\} \,.\, s \neq \emptyset_{\{M\}} \wedge (\cdot|_{s\times s} \downarrow (s \times s) \to s) \wedge \mathsf{e} \in s$ (submonoid).

Thm4: $\forall\, s : \{M\} \,.\, \mathsf{submonoid}\, s \Rightarrow \mathsf{MONOID}(s, \cdot|_{s\times s}, \mathsf{e})$ (submonoids are monoids).

Thm5: $\mathsf{submonoid}\, \{\mathsf{e}\}$ (minimum submonoid).

Thm6: $\mathsf{submonoid}\, U_{\{M\}}$ (maximum submonoid).

Def2: $\cdot^{\mathrm{op}}_{(M\times M)\to M} = \lambda\, p : M \times M \,.\, (\mathsf{snd}\, p) \cdot (\mathsf{fst}\, p)$ (opposite of $\cdot$).

Thm7: $\forall\, x, y, z : M \,.\, x \cdot^{\mathrm{op}} (y \cdot^{\mathrm{op}} z) = (x \cdot^{\mathrm{op}} y) \cdot^{\mathrm{op}} z$ ($\cdot^{\mathrm{op}}$ is associative).

Thm8: $\forall\, x : M \,.\, \mathsf{e} \cdot^{\mathrm{op}} x = x \cdot^{\mathrm{op}} \mathsf{e} = x$ (e is an identity element with respect to $\cdot^{\mathrm{op}}$).

Def3: $\odot_{(\{M\}\times\{M\})\to\{M\}} = \mathsf{set\text{-}op}_{((M\times M)\to M)\to((\{M\}\times\{M\})\to\{M\})}\, \cdot$ (set product).

Def4: $\mathsf{E}_{\{M\}} = \{\mathsf{e}_M\}$ (set identity element).

Thm9: $\forall\, x, y, z : \{M\} \,.\, x \odot (y \odot z) = (x \odot y) \odot z$ ($\odot$ is associative).

Thm10: $\forall\, x : \{M\} \,.\, \mathsf{E} \odot x = x \odot \mathsf{E} = x$ (E is an identity element with respect to $\odot$).

$\mathsf{set\text{-}op}_{((M\times M)\to M)\to(\{\{M\}\times\{M\}\}\to\{M\})}$ is an instance of the parametric pseudoconstant $\mathsf{set\text{-}op}_{((\alpha\times\beta)\to\gamma)\to(\{\{\alpha\}\times\{\beta\}\}\to\{\gamma\})}$ defined in Table 9.

Thm1 states that each model of MON defines a monoid. Thm2 states that the monoid's binary function is total (which is implied by the first axiom of MON). Thm3 states that a monoid's identity element is unique. Def1 defines the notion of a *submonoid* and Thm4–Thm6 are three theorems about submonoids. Notice that $\cdot|_{s\times s}$, the restriction of $\cdot$ to $s\times s$, denotes a partial function. Notice also that

$$\cdot|_{s\times s}\downarrow(s\times s)\to s$$

in Def1 asserts that s is closed under $\cdot|_{s\times s}$ since $\cdot$ is total by Thm2. Def2 defines $\cdot^{\mathrm{op}}_{(M\times M)\to M}$, the *opposite* of $\cdot$, and Thm7–Thm8 are key theorems about $\cdot^{\mathrm{op}}$. Def3 defines $\odot_{(\{M\}\times\{M\})\to\{M\}}$, the *set product* on $\{M\}$; Def4 defines $\mathsf{E}_{\{M\}}$, the identity element with respect to $\odot$; and Thm9–Thm10 are key theorems about $\odot$. These four definitions and ten theorems require proofs that show the RHS of each definition (i.e., the definition's definiens) is defined and each theorem is valid. The proofs are given in Appendix A.

5 Transportation of definitions and theorems

Let T be a theory such that $T\models\mathbf{X}_o$ where $\mathbf{X}_o$ is the sentence

$$\mathsf{MONOID}(\mathbf{M}_{\{\alpha\}},\mathbf{F}_{(\alpha\times\alpha)\to\alpha},\mathbf{E}_\alpha),$$

and assume that D is some development of T (which could be T itself). We would like to show how the definitions and theorems of the development MON-1 can be transported to D.[6]

Before considering the general case, we will consider the special case when $\mathbf{M}_{\{\alpha\}}$ is $U_{\{\alpha\}}$, which denotes the entire domain for the type α, and $\mathbf{F}_{(\alpha\times\alpha)\to\alpha}$ and $\mathbf{E}_\alpha$ are constants $\mathbf{c}_{(\alpha\times\alpha)\to\alpha}$ and $\mathbf{d}_\alpha$. We start by defining a theory morphism from MON to T using a theory translation definition module:

Theory Translation Definition 5.1 (Special MON to T)**.**

Name: special-MON-to-T.

Source theory: MON.

Target theory: T.

Base type mapping:

[6] A *transportation* is presented in Subsection 14.4.2 of [20].

1. $M \mapsto \alpha$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \mathbf{c}_{(\alpha\times\alpha)\to\alpha}$.
2. $\mathsf{e}_M \mapsto \mathbf{d}_\alpha$.

Since special-MON-to-T is a normal translation[7], it has no obligations of the first kind by [20, Lemma 14.10] and two obligations of the second kind which are valid in T by [20, Lemma 14.11]. It has two obligations of the third kind corresponding to the two axioms of MON. $T \models \mathbf{X}_o$ implies that each of these two obligations is valid in T. Therefore, special-MON-to-T is a theory morphism from MON to T by the Morphism Theorem [20, Theorem 14.16].[8]

Now we can transport the definitions and theorems of MON-1 to D via special-MON-to-T using definition and theorems transportation modules. For example, Thm3 and Def1 can be transported using the following two modules:

Theorem Transportation 5.2 (Transport of Thm3 to D)**.**

Name: uniqueness-of-identity-element-via-special-MON-to-D.

Source development: MON-1.

Target development: D.

Development morphism: special-MON-to-T.

Theorem:

Thm3: $\forall\, x : M \,.\, (\forall\, y : M \,.\, x \cdot y = y \cdot x = y) \Rightarrow x = \mathsf{e}$
(uniqueness of identity element).

Transported theorem:

Thm3-via-special-MON-to-T:
$\forall\, x : \alpha \,.\, (\forall\, y : \alpha \,.\, x\ \mathbf{c}\ y = y\ \mathbf{c}\ x = y) \Rightarrow x = \mathbf{d}$
(uniqueness of identity element).

New target development: D'.

[7] A *theory translation* and a *development translation* of Alonzo are presented in Subsections 14.3.1 and 14.4.1, respectively, of [20].

[8] An *obligation* of a theory translation and the Morphism Theorem are presented in Subsection 14.3.2 of [20].

Definition Transportation 5.3 (Transport of Def1 to D')**.**

Name: submonoid-via-special-MON-to-D'.

Source development: MON-1.

Target development: D'.

Development morphism: special-MON-to-T.

Definition:

Def1: $\mathsf{submonoid}_{\{M\}\to o} =$
$\lambda\, s : \{M\}\ .\ s \neq \emptyset_{\{M\}} \wedge (\cdot|_{s\times s} \downarrow (s \times s) \to s) \wedge \mathsf{e} \in s$ (submonoid).

Transported definition:

Def1-via-special-MON-to-T: $\mathsf{submonoid}_{\{\alpha\}\to o} =$
$\lambda\, s : \{\alpha\}\ .\ s \neq \emptyset_{\{\alpha\}} \wedge (\mathbf{c}|_{s\times s} \downarrow (s \times s) \to s) \wedge \mathbf{d} \in s$ (submonoid).

New target development: D''.

New development morphism: special-MON-1-to-D'.

We will next consider the general case when $\mathbf{M}_{\{\alpha\}}$ may be different from $U_{\{\alpha\}}$ and $\mathbf{F}_{(\alpha\times\alpha)\to\alpha}$ and $\mathbf{E}_\alpha$ may not be constants. The general case is usually more complicated and less succinct than the special case. We start again by defining a theory morphism from MON to T using a theory translation definition module:

Theory Translation Definition 5.4 (General MON to T)**.**

Name: general-MON-to-T.

Source theory: MON.

Target theory: T.

Base type mapping:

1. $M \mapsto \mathbf{M}_{\{\alpha\}}$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \mathbf{F}_{(\alpha\times\alpha)\to\alpha}$.
2. $\mathsf{e}_M \mapsto \mathbf{E}_\alpha$.

Let general-MON-to-T $= (\mu, \nu)$. Then general-MON-to-T has the following five obligations (one of the first, two of the second, and two of the third kind):

1. $\overline{\nu}(U_{\{M\}} \neq \emptyset_{\{M\}}) \equiv (\lambda\, x : \mathbf{M}_{\{\alpha\}}\ .\ T_o) \neq (\lambda\, x : \mathbf{M}_{\{\alpha\}}\ .\ F_o)$.
2. $\overline{\nu}(\cdot_{(M\times M)\to M} \downarrow U_{\{(M\times M)\to M\}}) \equiv$
   $\mathbf{F}_{(\alpha\times\alpha)\to\alpha} \downarrow (\lambda\, x : (\mathbf{M}_{\{\alpha\}} \times \mathbf{M}_{\{\alpha\}}) \to \mathbf{M}_{\{\alpha\}}\ .\ T_o)$.
3. $\overline{\nu}(\mathsf{e}_M \downarrow U_{\{M\}}) \equiv \mathbf{E}_\alpha \downarrow (\lambda\, x : \mathbf{M}_{\{\alpha\}}\ .\ T_o)$.
4. $\overline{\nu}(\forall\, x, y, z : M\ .\ x \cdot (y \cdot z) = (x \cdot y) \cdot z) \equiv$
   $\forall\, x, y, z : \mathbf{M}_{\{\alpha\}}\ .$
   $\mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(y, z)) = \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(\mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, y), z)$.
5. $\overline{\nu}(\forall\, x : M\ .\ \mathsf{e} \cdot x = x \cdot \mathsf{e} = x) \equiv$
   $\forall\, x : \mathbf{M}_{\{\alpha\}}\ .\ \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(\mathbf{E}_\alpha, x) = \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, \mathbf{E}_\alpha) = x$.

$\mathbf{A}_\alpha \equiv \mathbf{B}_\alpha$ means the expressions denoted by $\mathbf{A}_\alpha$ and $\mathbf{B}_\alpha$ are identical.

$T \vDash \mathbf{X}_o$ implies that each of these obligations is valid in T as follows. The first and second conjuncts of $\mathbf{X}_o$ imply that the first obligation is valid in T by part 3 of [20, Lemma 14.9]. The first and third conjuncts imply that the second obligation is valid in T by part 5 of [20, Lemma 14.9]. The first and fourth conjuncts imply that the third obligation is valid in T by part 5 of [20, Lemma 14.9]. And the fifth and sixth conjuncts imply, respectively, that the fourth and fifth obligations are valid in T. Therefore, general-MON-to-T is a theory morphism by the Morphism Theorem [20, Theorem 14.16].

We can now transport, as before, the definitions and theorems of MON-1 to D via general-MON-to-T using definition and theorem transportation modules, but we can also transport them using a group transportation module[9]. For example, Thm3 and Def1 can be transported as a group using the following group transportation module:

Group Transportation 5.5 (Transport of Thm3 and Def1 to D)**.**

Name: uniqueness-of-identity-element-and-submonoid-to-D.

Source development: MON-1.

Target development: D.

[9]This kind of module transports a set of definitions and theorems as a group in which order does not matter. A group transportation has nothing to do with the algebraic structure called a group.

Development morphism: general-MON-to-T.

Definitions and theorems:

Thm3: $\forall\, x : M \,.\, (\forall\, y : M \,.\, x \cdot y = y \cdot x = y) \Rightarrow x = \mathsf{e}$
(uniqueness of identity element).

Def1: $\mathsf{submonoid}_{\{M\}\to o} =$
$\lambda\, s : \{M\} \,.\, s \neq \emptyset_{\{M\}} \wedge (\cdot|_{s\times s} \downarrow (s \times s) \to s) \wedge \mathsf{e} \in s$ (submonoid).

Transported definitions and theorems:

Thm3-via-general-MON-to-T:
$\forall\, x : \mathbf{M}_{\{\alpha\}} \,.$
$(\forall\, y : \mathbf{M}_{\{\alpha\}} \,.\, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(x, y) = \mathbf{F}_{(\alpha\times\alpha)\to\alpha}\,(y, x) = y) \Rightarrow x = \mathbf{E}_\alpha$
(uniqueness of identity element).

Def1-via-general-MON-to-T: $\mathsf{submonoid}_{\{\alpha\}\to o} =$
$\lambda\, s : \mathcal{P}(\mathbf{M}_{\{\alpha\}}) \,.$
$s \neq (\lambda\, x : \mathbf{M}_{\{\alpha\}} \,.\, F_o) \wedge$
$(\mathbf{F}_{(\alpha\times\alpha)\to\alpha}|_{s\times s} \downarrow (s \times s) \to s) \wedge$
$\mathbf{E}_\alpha \in s$ (submonoid).

New target development: D'.

New development morphism: general-MON-1-to-D'.

The abbreviation $\mathcal{P}(\mathbf{M}_{\{\alpha\}})$, which denotes the power set of $\mathbf{M}_{\{\alpha\}}$, is defined in Table 8.

6 Opposite and set monoids

For every monoid $(m, \cdot, e)$, there is (1) an associated monoid $(m, \cdot^{\mathrm{op}}, e)$, where $\cdot^{\mathrm{op}}$ is the opposite of $\cdot$, called the *opposite monoid* of $(m, \cdot, e)$ and (2) a monoid $(\mathcal{P}(m), \odot, \{e\})$, where $\mathcal{P}(m)$ is the power set of m and $\odot$ is the set product on $\mathcal{P}(m)$, called the *set monoid* of $(m, \cdot, e)$.

We will construct a development morphism named MON-to-opposite-monoid from the theory MON to its development MON-1 that maps

$(M, \cdot_{(M\times M)\to M}, \mathsf{e})$

to

$$(M, \cdot^{\mathrm{op}}_{(M\times M)\to M}, \mathsf{e}).$$

Then we will be able to use this morphism to transport abstract definitions and theorems about monoids to more concrete definitions and theorems about opposite monoids. Here is the definition of MON-to-opposite-monoid:

Development Translation Definition 6.1 (MON to Op. Monoid)**.**

Name: MON-to-opposite-monoid.

Source development: MON.

Target development: MON-1.

Base type mapping:

1. $M \mapsto M$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \cdot^{\mathrm{op}}_{(M\times M)\to M}$.
2. $\mathsf{e}_M \mapsto \mathsf{e}_M$.

Since MON-to-opposite-monoid is a normal translation, it has no obligations of the first kind by [20, Lemma 14.10] and two obligations of the second kind which are valid in the top theory of MON-1 by [20, Lemma 14.11]. It has two obligations of the third kind corresponding to the two axioms of MON. These two obligations are logically equivalent to Thm7 and Thm8, respectively, in MON-1, and so these two theorems are obviously valid in the top theory of MON-1. Therefore, MON-to-opposite-monoid is a development morphism from MON to MON-1 by the Morphism Theorem [20, Theorem 14.16].

We can now transport Thm1 via MON-to-opposite-monoid to show that opposite monoids are indeed monoids:

Theorem Transportation 6.2 (Transport of Thm1 to MON-1)**.**

Name: monoid-via-MON-to-opposite-monoid.

Source development: MON.

Target development: MON-1.

Development morphism: MON-to-opposite-monoid.

Theorem:

Thm1: $\mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$
(models of MON define monoids).

Transported theorem:

Thm11 (Thm1-via-MON-to-opposite-monoid):
$\mathsf{MONOID}(U_{\{M\}}, \cdot^{\mathrm{op}}_{(M\times M)\to M}, \mathsf{e}_M)$ (opposite monoids are monoids).

New target development: MON-2.

Similarly, we will construct a development morphism named MON-to-set-monoid from the theory MON to its development MON-2 that maps

$$(M, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$$

to

$$(\{M\}, \odot_{(\{M\}\times\{M\})\to\{M\}}, \mathsf{E}_{\{M\}}).$$

Then we will be able to use this morphism to transport abstract definitions and theorems about monoids to more concrete definitions and theorems about set monoids. Here is the definition of MON-to-set-monoid:

Development Translation Definition 6.3 (MON to Set Monoid)**.**

Name: MON-to-set-monoid.

Source development: MON.

Target development: MON-2.

Base type mapping:

1. $M \mapsto \{M\}$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \odot_{(\{M\}\times\{M\})\to\{M\}}$.
2. $\mathsf{e}_M \mapsto \mathsf{E}_{\{M\}}$.

Since MON-to-set-monoid is a normal translation, it has no obligations of the first kind by [20, Lemma 14.10]. It has two obligations of the second kind. The first one is valid in the top theory of MON-2 by part 4 of [20, Lemma 14.9] since $\odot_{(\{M\}\times\{M\})\to\{M\}}$ beta-reduces by [20, Axiom A4] to a function abstraction which is defined by [20, Axiom A5.11]. The second one is valid in the top theory of MON-2 by part 4 of [20, Lemma 14.9] since $\mathsf{E}_{\{M\}}$ is a function abstraction which is defined by [20, Axiom A5.11]. It has two obligations of the third kind corresponding to the two axioms of MON. These two obligations are Thm9 and Thm10, respectively, in MON-2, and so these two theorems are obviously valid in the top theory of MON-2. Therefore, MON-to-set-monoid is a development morphism from MON to MON-2 by the Morphism Theorem [20, Theorem 14.16].

We can now transport Thm1 via MON-to-set-monoid to show that set monoids are indeed monoids:

Theorem Transportation 6.4 (Transport of Thm1 to MON-2)**.**

Name: monoid-via-MON-to-set-monoid.

Source development: MON.

Target development: MON-2.

Development morphism: MON-to-set-monoid.

Theorem:

Thm1: $\mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$
(models of MON define monoids).

Transported theorem:

Thm12 (Thm1-via-MON-to-set-monoid):
$\mathsf{MONOID}(U_{\{\{M\}\}}, \odot_{(\{M\}\times\{M\})\to\{M\}}, \mathsf{E}_{\{M\}})$
(set monoids are monoids).

New target development: MON-3.

7 Commutative monoids

A monoid $(m, \cdot, e)$ is *commutative* if $\cdot$ is commutative.

Let $\mathbf{Y}_o$ be the formula

$$\mathsf{COM\text{-}MONOID}(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha),$$

where COM-MONOID is the abbreviation introduced by the notational definition given in Table 10. $\mathbf{Y}_o$ asserts that the tuple

$$(\mathbf{M}_{\{\alpha\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha)$$

denotes a commutative monoid $(m, \cdot, e)$.

We can define a theory of commutative monoids, named COM-MON, by adding an axiom that says $\cdot$ is commutative to the theory MON using a theory extension module:

Theory Extension 7.1 (Commutative Monoids)**.**

Name: COM-MON.

Extends MON.

New base types:

New constants:

New axioms:

3. $\forall x, y : M \,.\, x \cdot y = y \cdot x$ ($\cdot$ is commutative).

Then we can develop the theory COM-MON using the following development definition module:

Development Definition 7.2 (Commutative Monoids 1)**.**

Name: COM-MON-1.

Bottom theory: COM-MON.

Definitions and theorems:

Thm13: COM-MONOID$(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$
(models of COM-MON define commutative monoids).

Def5: $\leq_{M\to M\to o} = \lambda\, x, y : M \,.\, \exists\, z : M \,.\, x \cdot z = y$ (weak order).

Thm14: $\forall\, x : M \,.\, x \leq x$ (reflexivity).

Thm15: $\forall\, x, y, z : M \,.\, (x \leq y \wedge y \leq z) \Rightarrow x \leq z$ (transitivity).

Thm13 states that each model of COM-MON defines a commutative monoid. Def5 defines a weak (nonstrict) order that is a pre-order by Thm14 and Thm15. We could have put Def5, Thm14, and Thm15 in a development of MON since Thm14 and Thm15 do not require that $\cdot$ is commutative, but we have put these in COM-MON instead since $\leq_{M\to M\to o}$ is more natural for commutative monoids than for noncommutative monoids.

Since COM-MON is an extension of MON, there is an inclusion (i.e., a theory morphism whose mapping is the identity function) from MON to COM-MON. This inclusion is defined by the following theory translation definition module:

Theory Translation Definition 7.3 (MON to COM-MON)**.**

Name: MON-to-COM-MON.

Source theory: MON.

Target theory: COM-MON.

Base type mapping:

1. $M \mapsto M$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \cdot_{(M\times M)\to M}$.
2. $\mathsf{e}_M \mapsto \mathsf{e}_M$.

We will assume that, whenever we define a theory extension T' of a theory T, we also simultaneously define the inclusion from T to T'.

Since MON-to-COM-MON is an inclusion from MON to COM-MON, it is also a development morphism from MON-3 to COM-MON-1 and the definitions and theorems of MON-3 can be freely transported verbatim to COM-MON-1. In the rest of the paper, when a theory T' is an extension of a theory T and D is a development of T, we will assume that the definitions and theorems of D are also definitions and theorems of any trivial or nontrivial development of T' without explicitly transporting them via the inclusion from T to T' as long as there are no name clashes. This assumption is given the name *inclusion transportation convention* in [20, Subsection 14.4.3].

8 Transformation monoids

A very important type of monoid is a monoid composed of transformations of a set. Let s be a nonempty set. Then $(f, \circ, \mathsf{id})$, where f is a set of (partial or total) functions from s to s,

$$\circ : ((s \to s) \times (s \to s)) \to (s \to s)$$

is function composition, and $\mathsf{id} : s \to s$ is the identity function, is a *transformation monoid on* s if f is closed under $\circ$ and $\mathsf{id} \in f$. It is easy to verify that every transformation monoid is a monoid. If f contains every function in the function space $s \to s$, then $(f, \circ, \mathsf{id})$ is clearly a transformation monoid which is called the *full transformation monoid on* s. Let us say that a transformation monoid $(f, \circ, \mathsf{id})$ is *standard* if f contains only total functions. In many developments, nonstandard transformation monoids are ignored, but there is no reason to do that here since Alonzo admits undefined expressions and partial functions.

Consider the following theory ONE-BT of one base type:

Theory Definition 8.1 (One Base Type)**.**

Name: ONE-BT.

Base types: S.

Constants:

Axioms:

We can define the notion of a transformation monoid in a development of this theory, but we must first introduce some general facts about function composition. To do that, we need a theory FUN-COMP with four base types in order to state the associativity theorem for function composition in full generality:

Theory Definition 8.2 (Function Composition)**.**

Name: FUN-COMP.

Base types: A, B, C, D.

Constants:

Axioms:

We introduce two theorems for function composition in a development of FUN-COMP:

Development Definition 8.3 (Function Composition 1)**.**

Name: FUN-COMP-1.

Bottom theory: FUN-COMP.

Definitions and theorems:

Thm16: $\forall f : A \to B,\, g : B \to C,\, h : C \to D \,.\, f \circ (g \circ h) = (f \circ g) \circ h$
($\circ$ is associative).

Thm17: $\forall f : A \to B \,.\, \mathsf{id}_{A\to A} \circ f = f \circ \mathsf{id}_{B\to B} = f$
(identity functions are left and right identity elements).

The parametric pseudoconstants $\circ_{((\alpha\to\beta)\times(\beta\to\gamma))\to(\alpha\to\gamma)}$ and $\mathsf{id}_{\alpha\to\alpha}$ are defined in Tables 9 and 7, respectively. The infix notation for the application of

$$\circ_{((\alpha\to\beta)\times(\beta\to\gamma))\to(\alpha\to\gamma)}$$

is also defined in Table 9.

Next we define a theory morphism from FUN-COMP to ONE-BT:

Theory Translation Definition 8.4 (FUN-COMP to ONE-BT)**.**

Name: FUN-COMP-to-ONE-BT.

Source theory: FUN-COMP.

Target theory: ONE-BT.

Base type mapping:

1. $A \mapsto S$.
2. $B \mapsto S$.
3. $C \mapsto S$.
4. $D \mapsto S$.

Constant mapping:

The translation FUN-COMP-to-ONE-BT is clearly a theory morphism by the Morphism Theorem [20, Theorem 14.16] since it is a normal translation and FUN-COMP contains no constants or axioms. So we can transport the theorems of FUN-COMP-1 to ONE-BT via FUN-COMP-to-ONE-BT:

Group Transportation 8.5 (Transport of Thm16–Thm17 to ONE-BT)**.**

Name: function-composition-theorems-via-FUN-COMP-to-ONE-BT.

Source development: FUN-COMP-1.

Target development: ONE-BT.

Development morphism: FUN-COMP-to-ONE-BT.

Definitions and theorems:

Thm16: $\forall\, f : A \to B,\, g : B \to C,\, h : C \to D \,.\, f \circ (g \circ h) = (f \circ g) \circ h$
($\circ$ is associative).

Thm17: $\forall\, f : A \to B \,.\, \mathsf{id}_{A \to A} \circ f = f \circ \mathsf{id}_{B \to B} = f$
(identity functions are left and right identity elements).

Transported definitions and theorems:

Thm18 (Thm16-via-FUN-COMP-to-ONE-BT):
$\forall\, f, g, h : S \to S \,.\, f \circ (g \circ h) = (f \circ g) \circ h$ ($\circ$ is associative).

Thm19 (Thm17-via-FUN-COMP-to-ONE-BT):
$\forall\, f : S \to S \,.\, \mathsf{id}_{S \to S} \circ f = f \circ \mathsf{id}_{S \to S} = f$
($\mathsf{id}_{S \to S}$ is an identity element with respect to $\circ$).

New target development: ONE-BT-1.

New development morphism: FUN-COMP-1-to-ONE-BT-1.

We can obtain the theorem that all transformation monoids are monoids almost for free by transporting results from MON-1 to ONE-BT-1. We start by creating the theory morphism from MON to ONE-BT that maps

$$(M, \cdot_{(M \times M) \to M}, \mathbf{e}_M)$$

to

$$(S \to S, \circ_{((S \to S) \times (S \to S)) \to (S \to S)}, \mathsf{id}_{S \to S}) :$$

Theory Translation Definition 8.6 (MON to ONE-BT)**.**

Name: MON-to-ONE-BT.

Source theory: MON.

Target theory: ONE-BT.

Base type mapping:

1. $M \mapsto S \to S$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \circ_{((S\to S)\times(S\to S))\to(S\to S)}$
2. $\mathsf{e}_M \mapsto \mathsf{id}_{S\to S}$.

The theory translation MON-to-ONE-BT is normal so that it has no obligations of the first kind by [20, Lemma 14.10]. It has two obligations of the second kind. These are valid in ONE-BT by part 4 of [20, Lemma 14.9] since $\circ_{((S\to S)\times(S\to S))\to(S\to S)}$ and $\mathsf{id}_{S\to S}$ are function abstractions which are defined by [20, Axiom A5.11]. It has two obligations of the third kind corresponding to the two axioms of MON. The two obligations are Thm18 and Thm19, respectively, in ONE-BT-1, and so these two theorems are obviously valid in the top theory of ONE-BT-1. Therefore, MON-to-ONE-BT is a theory morphism from MON to ONE-BT by the Morphism Theorem [20, Theorem 14.16].

We can transport Def1, the definition of $\mathsf{submonoid}_{\{M\}\to o}$, and Thm4, the theorem that says all submonoids are monoids, to ONE-BT-1 via MON-to-ONE-BT by a group transportation module:

Group Transportation 8.7 (Transport of Def1 & Thm2 to ONE-BT-1)**.**

Name: submonoids-via-MON-to-ONE-BT.

Source development: MON-1.

Target development: ONE-BT-1.

Development morphism: MON-to-ONE-BT.

Definitions and theorems:

Def1: $\mathsf{submonoid}_{\{M\}\to o} =$
$\lambda\, s : \{M\} \,.\, s \neq \emptyset_{\{M\}} \wedge (\cdot|_{s\times s} \downarrow (s \times s) \to s) \wedge \mathsf{e} \in s$ (submonoid).

Thm4: $\forall\, s : \{M\} \,.\, \mathsf{submonoid}\, s \Rightarrow \mathsf{MONOID}(s, \cdot|_{s\times s}, \mathsf{e})$
(submonoids are monoids).

Transported definitions and theorems:

Def6 (Def1-via-MON-to-ONE-BT): $\mathsf{trans\text{-}monoid}_{\{S\to S\}\to o} =$
$\lambda\, s : \{S \to S\}\ .$
$s \neq \emptyset_{\{S\to S\}} \wedge$
$(\circ_{((S\to S)\times(S\to S))\to(S\to S)}|_{s\times s} \downarrow (s \times s) \to s) \wedge$
$\mathsf{id}_{S\to S} \in s$ (transformation monoid).

Thm20 (Thm4-via-MON-to-ONE-BT):
$\forall\, s : \{S \to S\}\ .$
$\mathsf{trans\text{-}monoid}\, s \Rightarrow \mathsf{MONOID}(s, \circ_{((S\to S)\times(S\to S))\to(S\to S)}|_{s\times s}, \mathsf{id}_{S\to S})$
(transformation monoids are monoids).

New target development: ONE-BT-2.

New development morphism: MON-1-to-ONE-BT-2.

trans-monoid is a predicate that is true when it is applied to a set of functions of $S \to S$ that forms a transformation monoid. Thm20 says that every transformation monoid — including the full transformation monoid — is a monoid.

9 Monoid actions

A *(left) monoid action* is a mathematical structure $(m, s, \cdot, e, \mathsf{act})$ where $(m, \cdot, e)$ is a monoid and $\mathsf{act} : (m \times s) \to s$ is a function such that

(1) $x\ \mathsf{act}\ (y\ \mathsf{act}\ z) = (x \cdot y)\ \mathsf{act}\ z$

for all $x, y \in m$ and $z \in s$ and

(2) $e\ \mathsf{act}\ z = z$

for all $z \in s$. We say in this case that the monoid $(m, \cdot, e)$ *acts on* the set s *by* the function act.

Let $\mathbf{Z}_o$ be the formula

$$\mathsf{MON\text{-}ACTION}(\mathbf{M}_{\{\alpha\}}, \mathbf{S}_{\{\beta\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha, \mathbf{G}_{(\alpha\times\beta)\to\beta}),$$

where MON-ACTION is the abbreviation introduced by the notational definition given in Table 10. $\mathbf{Z}_o$ asserts that the tuple

$$(\mathbf{M}_{\{\alpha\}}, \mathbf{S}_{\{\beta\}}, \mathbf{F}_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}_\alpha, \mathbf{G}_{(\alpha\times\beta)\to\beta})$$

denotes a monoid action $(m, s, \cdot, e, \mathsf{act})$.

A theory of monoid actions is defined as an extension of the theory of monoids:

Theory Extension 9.1 (Monoid Actions)**.**

Name: MON-ACT.

Extends MON.

New base types: S.

New constants: $\mathsf{act}_{(M\times S)\to S}$.

New axioms:

3. $\forall\, x, y : M,\ s : S\ .\ x\ \mathsf{act}\ (y\ \mathsf{act}\ s) = (x \cdot y)\ \mathsf{act}\ s$ (act is compatible with $\cdot$).

4. $\forall\, s : S\ .\ \mathsf{e}\ \mathsf{act}\ s = s$ (act is compatible with e).

We begin a development of MON-ACT by adding the definitions and theorems below:

Development Definition 9.2 (Monoid Actions 1)**.**

Name: MON-ACT-1.

Bottom theory: MON-ACT.

Definitions and theorems:

Thm21: $\mathsf{MON\text{-}ACTION}(U_{\{M\}}, U_{\{S\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \mathsf{act}_{(M\times S)\to S})$ (models of MON-ACT define monoid actions).

Thm22: $\mathsf{TOTAL}(\mathsf{act}_{(M\times S)\to S})$ (act is total).

Def7: $\mathsf{orbit}_{S\to\{S\}} = \lambda\, s : S\ .\ \{t : S \mid \exists\, x : M\ .\ x\ \mathsf{act}\ s = t\}$ (orbit).

Def8: $\mathsf{stabilizer}_{S\to\{M\}} = \lambda\, s : S\ .\ \{x : M \mid x\ \mathsf{act}\ s = s\}$ (stabilizer).

Thm23: $\forall\, s : S\ .\ \mathsf{submonoid}\ (\mathsf{stabilizer}\ s)$ (stabilizers are submonoids).

Thm21 states that each model of MON-ACTION defines a monoid action. Thm22 says that $\mathsf{act}_{(M\times S)\to S}$ is total (which is implied by the third axiom of MON-ACTION). Def7 and Def8 introduce the concepts of an orbit and a stabilizer. And Thm23 states that a stabilizer of a monoid action $(m, s, \cdot, e, \mathsf{act})$ is a submonoid of the monoid $(m, \cdot, e)$. The power of this machinery — monoid actions with orbits and stabilizers — is low with arbitrary monoids but very high with groups, i.e., monoids in which every element has an inverse.

Monoid actions are common in monoid theory. We will present two important examples of monoid actions. The first is the monoid action $(m, m, \cdot, e, \cdot)$ such that the monoid $(m, \cdot, e)$ acts on the set m of its elements by its function $\cdot$. We formalize this by creating the theory morphism from MON-ACT to MON that maps

$$(M, S, \cdot_{(M\times M)\to M}, \mathbf{e}_M, \mathsf{act}_{(M\times S)\to S})$$

to

$$(M, M, \cdot_{(M\times M)\to M}, \mathbf{e}_M, \cdot_{(M\times M)\to M}) :$$

Theory Translation Definition 9.3 (MON-ACT to MON)**.**

Name: MON-ACT-to-MON.

Source theory: MON-ACT.

Target theory: MON.

Base type mapping:

1. $M \mapsto M$.
2. $S \mapsto M$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \cdot_{(M\times M)\to M}$.
2. $\mathsf{e}_M \mapsto \mathsf{e}_M$.
3. $\mathsf{act}_{(M\times S)\to S} \mapsto \cdot_{(M\times M)\to M}$.

It is an easy exercise to verify, arguing as we have above, that MON-ACT-to-MON is a theory morphism.

We can now transport Thm21 from MON-ACT to MON-3 via MON-ACT-to-MON to show that the action of a monoid $(m, \cdot, e)$ on m by $\cdot$ is a monoid action:

Theorem Transportation 9.4 (Transport of Thm21 to MON-3)**.**

Name: monoid-action-via-MON-ACT-to-MON.

Source development: MON-ACT.

Target development: MON-3.

Development morphism: MON-ACT-to-MON.

Theorem:

Thm21: MON-ACTION$(U_{\{M\}}, U_{\{S\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \mathsf{act}_{(M\times S)\to S})$
(models of MON-ACT define monoid actions).

Transported theorem:

Thm24 (Thm21-via-MON-ACT-to-MON):
MON-ACTION$(U_{\{M\}}, U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \cdot_{(M\times M)\to M})$
(first example is a monoid action).

New target development: MON-4.

The second example is a standard transformation monoid $(f, \circ, \mathsf{id})$ on s acting on s by the function that applies a transformation to a member of s. (Note that all the functions in f are total by virtue of the transformation monoid being standard.) We formalize this example as a theory morphism from MON-ACT to ONE-BT extended with a set constant that denotes a standard transformation monoid. Here is the extension with a set constant $\mathsf{F}_{\{S\to S\}}$ and two axioms:

Theory Extension 9.5 (One Base Type with a Set Constant)**.**

Name: ONE-BT-with-SC.

Extends ONE-BT.

New base types:

New constants: $\mathsf{F}_{\{S\to S\}}$.

New axioms:

1. trans-monoid F (F forms a transformation monoid).
2. $\forall\, f : \mathsf{F}\ .\ \mathsf{TOTAL}(f)$ (the members of F are total functions).

And here is the theory morphism from MON-ACT to ONE-BT-with-SC that maps

$$(M, S, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \mathsf{act}_{(M\times S)\to S})$$

to

$$(\mathsf{F}_{\{S\to S\}}, S, \circ_{((S\to S)\times(S\to S))\to(S\to S)}|_{\mathsf{F}\times\mathsf{F}}, \mathsf{id}_{S\to S}, \bullet_{((S\to S)\times S)\to S}|_{\mathsf{F}\times S}) :$$

Theory Translation Definition 9.6 (MON-ACT to ONE-BT-with-SC)**.**

Name: MON-ACT-to-ONE-BT-with-SC.

Source theory: MON-ACT.

Target theory: ONE-BT-with-SC.

Base type mapping:

1. $M \mapsto \mathsf{F}_{\{S \to S\}}$.
2. $S \mapsto S$.

Constant mapping:

1. $\cdot_{(M \times M) \to M} \mapsto \circ_{((S \to S) \times (S \to S)) \to (S \to S)} | \mathsf{F} \times \mathsf{F}$.
2. $\mathsf{e}_M \mapsto \mathsf{id}_{S \to S}$.
3. $\mathsf{act}_{(M \times S) \to S} \mapsto \bullet_{((S \to S) \times S) \to S} | \mathsf{F} \times S$.

The parametric pseudoconstant $\bullet_{((S \to S) \times S) \to S} | \mathsf{F} \times S$ is defined in Table 9. It is a straightforward exercise to verify, arguing as we have above, that MON-ACT-to-ONE-BT-with-SC is a theory morphism.

We can now transport Thm21 from MON-ACT to ONE-BT-with-S via MON-ACT-to-ONE-BT-with-SC to show that a standard transformation monoid $(f, \circ, \mathsf{id})$ on s acting on s by the function that applies a (total) transformation to a member of s is a monoid action:

Theorem Transportation 9.7 (Trans. of Thm21 to ONE-BT-with-SC)**.**

Name: monoid-action-via-MON-ACT-to-ONE-BT-with-SC.

Source development: MON-ACT.

Target development: ONE-BT-with-SC.

Development morphism: MON-ACT-to-ONE-BT-with-SC.

Theorem:

Thm21: MON-ACTION$(U_{\{M\}}, U_{\{S\}}, \cdot_{(M \times M) \to M}, \mathsf{e}_M, \mathsf{act}_{(M \times S) \to S})$
(models of MON-ACT define monoid actions).

Transported theorem:

Thm25 (Thm21-via-MON-ACT-to-ONE-BT-with-SC):
MON-ACTION($\mathsf{F}_{\{S\to S\}}$,
$U_{\{S\}}$,
$\circ_{((S\to S)\times(S\to S))\to(S\to S)|\mathsf{F}\times\mathsf{F}}$,
$\mathsf{id}_{S\to S}$,
$\bullet_{((S\to S)\times S)\to S|\mathsf{F}\times S}$)
(second example is a monoid action).

New target development: ONE-BT-with-SC-1.

10 Monoid homomorphisms

Roughly speaking, a *monoid homomorphism* is a structure-preserving mapping from one monoid to another.

Let $\mathbf{W}_o$ be the formula

$$\mathsf{MON\text{-}HOMOM}(\mathbf{M}^1_{\{\alpha\}}, \mathbf{M}^2_{\{\beta\}}, \mathbf{F}^1_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}^1_\alpha, \mathbf{F}^2_{(\beta\times\beta)\to\beta}, \mathbf{E}^2_\beta, \mathbf{H}_{\alpha\to\beta}),$$

where MON-HOMOM is the abbreviation introduced by the notational definition given in Table 10. $\mathbf{W}_o$ asserts that the tuple

$$(\mathbf{M}^1_{\{\alpha\}}, \mathbf{M}^2_{\{\beta\}}, \mathbf{F}^1_{(\alpha\times\alpha)\to\alpha}, \mathbf{E}^1_\alpha, \mathbf{F}^2_{(\beta\times\beta)\to\beta}, \mathbf{E}^2_\beta, \mathbf{H}_{\alpha\to\beta})$$

denotes a mathematical structure $(m_1, m_2, \cdot_1, e_1, \cdot_2, e_2, h)$ where $(m_1, \cdot_1, e_1)$ is a monoid, $(m_2, \cdot_2, e_2)$ is a monoid, and $h : m_1 \to m_2$ is a monoid homomorphism from $(m_1, \cdot_1, e_1)$ to $(m_2, \cdot_2, e_2)$.

The notion of a monoid homomorphism is captured in the theory MON-HOM:

Theory Definition 10.1 (Monoid Homomorphisms).

Name: MON-HOM.

Base types: M_1, M_2.

Constants: $\cdot_{(M_1\times M_1)\to M_1}$, e_{M_1}, $\cdot_{(M_2\times M_2)\to M_2}$, e_{M_2}, $\mathsf{h}_{M_1\to M_2}$.

Axioms:

1. $\forall x, y, z : M_1 \,.\, x \cdot (y \cdot z) = (x \cdot y) \cdot z$ ($\cdot_{(M_1\times M_1)\to M_1}$ is associative).
2. $\forall x : M_1 \,.\, \mathsf{e} \cdot x = x \cdot \mathsf{e} = x$ (e_{M_1} is an identity element).
3. $\forall x, y, z : M_2 \,.\, x \cdot (y \cdot z) = (x \cdot y) \cdot z$ ($\cdot_{(M_2\times M_2)\to M_2}$ is associative).

4. $\forall\, x : M_2 \,.\, \mathsf{e} \cdot x = x \cdot \mathsf{e} = x$ (e_{M_2} is an identity element).
5. $\forall\, x, y : M_1 \,.\, \mathsf{h}\,(x \cdot y) = (\mathsf{h}\,x) \cdot (\mathsf{h}\,y)$ (first homomorphism property).
6. $\mathsf{h}\,\mathsf{e}_{M_1} = \mathsf{e}_{M_2}$ (second homomorphism property).

$\mathsf{h}_{M_1 \to M_2}$ denotes a monoid homomorphism from the monoid denoted by

$$(M_1, \cdot_{(M_1 \times M_1) \to M_1}, \mathsf{e}_{M_1})$$

to the monoid denoted by

$$(M_2, \cdot_{(M_2 \times M_2) \to M_2}, \mathsf{e}_{M_2}).$$

Here is a simple development of MON-HOM:

Development Definition 10.2 (Monoid Homomorphisms 1)**.**

Name: MON-HOM-1.

Bottom theory: MON-HOM.

Definitions and theorems:

Thm26:
MON-HOM($U_{\{M_1\}}$,
$U_{\{M_2\}}$,
$\cdot_{(M_1 \times M_1) \to M_1}$,
e_{M_1},
$\cdot_{(M_2 \times M_2) \to M_2}$,
e_{M_2},
$\mathsf{h}_{M_1 \to M_2}$)
(models of MON-HOM define monoid homomorphisms).

Thm27: TOTAL($\mathsf{h}_{M_1 \to M_2}$) ($\mathsf{h}_{M_1 \to M_2}$ is total).

There are embeddings (i.e., theory morphisms whose mappings are injective) from MON to the two copies of MON within MON-HOM defined by the following two theory translation definitions:

Theory Translation Definition 10.3 (First MON to MON-HOM)**.**

Name: first-MON-to-MON-HOM.

Source theory: MON.

Target theory: MON-HOM.

Base type mapping:

1. $M \mapsto M_1$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \cdot_{(M_1\times M_1)\to M_1}$.
2. $\mathsf{e}_M \mapsto \mathsf{e}_{M_1}$.

Theory Translation Definition 10.4 (Second MON to MON-HOM)**.**

Name: second-MON-to-MON-HOM.

Source theory: MON.

Target theory: MON-HOM.

Base type mapping:

1. $M \mapsto M_2$.

Constant mapping:

1. $\cdot_{(M\times M)\to M} \mapsto \cdot_{(M_2\times M_2)\to M_2}$.
2. $\mathsf{e}_M \mapsto \mathsf{e}_{M_2}$.

An example of a monoid homomorphism from the monoid denoted by

$$(M, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$$

to the monoid denoted by

$$(\{M\}, \odot_{(\{M\}\times\{M\})\to\{M\}}, \mathsf{E}_{\{M\}})$$

is the function that maps a member x of the denotation of M to the singleton $\{x\}$. This monoid homomorphism is formalized by the following development morphism:

Development Translation Definition 10.5 (MON-HOM to MON)**.**

Name: MON-HOM-to-MON-4.

Source development: MON-HOM.

Target development: MON-4.

Base type mapping:

1. $M_1 \mapsto M$.
2. $M_2 \mapsto \{M\}$.

Constant mapping:

1. $\cdot_{(M_1 \times M_1) \to M_1} \mapsto \cdot_{(M \times M) \to M}$.
2. $\mathsf{e}_{M_1} \mapsto \mathsf{e}_M$.
3. $\cdot_{(M_2 \times M_2) \to M_2} \mapsto \odot_{(\{M\} \times \{M\}) \to \{M\}}$.
4. $\mathsf{e}_{M_2} \mapsto \mathsf{E}_{\{M\}}$.
5. $\mathsf{h}_{M_1 \to M_2} \mapsto \lambda\, x : M \,.\, \{x\}$.

It is a straightforward exercise to verify that HOM-MON-to-MON-4 is a theory morphism by the arguments we employed above.

We can now transport Thm26 from MON-HOM to MON-4 via MON-HOM-to-MON-4 to show the example is a monoid homomorphism:

Theorem Transportation 10.6 (Transport of Thm26 to MON-4)**.**

Name: monoid-action-via-MON-HOM-to-MON-4.

Source development: MON-HOM.

Target development: MON-4.

Development morphism: MON-HOM-to-MON-4.

Theorem:

Thm26:
MON-HOM($U_{\{M_1\}}$,
$U_{\{M_2\}}$,
$\cdot_{(M_1 \times M_1) \to M_1}$,
e_{M_1},
$\cdot_{(M_2 \times M_2) \to M_2}$,
e_{M_2},
$\mathsf{h}_{M_1 \to M_2}$)
(models of MON-HOM define monoid homomorphisms).

Transported theorem:

Thm28 (Thm26-via-MON-HOM-to-MON-4)
MON-HOM($U_{\{M\}}$,
$U_{\{\{M\}\}}$,
$\cdot_{(M\times M)\to M}$,
e_M,
$\odot_{(\{M\}\times\{M\})\to\{M\}}$,
$\mathsf{E}_{\{M\}}$,
$\lambda\, x : M \,.\, \{x\}$) (example is a monoid homomorphism).

New target development: MON-5.

11 Monoids over real number arithmetic

We need machinery concerning real number arithmetic to express some concepts about monoids. For instance, an iterated product operator for monoids involves integers. To formalize these kinds of concepts, we need a theory of monoids that includes real number arithmetic. Chapter 13 of [20] presents COF, a theory of complete ordered fields. COF is categorical in the standard sense (see [20]). That is, it has a single standard model up to isomorphism that defines the structure of real number arithmetic.

We define a theory of monoids over COF by extending COF with the language and axioms of MON:

Theory Extension 11.1 (Monoids over COF)**.**

Name: MON-over-COF.

Extends COF.

New base types: M.

New constants: $\cdot_{(M\times M)\to M}$, e_M.

New axioms:

19. $\forall\, x, y, z : M \,.\, x \cdot (y \cdot z) = (x \cdot y) \cdot z$ ($\cdot$ is associative).

20. $\forall\, x : M \,.\, \mathsf{e} \cdot x = x \cdot \mathsf{e} = x$ (e is an identity element).

We can now define an iterated product operator for monoids in a development of MON-over-COF-1:

$\left(\prod_{\mathbf{i}=\mathbf{M}_R}^{\mathbf{N}_R} \mathbf{A}_M\right)$	stands for $\mathsf{prod}_{R\to R\to(R\to M)\to M}\ \mathbf{M}_R\,\mathbf{N}_R\,(\lambda\,\mathbf{i}:R\,.\,\mathbf{A}_M).$

Table 11: Notational Definition for Monoids: Iterated Product Operator

Development Definition 11.2 (Monoids over COF 1)**.**

Name: MON-over-COF-1.

Bottom theory: MON-over-COF.

Definitions and theorems:

Def9: $\mathsf{prod}_{R\to R\to(R\to M)\to M} =$
$\mathrm{I}\,f : Z_{\{R\}} \to Z_{\{R\}} \to (Z_{\{R\}} \to M) \to M\ .$
$\forall\, m, n : Z_{\{R\}},\ g : Z_{\{R\}} \to M\ .\ f\,m\,n\,g \simeq$
$(m > n \mapsto \mathsf{e} \mid (f\,m\,(n-1)\,g)\cdot(g\,n))$ (iterated product).

Thm29: $\forall\, m : Z_{\{R\}},\ g : Z_{\{R\}} \to M\ .\ \left(\prod_{i=m}^{m} g\,i\right) \simeq g\,m$
(trivial product).

Thm30: $\forall\, m, k, n : Z_{\{R\}},\ g : Z_{\{R\}} \to M\ .$
$m < k < n \Rightarrow \left(\prod_{i=m}^{k} g\,i\right)\cdot\left(\prod_{i=k+1}^{n} g\,i\right) \simeq \prod_{i=m}^{n} g\,i$
(extended iterated product).

We are utilizing the notation for the iterated product operator defined in Table 11. $Z_{\{R\}}$ is a quasitype defined in the development COF-dev-2 of COF found in [20] that denotes the set of integers. ($Z_{\{R\}}$ is automatically available in MON-over-COF by the inclusion transportation convention presented in Section 7.) Def9 defines the iterated product operator, and Thm29 and Thm30 are two theorems about the operator.

We can similarly define extensions of MON over COF. For example, here is a theory of commutative monoids over COF and a development of it:

Theory Extension 11.3 (Commutative Monoids over COF)**.**

Name: COM-MON-over-COF.

Extends MON-over-COF.

New base types:

New constants:

New axioms:

21. $\forall x, y : M \,.\, x \cdot y = y \cdot x$ ($\cdot$ is commutative).

Development Definition 11.4 (Com. Monoids over COF 1)**.**

Name: COM-MON-over-COF-1.

Bottom theory: COM-MON-over-COF.

Definitions and theorems:

Thm31: $\forall m, n : Z_{\{R\}},\, g, h : Z_{\{R\}} \to M \,.$
$\left(\prod_{i=m}^{n} g\, i\right) \cdot \left(\prod_{i=m}^{n} h\, i\right) \simeq \prod_{i=m}^{n} (g\, i) \cdot (h\, i)$
(product of iterated products).

Notice that this theorem holds only if $\cdot$ is commutative.

For another example, here is a theory of commutative monoid actions over COF and a development of it:

Theory Extension 11.5 (Commutative Monoid Actions over COF)**.**

Name: COM-MON-ACT-over-COF.

Extends COM-MON-over-COF.

New base types: S.

New constants: $\mathsf{act}_{(M \times S) \to S}$.

New axioms:

22. $\forall x, y : M,\, s : S \,.\, x \;\mathsf{act}\; (y \;\mathsf{act}\; s) = (x \cdot y) \;\mathsf{act}\; s$
(act is compatible with $\cdot$).

23. $\forall s : S \,.\, \mathsf{e} \;\mathsf{act}\; s = s$ (act is compatible with e).

Development Definition 11.6 (Com. Monoid Actions over COF 1)**.**

Name: COM-MON-ACT-over-COF-1.

Bottom theory: COM-MON-ACT-over-COF.

Definitions and theorems:

Thm32: $\forall x, y : M,\, s : S \,.\, x \;\mathsf{act}\; (y \;\mathsf{act}\; s) = y \;\mathsf{act}\; (x \;\mathsf{act}\; s)$
(act has commutative-like property).

$\mathsf{sequences}_{\{\alpha\to\beta\}}$	stands for	$\mathbf{C}^N_{\{\alpha\}} \to \beta$.
$\langle\!\langle\beta\rangle\!\rangle$	stands for	$\mathsf{sequences}_{\{\alpha\to\beta\}}$.
$\mathsf{streams}_{\{\alpha\to\beta\}}$	stands for	$\{s : \langle\!\langle\beta\rangle\!\rangle \mid \mathsf{TOTAL}(s)\}$.
$\langle\beta\rangle$	stands for	$\mathsf{streams}_{\{\alpha\to\beta\}}$.
$\mathsf{lists}_{\{\alpha\to\beta\}}$	stands for	$\{s : \langle\!\langle\beta\rangle\!\rangle \mid \exists\, n : \mathbf{C}^N_{\{\alpha\}}\ .\ \forall\, m : \mathbf{C}^N_{\{\alpha\}}\ .$ $(s\,m)\!\downarrow \Leftrightarrow \mathbf{C}^{\leq}_{\alpha\to\alpha\to o}\, m\, (\mathbf{C}^P_{\alpha\to\alpha}\, n)\}$.
$[\beta]$	stands for	$\mathsf{lists}_{\{\alpha\to\beta\}}$.
$\mathsf{cons}_{\beta\to(\alpha\to\beta)\to(\alpha\to\beta)}$	stands for	$\lambda\, x : \beta\ .\ \lambda\, s : \langle\!\langle\beta\rangle\!\rangle\ .\ \lambda\, n : \mathbf{C}^N_{\{\alpha\}}\ .$ $n = \mathbf{C}^0_\alpha \mapsto x \mid s\, (\mathbf{C}^P_{\alpha\to\alpha}\, n)$.
$(\mathbf{A}_\beta :: \mathbf{B}_{\alpha\to\beta})$	stands for	$\mathsf{cons}_{\beta\to(\alpha\to\beta)\to(\alpha\to\beta)}\, \mathbf{A}_\beta\, \mathbf{B}_{\alpha\to\beta}$.
$\mathsf{nil}_{\alpha\to\beta}$	stands for	$\Delta_{\alpha\to\beta}$.
$[\,]_{\alpha\to\beta}$	stands for	$\mathsf{nil}_{\alpha\to\beta}$.
$[\mathbf{A}_\beta]$	stands for	$(\mathbf{A}_\beta :: [\,]_{\alpha\to\beta})$.
$[\mathbf{A}^1_\beta, \ldots, \mathbf{A}^n_\beta]$	stands for	$(\mathbf{A}^1_\beta :: [\mathbf{A}^2_\beta, \ldots, \mathbf{A}^n_\beta])$ where $n \geq 2$.
$\mathsf{len}_{(\alpha\to\beta)\to\alpha}$	stands for	$\mathrm{I}\, f : [\beta] \to \mathbf{C}^N_{\{\alpha\}}\ .$ $f\,[\,]_{\alpha\to\beta} = \mathbf{C}^0_\alpha \wedge$ $\forall\, x : \beta,\, s : [\beta]\ .$ $f\,(x :: s) = \mathbf{C}^+_{\alpha\to\alpha\to\alpha}\, (f\, s)\, (\mathbf{C}^S_{\alpha\to\alpha}\, \mathbf{C}^0_\alpha)$.
$\lvert\mathbf{A}_{\alpha\to\beta}\rvert$	stands for	$\mathsf{len}_{(\alpha\to\beta)\to\alpha}\, \mathbf{A}_{\alpha\to\beta}$.
$+\!+_{(\alpha\to\beta)\to(\alpha\to\beta)\to(\alpha\to\beta)}$	stands for	$\mathrm{I}\, f : [\beta] \to [\beta] \to [\beta]\ .$ $\forall\, t : [\beta]\ .\ f\,[\,]_{\alpha\to\beta}\, t = t \wedge$ $\forall\, x : \beta,\, s, t : [\beta]\ .\ f\,(x :: s)\, t = (x :: f\, s\, t)$.

Table 12: Notational Definitions for Sequences

12 Monoid theory applied to strings

In this section we will show how the machinery of our monoid theory formalization can be applied to a theory of strings over an abstract alphabet. A string over an alphabet A is a finite sequence of values from A. The finite sequence s can be represented as a partial function $s : \mathbb{N} \to A$ such that, for some $n \in \mathbb{N}$, $s(m)$ is defined iff $m < n$.

In Table 12 we introduce compact notation for finite (and infinite) sequences represented in this manner. The notation requires a system of natural numbers as defined in Chapter 11 of [20]. We also introduce some special notation for strings in Table 13.

The development COF-dev-2 of the theory COF presented in Chapter 13 of [20] includes a system of natural numbers [20, Proposition 13.11]. Therefore, we can define a theory of strings as an extension of COF plus a base type A that represents an abstract alphabet:

$(\mathbf{X}_{R\to A}\mathbf{Y}_{R\to A})$	stands for	$\mathbf{X}_{R\to A}$ cat $\mathbf{Y}_{R\to A}$.
$(\mathbf{S}_{\{R\to A\}}\mathbf{T}_{\{R\to A\}})$	stands for	$\mathbf{S}_{\{R\to A\}}$ set-cat $\mathbf{T}_{\{R\to A\}}$.
$\left(\underset{\mathbf{i}=\mathbf{M}_R}{\overset{\mathbf{N}_R}{\mathsf{cat}}}\ \mathbf{A}_{R\to A}\right)$	stands for	iter-cat$_{R\to R\to(R\to(R\to A))\to(R\to A)}$ $\mathbf{M}_R\,\mathbf{N}_R\,(\lambda\,\mathbf{i} : R\,.\,\mathbf{A}_{R\to A})$.

Table 13: Notational Definitions for Monoids: Special Notation

Theory Extension 12.1 (Strings)**.**

Name: STR.

Extends COF.

New base types: A.

New constants:

New axioms:

Since STR is an extension of COF, we can assume that STR-1 is a development of STR that contains the 7 definitions of COF-dev-2 named as COF-Def1, ..., COF-Def7 and the 22 theorems of COF-dev-2 named as COF-Thm1, ..., COF-Thm22. We can extend STR-1 as follows to include the basic definitions and theorems of strings:

Development Extension 12.2 (Strings 2)**.**

Name: STR-2.

Extends STR-1.

New definitions and theorems:

Def10: $\mathsf{str}_{\{R\to A\}} = [A]$ (string quasitype).

Def11: $\epsilon_{R\to A} = [\,]_{R\to A}$ (empty string).

Def12: $\mathsf{cat}_{((R\to A)\times(R\to A))\to(R\to A)} = {+\!+}_{(R\to A)\to(R\to A)\to(R\to A)}$ (concatenation).

Thm33: $\forall\, x : \mathsf{str}\,.\,\epsilon x = x\epsilon = x$ (ϵ is an identity element).

Thm34: $\forall\, x, y, z : \mathsf{str}\,.\,x(yz) = (xy)z$ (cat is associative).

Def10–Def12 utilize the compact notation introduced in Table 12 and Thm33–Thm34 utilize the compact notation introduced in Table 13.

We can define a development translation from MON-over-COF to STR-2 as follows:

Development Translation Definition 12.3 (MON-over-COF to STR-2)**.**

Name: MON-over-COF-to-STR-2.

Source development: MON-over-COF.

Target development: STR-2.

Base type mapping:

1. $R \mapsto R$.
2. $M \mapsto \mathsf{str}_{\{R \to A\}}$.

Constant mapping:

1. $0_R \mapsto 0_R$.

   ⋮

10. $\mathsf{lub}_{R \to \{R\} \to o} \mapsto \mathsf{lub}_{R \to \{R\} \to o}$.
11. $\cdot_{(M \times M) \to M} \mapsto \mathsf{cat}_{((R \to A) \times (R \to A)) \to (R \to A)}$.
12. $\mathsf{e}_M \mapsto \epsilon_{R \to A}$.

MON-over-COF-to-STR-2 has one obligation of the first kind for the mapped base type M, which is clearly valid since $\mathsf{str}_{\{R \to A\}}$ is nonempty. MON-over-COF-to-STR-2 has 12 obligations of the second kind for the 12 mapped constants. The first 10 are trivially valid. The last 2 are valid by Def12 and Def11, respectively. And MON-over-COF-to-STR-2 has 20 obligations of the third kind for the 20 axioms of MON-over-COF. The first 18 are trivially valid. The last 2 are valid by Thm34 and Thm33, respectively. Therefore, MON-over-COF-to-STR-2 is a development morphism from the theory MON-over-COF to the development STR-2 by the Morphism Theorem [20, Theorem 14.16].

The development morphism MON-over-COF-to-STR-2 allows us to transport definitions and theorems about monoids to the development STR-2. Here are five examples transported as a group:

Group Transportation 12.4 (Transport to STR-2).

Name: monoid-machinery-via-MON-over-COF-1-to-STR-2.

Source development: MON-over-COF-1.

Target development: STR-2.

Development morphism: MON-over-COF-to-STR-2.

Definitions and theorems:

Thm1: $\mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$
(models of MON define monoids).

Def3: $\odot_{(\{M\}\times\{M\})\to\{M\}} = \mathsf{set\text{-}op}_{((M\times M)\to M)\to((\{M\}\times\{M\})\to\{M\})}\ \cdot$
(set product).

Def4: $\mathsf{E}_{\{M\}} = \{\mathsf{e}_M\}$ (set identity element).

Thm12 (Thm1-via-MON-to-set-monoid):
$\mathsf{MONOID}(U_{\{\{M\}\}}, \odot_{(\{M\}\times\{M\})\to\{M\}}, \mathsf{E}_{\{M\}})$
(set monoids are monoids).

Def9: $\mathsf{prod}_{R\to R\to(R\to M)\to M} =$
$\mathrm{I}\, f : Z_{\{R\}} \to Z_{\{R\}} \to (Z_{\{R\}} \to M) \to M\ .$
$\forall\, m, n : Z_{\{R\}},\ g : Z_{\{R\}} \to M\ .\ f\, m\, n\, g \simeq$
$(m > n \mapsto \mathsf{e} \mid (f\, m\, (n-1)\, g) \cdot (g\, n))$ (iterated product).

Transported definitions and theorems:

Thm35 (Thm1-via-MON-over-COF-to-STR-2):
$\mathsf{MONOID}(\mathsf{str}_{\{R\to A\}}, \mathsf{cat}_{((R\to A)\times(R\to A))\to(R\to A)}, \epsilon_{R\to A})$
(strings form a monoid).

Def13 (Def3-via-MON-over-COF-to-STR-2):
$\mathsf{set\text{-}cat}_{(\{R\to A\}\times\{R\to A\})\to\{R\to A\}} =$
$\mathsf{set\text{-}op}_{(((R\to A)\times(R\to A))\to(R\to A))\to((\{R\to A\}\times\{R\to A\})\to\{R\to A\})}\ \mathsf{cat}$
(set concatenation).

Def14 (Def4-via-MON-over-COF-to-STR-2):
$\mathsf{E}_{\{R\to A\}} = \{\epsilon_{R\to A}\}$ (set identity element).

Thm36 (Thm12-via-MON-over-COF-1-to-STR-2):
$\mathsf{MONOID}(\mathcal{P}(\mathsf{str}_{\{R\to A\}}), \mathsf{set\text{-}cat}_{(\{R\to A\}\times\{R\to A\})\to\{R\to A\}}, \mathsf{E}_{\{R\to A\}})$
(string sets form a monoid).

Def15 (Def9-via-MON-over-COF-1-to-STR-2):
$\mathsf{iter\text{-}cat}_{R\to R\to(R\to(R\to A))\to(R\to A)} =$
$\mathrm{I}\, f : Z_{\{R\}} \to Z_{\{R\}} \to (Z_{\{R\}} \to (R \to A)) \to (R \to A)\ .$
$\forall\, m, n : Z_{\{R\}},\ g : Z_{\{R\}} \to (R \to A)\ .\ f\, m\, n\, g \simeq$
$(m > n \mapsto \epsilon \mid (f\, m\, (n-1)\, g)\ \mathsf{cat}\ (g\, n))$
(iterated concatenation).

New target development: STR-3.

New development morphism: MON-over-COF-1-to-STR-3.

Notation for the application of

$$\mathsf{set\text{-}cat}_{(\{R\to A\}\times\{R\to A\})\to\{R\to A\}}$$

and

$$\mathsf{iter\text{-}cat}_{R\to R\to(R\to(R\to A))\to(R\to A)}$$

are defined in Table 13.

13 Related work

As we have seen, a theory (or development) graph provides an effective architecture for formalizing a body of mathematical knowledge. It is especially useful for creating a large library of formal mathematical knowledge that, by necessity, must be constructed in parallel by multiple developers. The library is built in parts by separate development teams and then the parts are linked together by morphisms. Mathematical knowledge is organized as a theory graph in several proof assistants and logical frameworks including Ergo [44], IMPS [22, 24], Isabelle [5], LF [53], MMT [52], and PVS [47]. Theory graphs are also employed in several software specification and development systems including ASL [57], CASL [3, 4], EHDM [55], Hets [40], IOTA [41], KIDS [58], OBJ [27], and Specware [59].

Simple type theory in the form of Church's type theory is a popular logic for formal mathematics. There are several proof assistants that implement versions of Church's type theory including HOL [29], HOL Light [31], IMPS [23, 24], Isabelle/HOL [48], ProofPower [51], PVS [46], and TPS [2]. As we mentioned in Section 1, the IMPS proof assistant is especially noteworthy here since it implements LUTINS [13, 14, 15], a version of Church's type theory that admits undefined expressions and is closely related to Alonzo.

In recent years, there has been growing interest in formalizing mathematics within dependent logics. Several proof assistants and programming languages are based on versions of dependent type theory including Agda [7, 45], Automath [43], Epigram [11], F* [12], Idris [34], Lean [10], Nuprl [9], and Rocq [54]. So which type theory is better for formal mathematics, simple type theory or dependent type theory? This question has become hotly contested. We hope that the reader will see our formalization of monoid theory in Alonzo as evidence for the efficacy of simple type theory as a logical basis for formal mathematics. The reader might also be interested in looking at these recent papers that advocate for simple type theory: [6, 49, 50].

Since monoid theory is a relatively simple subject, there have not been many attempts to formalize it by itself, but there have been several formalizations of group theory. Here are some examples: [26, 28, 35, 56, 60, 61].

There are two other important alternatives to the standard approach to formal mathematics. The first is Tom Hales' *formal abstracts in mathematics* project [25, 30] in which proof assistants are used to the create *formal abstracts*, which are formal presentations of mathematical theorems without formal proofs. The second is Michael Kohlhase's *flexiformal mathematics* [33, 36, 37] initiative in which mathematics is a mixture of traditional and formal mathematics and proofs can be either traditional or formal. The alternative approach we offer is similar to both of these approaches, but there are important differences. The formal abstracts approach seeks to formalize *collections of theorems* without proofs using proof assistants, while we seek to formalize *theory graphs* with either traditional or formal proofs using supporting software that can be much simpler than a proof assistant. The objective of the flexiformal mathematics approach is to give the user the flexibility to produce mathematics with varying degrees of formality. In contrast, our approach is to produce mathematics that is fully formal except for proofs.

14 Conclusion

The developments and development morphisms presented in Sections 4–12 form the development graph G_{mon} shown in Figure 1. The development graph shows all the development morphisms that we have explicitly defined (7 inclusions via theory extension modules and 10 noninclusions via theory and development definition modules) plus an implicit inclusion from COM-MON to COM-MON-over-COF. A development morphism that is an inclusion is designated by a $\hookrightarrow$ arrow and a noninclusion is designated by a $\rightarrow$ arrow. There are many, many more useful development morphisms that are not shown in G_{mon}, including implicit inclusions and a vast number of development morphisms into the theory COF.

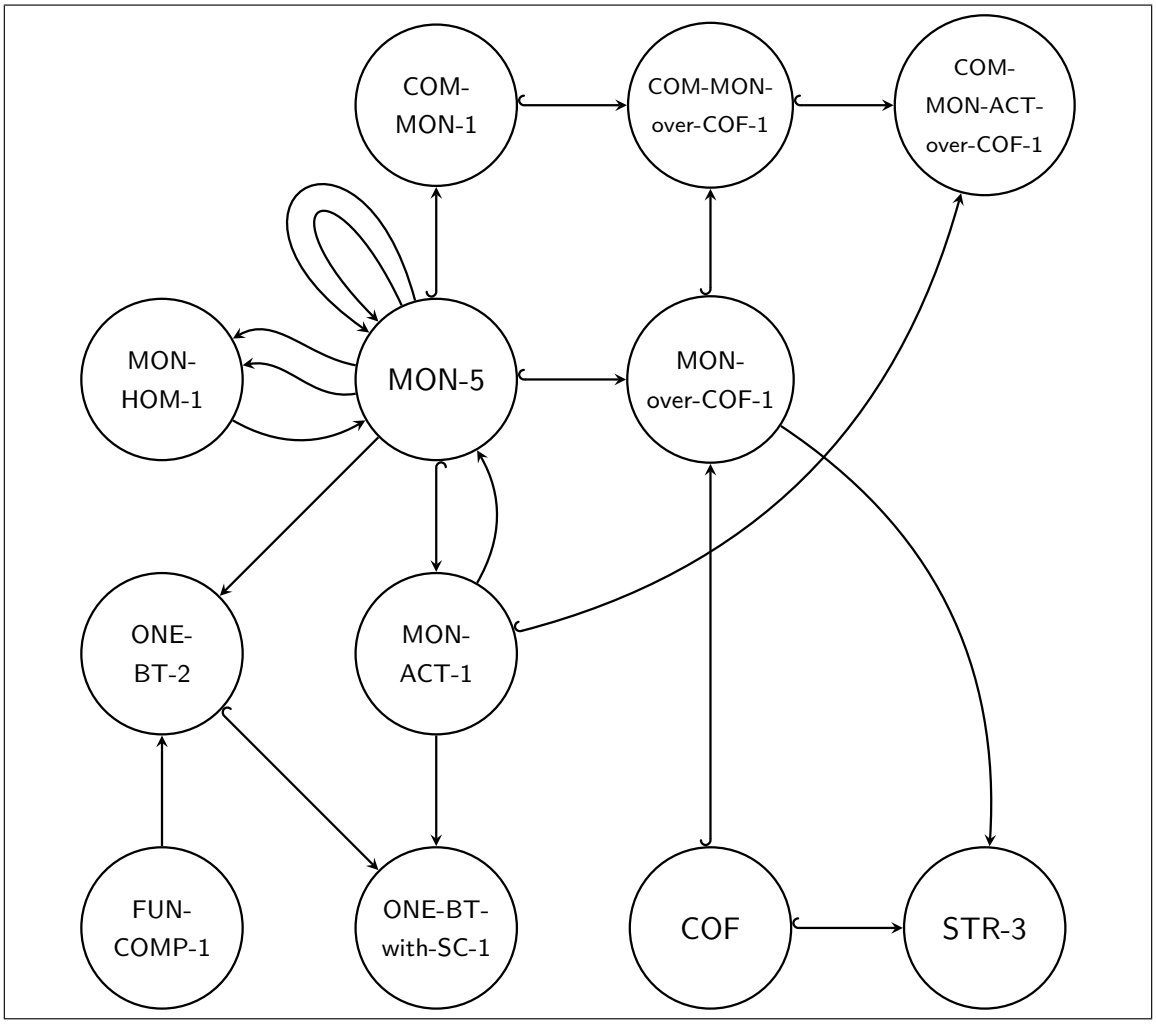

Figure 1: The Monoid Theory Development Graph

The construction of G_{mon} illustrates how a body of mathematical knowledge can be formalized in Alonzo as a development graph in accordance with the little theories method and the alternative approach. G_{mon} could be extended to include other mathematical concepts related to monoids such as categories. It could be incorporated in a development graph that formalizes a more extensive body of mathematical knowledge. And it could also be used as a foundation for building a formalization of group theory. This would be done by lifting each development D of a theory T that extends MON to a development D' of a theory T' that extends a theory GRP of groups obtained by adding an inverse operation to MON. The lifting of D to D' would include constructing inclusions from MON to GRP and from T to T' via

theory extensions.

The formalization of monoid theory we have presented demonstrates three things. First, it demonstrates the power of the little theories method. The formalization is largely free of redundancy since each mathematical topic is articulated in just one development D, the development for the little theory that is optimal for the topic in level of abstraction and choice of vocabulary. If we create a translation Φ from D to another development D' and prove that Φ is a morphism, then we can freely transport the definitions and theorems of D to D' via Φ. That is, an abstract concept or fact that has been validated in D can be translated to a concrete instance of the concept or fact that is automatically validated in D' provided the translation is a morphism. (This is illustrated by our use of the development morphism MON-over-COF-to-STR-2 to transport definitions and theorems about monoids to a development about strings.) As the result, the same concept or fact can appear in many places in the theory graph but under different assumptions and involving different vocabulary. (For example, the notion of a submonoid represented by the constant $\mathsf{submonoid}_{\{M\}\to o}$ defined in MON-1 appears in ONE-BT-2 as the notion of a transformation monoid represented by the constant $\mathsf{trans\text{-}monoid}_{\{S\to S\}\to o}$.) In short, we have shown how the little theories method enables mathematical knowledge to be formalized to maximize clarity and minimize redundancy.

Second, the formalization demonstrates that the alternative approach to formal mathematics (with traditional and formal proofs) has two advantages over the standard approach (with only formal proofs): (1) communication is more effective since the user has greater freedom of expression and (2) formalization is easier since the approach offers greater accessibility. The standard approach is done with the help of a proof assistant and all proofs are formal and mechanically checked. Proof assistants are consequently very complex and notoriously difficult to learn how to use. Traditional proofs are easier to read and write than formal proofs and are better suited for communicating the ideas behind proofs. Moreover, since the alternative approach does not require a facility for developing and checking formal proofs, it can be done with software support that is much simpler and easier to use than a proof assistant. (In this paper, our software support was just a set of LaTeX macros and environments.)

Third, the formalization demonstrates that Alonzo is well suited for expressing and reasoning about mathematical ideas. The simple type theory machinery of Alonzo — function and product types, function application and abstraction, definite description, and ordered pairs — enables mathematical expressions to be formulated in a direct and natural manner. It also enables almost every single mathematical structure or set of similar mathematical structures to be specified by an Alonzo development. (For example, the development ONE-BT-2 specifies the set of math-

ematical structures consisting of a set S and the set $S \to S$ of transformations on S.) The admission of undefined expressions in Alonzo enables statements involving partial and total functions and definite descriptions to be expressed directly, naturally, and succinctly. (For example, if $M = (m, \cdot, e)$ is a monoid, the operation that makes a submonoid $m' \subseteq m$ of M a monoid itself is exactly what is expected: the partial function that results from restricting $\cdot$ to $m' \times m'$.) And the notational definitions and conventions employed in Alonzo enables mathematical expressions to be presented with largely the same notation that is used mathematical practice. (For example, Thm33: $\forall\, x : \mathsf{str}\ .\ \epsilon x = x\epsilon = x$, that states ϵ is an identity element for concatenation, is written just as one would expect it to be written in mathematical practice.)

We believe that this paper achieves our overarching goal: To demonstrate that mathematical knowledge can be very effectively formalized in a version of simple type theory like Alonzo using the little theories method and the alternative approach to formal mathematics. We also believe that it illustrates the benefits of employing the little theories method, the alternative approach, and Alonzo in formal mathematics.

A Validation of definitions and theorems

Let $D = (T, \Xi)$ be a development where T is the bottom theory of the development and $\Xi = [P_1, \ldots, P_n]$ is the list of definition and theorem packages of the development. For each i with $1 \leq i \leq n$, P_i has the form $(p, \mathbf{c}_\alpha, \mathbf{A}_\alpha, \pi)$ if P_i is a definition package and has the form $P_i = (p, \mathbf{A}_o, \pi)$ if P_i is a theorem package. Define $T_0 = T$ and, for all i with $0 \leq i \leq n-1$, define $T_{i+1} = T[P_{i+1}]$ if P_{i+1} is a definition package and $T_{i+1} = T_i$ if P_{i+1} is a theorem package. In the former case, π is a proof that $\mathbf{A}_\alpha{\downarrow}$ is valid in T_i, and in the latter case, π is a proof that $\mathbf{A}_o$ is valid in T_i. These proofs may be either traditional or formal. See Chapter 12 of [20] for further details.

The validation proofs for the definitions and theorems of a development are not included in the modules we have used to construct developments and to transport definitions and theorems. Instead, we give in this appendix, for each of the definitions and theorems in the developments defined in Sections 4–12, a traditional proof that validates the definition or theorem. The proofs are almost entirely straightforward. The proofs extensively reference the axioms, rules of inference, and metatheorems of $\mathfrak{A}$, the formal proof system for Alonzo presented in [20]. These are legitimate to use since $\mathfrak{A}$ is sound by the Soundness Theorem [20, Theorem B.11].

A.1 Development of MON

1. Thm1: $\mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$ (models of MON define monoids).

   Proof of the theorem. Let $T = (L, \Gamma)$ be MON. We must show

   $$(\star)\ T \vDash \mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M).$$

   $$\Gamma \vDash U_{\{M\}}\downarrow \tag{1}$$
   $$\Gamma \vDash U_{\{M\}} \neq \emptyset_{\{M\}} \tag{2}$$
   $$\Gamma \vDash \cdot_{(M\times M)\to M} \downarrow (U_{\{M\}} \times U_{\{M\}}) \to U_{\{M\}} \tag{3}$$
   $$\Gamma \vDash \mathsf{e}_M \downarrow U_{\{M\}} \tag{4}$$
   $$\Gamma \vDash \forall\, x, y, z : U_{\{M\}}\ .\ x \cdot (y \cdot z) = (x \cdot y) \cdot z \tag{5}$$
   $$\Gamma \vDash \forall\, x : U_{\{M\}}\ .\ \mathsf{e} \cdot x = x \tag{6}$$
   $$\Gamma \vDash \mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M) \tag{7}$$

   (1) and (2) follow from parts 1 and 2, respectively, of Lemma B.1; (3) follows from [20, Axiom A5.2] and parts 8–10 of Lemma B.1; (4) follows from [20, Axiom A5.2] and part 8 of Lemma B.1; (5) and (6) follow from Axioms 1 and 2, respectively, of T and part 5 of Lemma B.1; and (7) follows from (1)–(6) and the definition of MONOID in Table 10. Therefore, $(\star)$ holds. □

2. Thm2: $\mathsf{TOTAL}(\cdot_{(M\times M)\to M})$ ($\cdot$ is total).

   Proof of the theorem. Let $\mathbf{A}_o$ be

   $$\forall\, x : M \times M\ .\ (\cdot_{(M\times M)\to M}\, x)\downarrow$$

   and $T = (L, \Gamma)$ be MON. TOTAL is the abbreviation introduced by the notational definition given in Table 7, and so $\mathsf{TOTAL}(\cdot_{(M\times M)\to M})$ stands for $\mathbf{A}_o$. Thus we must show $(\star)\ T \vDash \mathbf{A}_o$.

   $$\Gamma \vDash (x : M \times M)\downarrow \tag{1}$$
   $$\Gamma \vDash (x : M \times M) = (\mathsf{fst}\, x, \mathsf{snd}\, x) \tag{2}$$
   $$\Gamma \vDash (\mathsf{fst}\, x)\downarrow \wedge (\mathsf{snd}\, x)\downarrow \tag{3}$$
   $$\Gamma \vDash (\mathsf{fst}\, x) \cdot ((\mathsf{fst}\, x) \cdot (\mathsf{snd}\, x)) = ((\mathsf{fst}\, x) \cdot (\mathsf{fst}\, x)) \cdot (\mathsf{snd}\, x) \tag{4}$$
   $$\Gamma \vDash ((\mathsf{fst}\, x) \cdot (\mathsf{snd}\, x))\downarrow \tag{5}$$
   $$\Gamma \vDash (\cdot_{(M\times M)\to M}\, (\mathsf{fst}\, x, \mathsf{snd}\, x))\downarrow \tag{6}$$
   $$\Gamma \vDash \mathbf{A}_o \tag{7}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from (1) and [20, Axiom A7.4] by Universal Instantiation [20, Theorem A.14]; (3) follows from (2) by [20, Axioms A5.5, A7.2, and A7.3]; (4) follows from (3) and Axiom 1 of T by Universal Instantiation [20, Theorem A.14]; (5) follows from (4) by [20, Axioms A5.4 and A5.10]; (6) follows from (5) by notational definition; and (7) follows from (6) by Universal Generalization [20, Theorem A.30] using (2) and the fact that $(x : (M \times M))$ is not free in Γ since Γ is a set of sentences. □

3. Thm3: $\forall\, x : M \,.\, (\forall\, y : M \,.\, x \cdot y = y \cdot x = y) \Rightarrow x = \mathsf{e}$ (uniqueness of identity element).

Proof of the theorem. Let $\mathbf{A}_o$ be

$$\forall\, y : M \,.\, (x : M) \cdot y = y \cdot (x : M) = y$$

and $T = (L, \Gamma)$ be MON. We must show $(\star)$ $T \vDash \forall\, x : M \,.\, \mathbf{A}_o \Rightarrow x = \mathsf{e}$.

$$\Gamma \cup \{\mathbf{A}_o\} \vDash \mathsf{e}{\downarrow} \tag{1}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash (x : M){\downarrow} \tag{2}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash (x : M) \cdot \mathsf{e} = \mathsf{e} \cdot (x : M) = \mathsf{e} \tag{3}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \mathsf{e} \cdot (x : M) = (x : M) \cdot \mathsf{e} = (x : M) \tag{4}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash (x : M) = \mathsf{e} \tag{5}$$
$$\Gamma \vDash \mathbf{A}_o \Rightarrow (x : M) = \mathsf{e} \tag{6}$$
$$\Gamma \vDash \forall\, x : M \,.\, \mathbf{A}_o \Rightarrow x = \mathsf{e} \tag{7}$$

(1) follows from constants always being defined by [20, Axiom A5.2]; (2) follows from variables always being defined by [20, Axiom A5.1]; (3) follows (1) and $\mathbf{A}_o$ by Universal Instantiation [20, Theorem A.14]; (4) follows (2) and Axiom 2 of T by Universal Instantiation; (5) follows from (3) and (4) by the Equality Rules [20, Lemma A.13]; (6) follows from (5) by the Deduction Theorem [20, Lemma A.50]; and (7) follows from (6) by Universal Generalization [20, Theorem A.30] using the fact that $(x : M)$ is not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

4. Def1: $\mathsf{submonoid}_{\{M\}\to o} =$
$\lambda\, s : \{M\} \,.\, s \neq \emptyset_{\{M\}} \wedge (\cdot|_{s\times s}{\downarrow}\,(s \times s) \to s) \wedge \mathsf{e} \in s$ (submonoid).

Proof that RHS is defined. Let $\mathbf{A}_{\{M\}\to o}$ be the RHS of Def1. We must show that MON $\vDash \mathbf{A}_{\{M\}\to o}{\downarrow}$. This follows immediately from function abstractions always being defined by [20, Axiom A5.11]. □

5. Thm4: $\forall\, s : \{M\}\ .\ \mathsf{submonoid}\, s \Rightarrow \mathsf{MONOID}(s, \cdot|_{s\times s}, \mathsf{e})$ (submonoids are monoids).

Proof of the theorem. Let $\mathbf{A}_o$ be

$$\mathsf{submonoid}(s)$$

and $T = (L, \Gamma)$ be MON extended by Def1. We must show

$$(\star)\ T \vDash \forall\, s : \{M\}\ .\ \mathbf{A}_o \Rightarrow \mathsf{MONOID}(s, \cdot|_{(s\times s)}, \mathsf{e}).$$

$$\Gamma \cup \{\mathbf{A}_o\} \vDash s_{\{M\}}\!\downarrow \tag{1}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash s \neq \emptyset_{\{M\}} \tag{2}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \cdot|_{(s\times s)} \downarrow (s \times s) \to s \tag{3}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \mathsf{e} \in s \tag{4}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \mathsf{e} \downarrow s \tag{5}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \forall\, x, y, z : s\ .\ \cdot|_{(s\times s)}\,(x, \cdot|_{(s\times s)}\,(y, z)) = \cdot|_{(s\times s)}\,(\cdot|_{(s\times s)}\,(x, y), z) \tag{6}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \forall\, x : s\ .\ \cdot|_{(s\times s)}\,(\mathsf{e}, x) = \cdot|_{(s\times s)}\,(x, \mathsf{e}) = x \tag{7}$$
$$\Gamma \cup \{\mathbf{A}_o\} \vDash \mathsf{MONOID}(s, \cdot|_{(s\times s)}, \mathsf{e}) \tag{8}$$
$$\Gamma \vDash \mathbf{A}_o \Rightarrow \mathsf{MONOID}(s, \cdot|_{(s\times s)}, \mathsf{e}) \tag{9}$$
$$\Gamma \vDash \forall\, s : \{M\}\ .\ \mathbf{A}_o \Rightarrow \mathsf{MONOID}(s, \cdot|_{(s\times s)}, \mathsf{e}) \tag{10}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2), (3), and (4) follow directly from Def1; (5) follows from [20, Axiom A5.2] and (4); (6) and (7) follow from Thm1, $\cdot|_{(s\times s)} \sqsubseteq \cdot_{(M\times M)\to M}$, and the fact that $\cdot|_{(s\times s)}$ is total on $s \times s$ by Thm2; (8) follows from (1)–(3) and (5)–(7) by the definition of MONOID in Table 10; (9) follows from (8) by the Deduction Theorem [20, Theorem A.50]; and (10) follows from (9) by Universal Generalization [20, Theorem A.30] using the fact that $(s : \{M\})$ is not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

6. Thm5: $\mathsf{submonoid}\,\{\mathsf{e}\}$ (minimum submonoid).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON extended by Def1. We

must show $(\star)$ $T \models \mathsf{submonoid}\,\{\mathsf{e}\}$.

$$\Gamma \models \mathsf{e} \in \{\mathsf{e}\} \tag{1}$$
$$\Gamma \models \{\mathsf{e}\} \neq \emptyset_{\{M\}} \tag{2}$$
$$\Gamma \models \mathsf{e} \cdot \mathsf{e} = \mathsf{e} \tag{3}$$
$$\Gamma \models \cdot|_{\{\mathsf{e}\}\times\{\mathsf{e}\}} \downarrow (\{\mathsf{e}\} \times \{\mathsf{e}\}) \to \{\mathsf{e}\} \tag{4}$$
$$\Gamma \models \mathsf{submonoid}\,\{\mathsf{e}\} \tag{5}$$

(1) is trivial; (2) follows from (1) because $\{\mathsf{e}\}$ has at least one member; (3) follows from Axiom 2 of T by Universal Instantiation [20, Theorem A.14]; (4) follows directly from (1), (3), and the fact that the only member of $\{\mathsf{e}\}$ is e; and (5) follows from (1), (2), (4), and Def1. Therefore, $(\star)$ holds. □

7. Thm6: $\mathsf{submonoid}\,U_{\{M\}}$ (maximum submonoid).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON extended by Def1. We must show $(\star)$ $T \models \mathsf{submonoid}\,U_{\{M\}}$.

$$\Gamma \models \mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}) \tag{1}$$
$$\Gamma \models U_{\{M\}} \neq \emptyset_{\{M\}} \wedge \mathsf{e} \in U_{\{M\}} \tag{2}$$
$$\Gamma \models \cdot|_{U_{\{M\}}\times U_{\{M\}}} \downarrow (U_{\{M\}} \times U_{\{M\}}) \to U_{\{M\}} \tag{3}$$
$$\Gamma \models \mathsf{submonoid}\,U_{\{M\}} \tag{4}$$

(1) is Thm1; (2) follows immediately from (1); (3) follows from (1) by part 12 of Lemma B.1; and (4) follows from (1), (2), (3), and Def1. Therefore, $(\star)$ holds. □

8. Def2: $\cdot^{\mathrm{op}}_{(M\times M)\to M} = \lambda\, p : M \times M \,.\, (\mathsf{snd}\, p) \cdot (\mathsf{fst}\, p)$ (opposite of $\cdot$).

Proof that RHS is defined. Similar to the proof that the RHS of Def1 is defined. □

9. Thm7: $\forall\, x, y, z : M \,.\, x \cdot^{\mathrm{op}} (y \cdot^{\mathrm{op}} z) = (x \cdot^{\mathrm{op}} y) \cdot^{\mathrm{op}} z$ ($\cdot^{\mathrm{op}}$ is associative).

Proof of the theorem. Let $\mathbf{A}_o$ be

$$x \cdot^{\mathrm{op}} (y \cdot^{\mathrm{op}} z) = (x \cdot^{\mathrm{op}} y) \cdot^{\mathrm{op}} z$$

and $T = (L, \Gamma)$ be MON extended by Def2. We must show

$$(\star)\ T \models \forall\, x, y, z : M \,.\, \mathbf{A}_o.$$

$$\Gamma \models (x : M)\!\downarrow \wedge (y : M)\!\downarrow \wedge (z : M)\!\downarrow \tag{1}$$
$$\Gamma \models (z \cdot y) \cdot x = z \cdot (y \cdot x) \tag{2}$$
$$\Gamma \models \mathbf{A}_o \tag{3}$$
$$\Gamma \models \forall\, x, y, z : M \;.\; \mathbf{A}_o \tag{4}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from (1) and Axiom 1 of T by Universal Instantiation [20, Theorem A.14] and the Equality Rules [20, Theorem A.13]; (3) follows from Lemma B.2 and (2) by repeated applications of Rule R2′ [20, Lemma A.2] using ($\star\star$) the fact that $(x : M)$, $(y : M)$, and $(z : M)$ are not free in Γ since Γ is a set of sentences; and (4) follows from (3) by Universal Generalization [20, Theorem A.30] again using ($\star\star$). Therefore ($\star$) holds. □

10. Thm8: $\forall\, x : M \;.\; \mathsf{e} \cdot^{\mathrm{op}} x = x \cdot^{\mathrm{op}} \mathsf{e} = x$
(e is an identity element with respect to $\cdot^{\mathrm{op}}$).

Proof of the theorem. Let $\mathbf{A}_o$ be

$$\mathsf{e} \cdot^{\mathrm{op}} x = x \cdot^{\mathrm{op}} \mathsf{e} = x$$

and $T = (L, \Gamma)$ be MON extended by Def2. We must show

($\star$) $T \models \forall\, x : M \;.\; \mathbf{A}_o$.

$$\Gamma \models (x : M)\!\downarrow \tag{1}$$
$$\Gamma \models x \cdot \mathsf{e} = \mathsf{e} \cdot x = x \tag{2}$$
$$\Gamma \models \mathbf{A}_o \tag{3}$$
$$\Gamma \models \forall\, x : M \;.\; \mathbf{A}_o \tag{4}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from (1) and Axiom 2 of T by Universal Instantiation [20, Theorem A.14] and the Equality Rules [20, Theorem A.13]; (3) follows from Lemma B.2 and (2) by repeated applications of Rule R2′ [20, Lemma A.2] using ($\star\star$) the fact that $(x : M)$ is not free in Γ since Γ is a set of sentences; and (4) follows from (3) by Universal Generalization [20, Theorem A.30] again using ($\star\star$). Therefore ($\star$) holds. □

11. Def3: $\odot_{(\{M\}\times\{M\})\to\{M\}} = \mathsf{set\text{-}op}_{((M\times M)\to M)\to((\{M\}\times\{M\})\to\{M\})}$.
(set product).

Proof that RHS is defined. Let $\mathbf{A}_{(\{M\}\times\{M\})\to\{M\}}$ be the RHS of Def3. We must show $(\star)$ $\mathsf{MON} \vDash \mathbf{A}_{(\{M\}\times\{M\})\to\{M\}}\!\downarrow$. Since constants are always defined by [20, Axiom A5.2], $\mathbf{A}_{(\{M\}\times\{M\})\to\{M\}}$ beta-reduces to a function abstraction by [20, Axiom A4]. Since every function abstraction is defined by [20, Axiom A5.11], we have $(\star)$ by Quasi-Equality Substitution [20, Lemma A.2]. □

12. Def4: $\mathsf{E}_{\{M\}} = \{\mathsf{e}_M\}$ (set identity element).

Proof that RHS is defined. We must show $(\star)$ $\mathsf{MON} \vDash \{\mathsf{e}_M\}\!\downarrow$. Now $\{\mathsf{e}_M\}$ stands for

$$(\lambda\, x_1 : M \,.\, \lambda\, x : M \,.\, x = x_1)\,(\mathsf{e}_M).$$

Since constants are always defined by [20, Axiom A5.2], $\{\mathsf{e}_M\}$ beta-reduces to

$$\lambda\, x : M \,.\, x = \mathsf{e}_M$$

by [20, Axiom A4]. Since every function abstraction is defined by [20, Axiom A5.11], we have $(\star)$ by Quasi-Equality Substitution [20, Lemma A.2]. □

13. Thm9: $\forall\, x, y, z : \{M\} \,.\, x \odot (y \odot z) = (x \odot y) \odot z$ ($\odot$ is associative).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON extended by Def3. We must show

$$(\star)\ T \vDash \forall\, x, y, z : \{M\} \,.\, x \odot (y \odot z) = (x \odot y) \odot z.$$

$$\Gamma \vDash (x : \{M\})\!\downarrow \wedge\, (y : \{M\})\!\downarrow \wedge\, (z : \{M\})\!\downarrow \tag{1}$$

$$\Gamma \vDash x \odot (y \odot z) = \{d : M \mid \exists\, a : x,\, b : y,\, c : z \,.\, d = a \cdot (b \cdot c)\} \tag{2}$$

$$\Gamma \vDash (x \odot y) \odot z = \{d : M \mid \exists\, a : x,\, b : y,\, c : z \,.\, d = (a \cdot b) \cdot c\} \tag{3}$$

$$\Gamma \vDash x \odot (y \odot z) = (x \odot y) \odot z \tag{4}$$

$$\Gamma \vDash \forall\, x, y, z : \{M\} \,.\, x \odot (y \odot z) = (x \odot y) \odot z \tag{5}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (3) follow from (1) and Def3; (4) follows from (2) and (3) by Axiom 1 of T; and (5) follows from (4) by Universal Generalization [20, Theorem A.30] using the fact that x, y, and z are not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

14. Thm10: $\forall\, x : \{M\}\ .\ \mathsf{E} \odot x = x \odot \mathsf{E} = x$
(E is an identity element with respect to $\odot$).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON extended by Def3 and Def4. We must show

$$(\star)\ T \models \forall\, x : \{M\}\ .\ \mathsf{E} \odot x = x \odot \mathsf{E} = x.$$

$$\Gamma \models (x : \{M\})\downarrow \tag{1}$$
$$\Gamma \models \mathsf{E}\downarrow \tag{2}$$
$$\Gamma \models \mathsf{E} \odot x = \{b : M \mid \exists\, a : x\ .\ b = \mathsf{e} \cdot a\} \tag{3}$$
$$\Gamma \models x \odot \mathsf{E} = \{b : M \mid \exists\, a : x\ .\ b = a \cdot \mathsf{e}\} \tag{4}$$
$$\Gamma \models \mathsf{E} \odot x = x \odot \mathsf{E} = x \tag{5}$$
$$\Gamma \models \forall\, x : \{M\}\ .\ \mathsf{E} \odot x = x \odot \mathsf{E} = x \tag{6}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from constants always defined by [20, Axiom A5.2]; (3) and (4) follow from (1), (2), Def3, and Def4; (5) follows from (3) and (4) by Axiom 2 of T; and (6) follows from (5) by Universal Generalization [20, Theorem A.30] using the fact that x is not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

15. Thm11 (Thm1-via-MON-to-opposite-monoid):
$\mathsf{MONOID}(U_{\{M\}}, \cdot^{\mathrm{op}}_{(M\times M)\to M}, \mathsf{e}_M)$ (opposite monoids are monoids).

Proof of the theorem. Let T be the top theory of MON-1. We must show $T \models$ Thm11. We have previously proved $(\star)$ MON $\models$ Thm1. $\Phi =$ MON-to-opposite-monoid is a development morphism from MON to MON-1, and so $\widetilde{\Phi} = (\mu, \nu)$ is a theory morphism from MON to T. Thus $(\star)$ implies $T \models \nu(\mathsf{Thm1})$. Therefore, $T \models$ Thm11 since Thm11 $= \nu($Thm1$)$. □

16. Thm12 (Thm1-via-MON-to-set-monoid):
$\mathsf{MONOID}(U_{\{\{M\}\}}, \odot_{(\{M\}\times\{M\})\to\{M\}}, \mathsf{E}_{\{M\}})$
(set monoids are monoids).

Proof of the theorem. Similar to the proof of Thm11. □

A.2 Development of COM-MON

1. Thm13: $\mathsf{COM\text{-}MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M)$
(models of COM-MON define commutative monoids).

Proof of the theorem. Let $T = (L, \Gamma)$ be COM-MON. We must show

$$(\star)\ T \vDash \mathsf{COM\text{-}MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M).$$

$$\Gamma \vDash \mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M) \tag{1}$$
$$\Gamma \vDash \forall\, x, y : U_{\{M\}}\ .\ x \cdot y = y \cdot x \tag{2}$$

(1) follows from $\mathsf{MON} \leq T$ and the fact that Thm1 is a theorem of MON; and (2) follows from Axiom 3 of T and part 5 of Lemma B.1. Therefore, $(\star)$ follows from (1), (2), and the notational definition of COM-MONOID given in Table 10. □

2. Def5: $\leq_{M\to M\to o} = \lambda\, x, y : M\ .\ \exists\, z : M\ .\ x \cdot z = y$ (weak order).

   Proof that RHS is defined. Similar to the proof that the RHS of Def1 is defined. □

3. Thm14: $\forall\, x : M\ .\ x \leq x$ (reflexivity).

   Proof of the theorem. Let $T = (L, \Gamma)$ be COM-MON extended by Def5. We must show

   $$(\star)\ T \vDash \forall\, x : M\ .\ x \leq x.$$

   $$\Gamma \vDash (x : M)\!\downarrow \tag{1}$$
   $$\Gamma \vDash (x \leq x) \simeq (\exists\, z : M\ .\ x \cdot z = x) \tag{2}$$
   $$\Gamma \vDash x \cdot \mathsf{e} = x \tag{3}$$
   $$\Gamma \vDash \exists\, z : M\ .\ x \cdot z = x \tag{4}$$
   $$\Gamma \vDash x \leq x \tag{5}$$
   $$\Gamma \vDash \forall\, x : M\ .\ x \leq x \tag{6}$$

   (1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from Def5 and Extensionality [20, Axiom A3] using the Substitution Rule [20, Theorem A.31] and Beta-Reduction [20, Axiom A4]; (3) follows from (1) and Axiom 2 of T by Universal Instantiation [20, Theorem A.14]; (4) follows from (3) by Existential Generalization [20, Theorem A.51]; (5) follows from (2) and (4) by Rule R2′ [20, Lemma A.2]; and (6) follows from (5) by Universal Generalization [20, Theorem A.30] using the fact that x is not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

4. Thm15: $\forall\, x, y, z : M \,.\, (x \leq y \wedge y \leq z) \Rightarrow x \leq z$ (transitivity).

Proof of the theorem. Let $\mathbf{A}_o$ be $(x \leq y \wedge y \leq z)$, $\mathbf{B}_o$ be $x \cdot u = y$, and $\mathbf{C}_o$ be $y \cdot v = z$ (where these variables all have type M). Also let $T = (L, \Gamma)$ be COM-MON extended by Def5. We must show

$$(\star)\ T \vDash \forall\, x, y, z : M \,.\, \mathbf{A}_o \Rightarrow x \leq z.$$

$$\Gamma \cup \{\mathbf{B}_o, \mathbf{C}_o\} \vDash (x : M)\!\downarrow \wedge (y : M)\!\downarrow \wedge (z : M)\!\downarrow \wedge (u : M)\!\downarrow \wedge (v : M)\!\downarrow \tag{1}$$

$$\Gamma \cup \{\mathbf{B}_o, \mathbf{C}_o\} \vDash (x \cdot u) \cdot v = z \tag{2}$$

$$\Gamma \cup \{\mathbf{B}_o, \mathbf{C}_o\} \vDash (x \cdot u) \cdot v = x \cdot (u \cdot v) \tag{3}$$

$$\Gamma \cup \{\mathbf{B}_o, \mathbf{C}_o\} \vDash x \cdot (u \cdot v) = z \tag{4}$$

$$\Gamma \cup \{\mathbf{B}_o, \mathbf{C}_o\} \vDash \exists\, w : M \,.\, x \cdot w = z \tag{5}$$

$$\Gamma \cup \{\mathbf{B}_o\} \vDash (y \cdot v = z) \Rightarrow (\exists\, w : M \,.\, x \cdot w = z) \tag{6}$$

$$\Gamma \cup \{\mathbf{B}_o\} \vDash (\exists\, v : M \,.\, y \cdot v = z) \Rightarrow (\exists\, w : M \,.\, x \cdot w = z) \tag{7}$$

$$\Gamma \vDash (x \cdot u = y) \Rightarrow ((\exists\, v : M \,.\, y \cdot v = z) \Rightarrow (\exists\, w : M \,.\, x \cdot w = z)) \tag{8}$$

$$\Gamma \vDash (\exists\, u : M \,.\, x \cdot u = y) \Rightarrow ((\exists\, v : M \,.\, y \cdot v = z) \Rightarrow (\exists\, w : M \,.\, x \cdot w = z)) \tag{9}$$

$$\Gamma \vDash x \leq y \Rightarrow (y \leq z \Rightarrow x \leq z) \tag{10}$$

$$\Gamma \vDash \mathbf{A}_o \Rightarrow x \leq z \tag{11}$$

$$\Gamma \vDash \forall\, x, y, z : M \,.\, \mathbf{A}_o \Rightarrow x \leq z \tag{12}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from $\mathbf{B}_o$ and $\mathbf{C}_o$ by the Equality Rules [20, Lemma A.13]; (3) follows from Axiom 1 of T by Universal Instantiation [20, Theorem A.14]; (4) follows from (2) and (3) by the Equality Rules [20, Lemma A.13]; (5) follows from (1), (4), and Thm2 by Existential Generalization [20, Theorem A.51]; (6) and (8) follow from (5) and (7), respectively, by the Deduction Theorem [20, Theorem A.50]; (7) and (9) follow from (6) and (8), respectively, by Existential Instantiation [20, Theorem A.52]; (10) follows from (1), (9), and Def5 by Beta-Reduction [20, Axiom A4] and Alpha-Conversion [20, Theorem A.18]; (11) follows from (10) by the Tautology Rule [20, Corollary A.46]; and (12) follows from (11) by Universal Generalization [20, Theorem A.30] using the fact that x, y, and z are not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

A.3 Development of FUN-COMP

1. Thm16: $\forall\, f : A \to B,\, g : B \to C,\, h : C \to D \,.\, f \circ (g \circ h) = (f \circ g) \circ h$ ($\circ$ is associative).

Proof of the theorem. Let $\mathbf{A}_o$ be the theorem and $T = (L, \Gamma)$ be FUN-COMP. We must show $(\star)$ $T \models \mathbf{A}_o$.

$$\Gamma \models (f : A \to B)\!\downarrow \wedge\, (g : B \to C)\!\downarrow \wedge\, (h : C \to D)\!\downarrow \wedge\, (x : A)\!\downarrow \tag{1}$$
$$\Gamma \models ((f \circ g) \circ h)\, x \simeq h\,(g\,(f\,x)) \tag{2}$$
$$\Gamma \models (f \circ (g \circ h))\, x \simeq h\,(g\,(f\,x)) \tag{3}$$
$$\Gamma \models ((f \circ g) \circ h)\, x \simeq (f \circ (g \circ h))\, x \tag{4}$$
$$\Gamma \models \forall\, x : A \,.\, (f \circ (g \circ h))\, x \simeq ((f \circ g) \circ h)\, x \tag{5}$$
$$\Gamma \models f \circ (g \circ h) = (f \circ g) \circ h \tag{6}$$
$$\Gamma \models \mathbf{A}_o \tag{7}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (3) both follow from (1), the definition of $\circ$ in Table 9, function abstractions are always defined by [20, Axiom A5.11], ordered pairs of defined components are always defined by [20, Axiom A7.1], Beta-Reduction [20, Axiom A4], and Quasi-Equality Substitution [20, Lemma A.2]; (4) follows from (2) and (3) by the Quasi-Equality Rules [20, Lemma A.4]; (5) follows from (4) by Universal Generalization [20, Theorem A.30] using the fact that x is not free in Γ since Γ is a set of sentences; (6) follows from (5) by Extensionality [20, Axiom A3]; and (7) follows from (6) by Universal Generalization using the fact that f, g, and h are not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

2. Thm17: $\forall\, f : A \to B \,.\, \mathsf{id}_{A \to A} \circ f = f \circ \mathsf{id}_{B \to B} = f$ (identity functions are left and right identity elements).

Proof of the theorem. Let $\mathbf{A}_o$ be the theorem and $T = (L, \Gamma)$ be

FUN-COMP. We must show $(\star)$ $T \models \mathbf{A}_o$.

$$\Gamma \models (f : A \to B)\!\downarrow \wedge (x : A)\!\downarrow \tag{1}$$
$$\Gamma \models (\mathsf{id}_{A\to A} \circ f)\, x \simeq f\, x \tag{2}$$
$$\Gamma \models (f \circ \mathsf{id}_{B\to B})\, x \simeq f\, x \tag{3}$$
$$\Gamma \models \forall\, x : A\, .\, (\mathsf{id}_{A\to A} \circ f)\, x \simeq f\, x \tag{4}$$
$$\Gamma \models \forall\, x : A\, .\, (f \circ \mathsf{id}_{B\to B})\, x \simeq f\, x \tag{5}$$
$$\Gamma \models \mathsf{id}_{A\to A} \circ f = f \tag{6}$$
$$\Gamma \models f \circ \mathsf{id}_{B\to B} = f \tag{7}$$
$$\Gamma \models \mathsf{id}_{A\to A} \circ f = f \circ \mathsf{id}_{B\to B} = f \tag{8}$$
$$\Gamma \models \mathbf{A}_o \tag{9}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (3) both follow from (1), the definitions of id and $\circ$ in Table 9, function abstractions are always defined by [20, Axiom A5.11], ordered pairs of defined components are always defined by [20, Axiom A7.1], Beta-Reduction [20, Axiom A4], and Quasi-Equality Substitution [20, Lemma A.2]; (4) and (5) both follow from (2) and (3), respectively, by Universal Generalization [20, Theorem A.30] using the fact that x is not free in Γ since Γ is a set of sentences; (6) and (7) follow from (4) and (5), respectively, by Extensionality [20, Axiom A3]; (8) follows from (6) and (7) by the Equality Rules [20, Lemma A.13]; and (9) follows from (8) by Universal Generalization using the fact that f is not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

A.4 Development of ONE-BT

1. Thm18 (Thm16-via-FUN-COMP-to-ONE-BT):
   $\forall\, f, g, h : S \to S\, .\, f \circ (g \circ h) = (f \circ g) \circ h$ ($\circ$ is associative).

   Proof of the theorem. Similar to the proof of Thm11. □

2. Thm19 (Thm17-via-FUN-COMP-to-ONE-BT):
   $\forall\, f : S \to S\, .\, \mathsf{id}_{S\to S} \circ f = f \circ \mathsf{id}_{S\to S} = f$
   ($\mathsf{id}_{S\to S}$ is an identity element with respect to $\circ$).

   Proof of the theorem. Similar to the proof of Thm11. □

3. Def6 (Def1-via-MON-to-ONE-BT):
   $\mathsf{trans\text{-}monoid}_{\{S\to S\}\to o} =$
   $\lambda\, s : \{S \to S\}\, .$

$s \neq \emptyset_{\{S\to S\}} \wedge$
$\mathsf{TOTAL\text{-}ON}(\circ_{((S\to S)\times(S\to S))\to(S\to S)}|_{s\times s}, s\times s, s) \wedge$
$\mathsf{id}_{S\to S} \in s$ (transformation monoid).

Proof that RHS is defined. Let $\mathbf{A}^1_{\{M\}\to o}$ be the RHS of Def1, $\mathbf{A}^2_{\{S\to S\}\to o}$ be the RHS of Def6, T_1 be MON, and T_2 be ONE-BT, the top theory of ONE-BT-1. We must show $T_2 \vDash \mathbf{A}^2_{\{S\to S\}\to o}{\downarrow}$. We have previously proved ($\star$) $T_1 \vDash \mathbf{A}^1_{\{M\}\to o}{\downarrow}$. MON-to-ONE-BT $= (\mu, \nu)$ is a theory morphism from T_1 to T_2. Thus ($\star$) implies $T_2 \vDash \nu(\mathbf{A}^1_{\{M\}\to o}{\downarrow})$. Therefore, $T_2 \vDash \mathbf{A}^2_{\{S\to S\}\to o}{\downarrow}$ since $\mathbf{A}^2_{\{S\to S\}\to o} = \nu(\mathbf{A}^1_{\{M\}\to o})$. □

4. Thm20 (Thm4-via-MON-to-ONE-BT):
$\forall\, s : \{S \to S\}\,.$
$\mathsf{trans\text{-}monoid}\, s \Rightarrow \mathsf{MONOID}(s, \circ_{((S\to S)\times(S\to S))\to(S\to S)}|_{s\times s}, \mathsf{id}_{S\to S})$
(transformation monoids are monoids).

Proof of the theorem. Similar to the proof of Thm11. □

A.5 Development of MON-ACT

1. Thm21: $\mathsf{MON\text{-}ACTION}(U_{\{M\}}, U_{\{S\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \mathsf{act}_{(M\times S)\to S})$
(models of MON-ACT define monoid actions).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON-ACT. We must show

($\star$) $T \vDash \mathsf{MON\text{-}ACTION}(U_{\{M\}}, U_{\{S\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \mathsf{act}_{(M\times S)\to S})$.

$$\Gamma \vDash \mathsf{MONOID}(U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M) \tag{1}$$
$$\Gamma \vDash U_{\{S\}}{\downarrow} \tag{2}$$
$$\Gamma \vDash U_{\{S\}} \neq \emptyset_{\{S\}} \tag{3}$$
$$\Gamma \vDash \mathsf{act}_{(M\times S)\to S} \downarrow (U_{\{M\}} \times U_{\{S\}}) \to U_{\{S\}} \tag{4}$$
$$\Gamma \vDash \forall\, x, y : U_{\{M\}},\ s : U_{\{S\}}\ .\ x\ \mathsf{act}\ (y\ \mathsf{act}\ s) = (x \cdot y)\ \mathsf{act}\ s \tag{5}$$
$$\Gamma \vDash \forall\, s : U_{\{S\}}\ .\ \mathsf{e}\ \mathsf{act}\ s = s \tag{6}$$
$$\Gamma \vDash \mathsf{MON\text{-}ACTION}(U_{\{M\}}, U_{\{S\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \mathsf{act}_{(M\times S)\to S}) \tag{7}$$

(1) follows from MON $\vDash$ Thm1 and MON $\leq T$; (2) and (3) follow from parts 1 and 2, respectively, of Lemma B.1; (4) follows from [20, Axiom 5.2] and parts 8–10 of Lemma B.1; (5) and (6) follow from Axioms 3 and 4, respectively, of T and part 5 of Lemma B.1; and (7) follows from (1)–(6) and the definition of MON-ACTION in Table 10. Therefore, ($\star$) holds. □

Thm22: $\mathsf{TOTAL}(\mathsf{act}_{(M\times S)\to S})$ (act is total).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON-ACT.

$$T \vDash \mathsf{TOTAL}(\mathsf{act}_{(M\times S)\to S})$$

follows from Axiom 3 of T in the same way that $T \vDash \mathsf{TOTAL}(\cdot_{(M\times M)\to M})$ follows from Axiom 1 of MON as shown in the proof of Thm2. □

2. Def7: $\mathsf{orbit}_{S\to\{S\}} = \lambda\, s : S\,.\,\{t : S \mid \exists\, x : M\,.\, x\ \mathsf{act}\ s = t\}$ (orbit).

   Proof that RHS is defined. Similar to the proof that the RHS of Def1 is defined. □

3. Def8: $\mathsf{stabilizer}_{S\to\{M\}} = \lambda\, s : S\,.\,\{x : M \mid x\ \mathsf{act}\ s = s\}$ (stabilizer).

   Proof that RHS is defined. Similar to the proof that the RHS of Def1 is defined. □

4. Thm23: $\forall\, s : S\,.\,\mathsf{submonoid}\,(\mathsf{stabilizer}\, s)$ (stabilizers are submonoids).

   Proof of the theorem. Let $T = (L, \Gamma)$ be MON-ACT extended by Def7 and Def8. We must show

   $$(\star)\ T \vDash \forall\, s : S\,.\,\mathsf{submonoid}\,(\mathsf{stabilizer}\, s).$$

   $$\Gamma \vDash (s : S)\!\downarrow \tag{1}$$
   $$\Gamma \vDash \mathsf{e}_M\!\downarrow \tag{2}$$
   $$\Gamma \vDash (\mathsf{stabilizer}\, s) = \{x : M \mid x\ \mathsf{act}\ s = s\} \tag{3}$$
   $$\Gamma \vDash \mathsf{e} \in (\mathsf{stabilizer}\, s) \tag{4}$$
   $$\Gamma \vDash (\mathsf{stabilizer}\, s) \neq \emptyset_{\{M\}} \tag{5}$$
   $$\Gamma \vDash \cdot|_{(\mathsf{stabilizer}\, s)\times(\mathsf{stabilizer}\, s)}\downarrow$$
   $$((\mathsf{stabilizer}\, s) \times (\mathsf{stabilizer}\, s)) \to (\mathsf{stabilizer}\, s) \tag{6}$$
   $$\Gamma \vDash (\mathsf{stabilizer}\, s)\!\downarrow \tag{7}$$
   $$\Gamma \vDash \mathsf{submonoid}\,(\mathsf{stabilizer}\, s) =$$
   $$(\mathsf{stabilizer}\, s) \neq \emptyset_{\{M\}} \wedge$$
   $$\cdot|_{(\mathsf{stabilizer}\, s)\times(\mathsf{stabilizer}\, s)}\downarrow$$
   $$((\mathsf{stabilizer}\, s) \times (\mathsf{stabilizer}\, s)) \to (\mathsf{stabilizer}\, s) \wedge$$
   $$\mathsf{e} \in (\mathsf{stabilizer}\, s) \tag{8}$$
   $$\Gamma \vDash \mathsf{submonoid}\,(\mathsf{stabilizer}\, s) \tag{9}$$
   $$\Gamma \vDash \forall\, s : S\,.\,\mathsf{submonoid}\,(\mathsf{stabilizer}\, s) \tag{10}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from constants always being defined by [20, Axiom A5.2]; (3) follows from Def8 by the Equality Rules [20, Lemma A.13] and Beta-Reduction [20, Axiom A4] applied to (1) and the RHS of the result; (4) follows from (3) and Axiom 4 of T; (5) follows immediately from (4); (6) follows from Thm2, (3), and Axiom 3 of T; (7) follows from (3) and [20, Axiom A5.4]; (8) follows from Def1 by the Equality Rules and Beta-Reduction applied to (7) and the RHS of the result; (9) follows from (4), (5), (6), and (8) by the Tautology Rule [20, Corollary A.46]; (10) follows from (9) by Universal Generalization [20, Theorem A.30] using the fact that s is free in Γ because Γ is a set of sentences. Therefore, ($\star$) holds. □

5. Thm24 (Thm21-via-MON-ACT-to-MON):
   MON-ACTION($U_{\{M\}}, U_{\{M\}}, \cdot_{(M\times M)\to M}, \mathsf{e}_M, \cdot_{(M\times M)\to M}$)
   (first example is a monoid action).

   Proof of the theorem. Similar to the proof of Thm11. □

A.6 Development of ONE-BT-with-SC

1. Thm25 (Thm21-via-MON-ACT-to-ONE-BT-with-SC):
   MON-ACTION($\mathsf{F}_{\{S\to S\}}$,
   $U_{\{S\}}$,
   $\circ_{((S\to S)\times(S\to S))\to(S\to S)|\mathsf{F}\times\mathsf{F}}$,
   $\mathsf{id}_{S\to S}$,
   $\bullet_{((S\to S)\times S)\to S|\mathsf{F}\times S}$)
   (second example is a monoid action).

   Proof of the theorem. Similar to the proof of Thm11. □

A.7 Development of MON-HOM

1. Thm26:
   MON-HOM($U_{\{M_1\}}$,
   $U_{\{M_2\}}$,
   $\cdot_{(M_1\times M_1)\to M_1}$,
   e_{M_1},
   $\cdot_{(M_2\times M_2)\to M_2}$,
   e_{M_2},
   $\mathsf{h}_{M_1\to M_2}$)
   (models of MON-HOM define monoid homomorphisms).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON-HOM and $\mathbf{A}_o$ be

$$\mathsf{MON\text{-}HOM}(U_{\{M_1\}}, U_{\{M_2\}}, \cdot_{(M_1 \times M_1) \to M_1}, \mathsf{e}_{M_1}, \cdot_{(M_2 \times M_2) \to M_2}, \mathsf{e}_{M_2}, \mathsf{h}_{M_1 \to M_2}).$$

We must show $(\star)$ $T \models \mathbf{A}_o$.

$$\Gamma \models \mathsf{MONOID}(U_{\{M_1\}}, \cdot_{(M_1 \times M_1) \to M_1}, \mathsf{e}_{M_1}) \tag{1}$$
$$\Gamma \models \mathsf{MONOID}(U_{\{M_2\}}, \cdot_{(M_2 \times M_2) \to M_2}, \mathsf{e}_{M_2}) \tag{2}$$
$$\Gamma \models \mathsf{h}_{M_1 \to M_2} \downarrow U_{\{M_1\}} \to U_{\{M_2\}} \tag{3}$$
$$\Gamma \models \forall\, x, y : U_{\{M_1\}} \,.\, \mathsf{h}\,(x \cdot y) = (\mathsf{h}\,x) \cdot (\mathsf{h}\,y) \tag{4}$$
$$\Gamma \models \mathbf{A}_o \tag{5}$$

(1) and (2) follow similarly to the proof of Thm1; (3) follows from [20, Axiom 5.2] and parts 8 and 9 of Lemma B.1; (4) follows from Axiom 5 of T and part 5 of Lemma B.1; (5) follows from (1)–(4), Axiom 6 of T, and the definition of MON-HOM in Table 10. Therefore, $(\star)$ holds. □

Thm27: $\mathsf{TOTAL}(\mathsf{h}_{M_1 \to M_2})$ ($\mathsf{h}_{M_1 \to M_2}$ is total).

Proof of the theorem. Let $T = (L, \Gamma)$ be MON-HOM.

$$T \models \mathsf{TOTAL}(\mathsf{h}_{M_1 \to M_2})$$

follows from Axiom 5 of T in the same way that $T \models \mathsf{TOTAL}(\cdot_{(M \times M) \to M})$ follows from Axiom 1 of MON as shown in the proof of Thm2. □

2. Thm28 (Thm26-via-MON-HOM-to-MON-4)
   $\mathsf{MON\text{-}HOM}(U_{\{M\}}, U_{\{\{M\}\}}, \cdot_{(M \times M) \to M}, \mathsf{e}_M, \cdot_{(\{M\} \times \{M\}) \to \{M\}}, \mathsf{E}_{\{M\}}, \mathsf{h}_{M \to \{M\}})$ (example is a monoid homomorphism).

   Proof of the theorem. Similar to the proof of Thm11. □

A.8 Development of MON-over-COF

1. Def9: $\mathsf{prod}_{R \to R \to (R \to M) \to M} =$
   $\mathrm{I}\, f : Z_{\{R\}} \to Z_{\{R\}} \to (Z_{\{R\}} \to M) \to M$.

$$\forall\, m, n : Z_{\{R\}},\, g : Z_{\{R\}} \to M \,.\, f\, m\, n\, g \simeq (m > n \mapsto \mathsf{e} \mid (f\, m\, (n-1)\, g) \cdot (g\, n)) \qquad \text{(iterated product)}.$$

Proof that RHS is defined. Let

$$\mathbf{A}_o = \forall\, m, n : Z_{\{R\}},\, g : Z_{\{R\}} \to M \,.\, f\, m\, n\, g \simeq (m > n \mapsto \mathsf{e} \mid (f\, m\, (n-1)\, g) \cdot (g\, n)).$$

Suppose that two functions f_1 and f_2 satisfy $\mathbf{A}_o$. It is easy to see that f_1 and f_2 must be the same function based on the recursive structure of f in $\mathbf{A}_o$. Thus, $\mathbf{A}_o$ specifies a unique function, and so the RHS of Def9 is defined by [20, Axiom A6.1]. □

2. Thm29: $\forall\, m : Z_{\{R\}},\, g : Z_{\{R\}} \to M \,.\, \Big(\prod_{i=m}^{m} g\, i\Big) \simeq g\, m$

(trivial product).

Proof of the theorem. Let $\mathbf{A}_o$ be the theorem and $T = (L, \Gamma)$ be COM-MON-over-COF extended with Def9. We must show $(\star)$ $T \vDash \mathbf{A}_o$.

Let Δ be the set $\{m \in Z_{\{R\}}, g \in Z_{\{R\}} \to M\}$.

$$\Gamma \cup \Delta \vDash (m : R)\!\downarrow \wedge\, (g : R \to M)\!\downarrow \tag{1}$$

$$\Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{m} g\, i\Big) \simeq \Big(\prod_{i=m}^{m-1} g\, i\Big) \cdot g\, m \tag{2}$$

$$\Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{m-1} g\, i\Big) \cdot g\, m \simeq \mathsf{e} \cdot g\, m \tag{3}$$

$$\Gamma \cup \Delta \vDash \mathsf{e} \cdot g\, m \simeq g\, m \tag{4}$$

$$\Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{m} g\, i\Big) \simeq g\, m \tag{5}$$

$$\Gamma \vDash \mathbf{A}_o \tag{6}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (3) follow from (1) and Def9; (4) follows from Axiom 20 of T; (5) follows from (2), (3), and (4) by the Quasi-Equality Rules [20, Lemma A.4]; and (6) follows from (5) by the Deduction Theorem [20, Theorem A.50] and by Universal Generalization [20, Theorem A.30] using the fact that m and g are not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

3. Thm30: $\forall\, m, k, n : Z_{\{R\}},\ g : Z_{\{R\}} \to M\ .$
$$m < k < n \Rightarrow \Big(\prod_{i=m}^{k} g\, i\Big) \cdot \Big(\prod_{i=k+1}^{n} g\, i\Big) \simeq \prod_{i=m}^{n} g\, i$$
(extended iterated product).

Proof of the theorem. Let $\mathbf{A}_o$ be the theorem and $T = (L, \Gamma)$ be MON-over-COF extended by Def9. We must show (a) $T \vDash \mathbf{A}_o$.

Let Δ be the set

$$\{m \in Z_{\{R\}}, k \in Z_{\{R\}}, n \in Z_{\{R\}}, m < k < n\}.$$

We will prove

$$\text{(b) } \Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{k} g\, i\Big) \cdot \Big(\prod_{i=k+1}^{n} g\, i\Big) \simeq \Big(\prod_{i=m}^{n} g\, i\Big)$$

from all $n > k$ by induction on the n.

Base case: $n = k + 1$. Then:

$$\Gamma \cup \Delta \vDash (m : R)\!\downarrow \wedge (k : R)\!\downarrow \wedge (n : R)\!\downarrow \wedge (g : R \to M)\!\downarrow \tag{1}$$

$$\Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{k} g\, i\Big) \cdot \Big(\prod_{i=k+1}^{n} g\, i\Big) \simeq \Big(\prod_{i=m}^{k} g\, i\Big) \cdot g\, n \tag{2}$$

$$\Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{k} g\, i\Big) \cdot g\, n \simeq \Big(\prod_{i=m}^{n} g\, i\Big) \tag{3}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) follows from $n = k + 1$ and Thm29; and (3) follows from $n = k + 1$, (1), and Def9. Thus (b) holds by the Quasi-Equality Rules [20, Lemma A.4] when $n = k + 1$.

Induction step: $n > k + 1$ *and assume*

$$\Gamma \cup \Delta \vDash \Big(\prod_{i=m}^{k} g\, i\Big) \cdot \Big(\prod_{i=k+1}^{n-1} g\, i\Big) \simeq \Big(\prod_{i=m}^{n-1} g\, i\Big).$$

Then:

$$\Gamma \cup \Delta \models (m : R)\!\downarrow \wedge (k : R)\!\downarrow \wedge (n : R)\!\downarrow \wedge (g : R \to M)\!\downarrow \tag{1}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{k} g\,i\Big) \cdot \Big(\prod_{i=k+1}^{n} g\,i\Big) \simeq \Big(\prod_{i=m}^{k} g\,i\Big) \cdot \Big(\Big(\prod_{i=k+1}^{n-1} g\,i\Big) \cdot g\,n\Big) \tag{2}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{k} g\,i\Big) \cdot \Big(\Big(\prod_{i=k+1}^{n-1} g\,i\Big) \cdot g\,n\Big) \simeq \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot g\,n \tag{3}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot g\,n \simeq \Big(\prod_{i=m}^{n} g\,i\Big) \tag{4}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (4) follows from (1) and Def9; and (3) follows from Axiom 19 of T and the induction hypothesis. Thus (b) holds by the Quasi-Equality Rules [20, Lemma A.4] when $n > k + 1$.

Therefore, (b) holds for all $n > k$, and (a) follows from this by the Deduction Theorem [20, Theorem A.50] and by Universal Generalization [20, Theorem A.30] using the fact that m, k, n, and g are not free in Γ since Γ is a set of sentences. □

A.9 Development of COM-MON-over-COF

1. Thm31: $\forall\, m, n : Z_{\{R\}},\, g, h : Z_{\{R\}} \to M\,.$
$\Big(\prod_{i=m}^{n} g\,i\Big) \cdot \Big(\prod_{i=m}^{n} h\,i\Big) \simeq \prod_{i=m}^{n} (g\,i) \cdot (h\,i)$

(product of iterated products).

Proof of the theorem. Let $\mathbf{A}_o$ be the theorem and $T = (L, \Gamma)$ be COM-MON-over-COF extended by Def9. We must show (a) $T \models \mathbf{A}_o$.

Let Δ be the set $\{n \in Z_{\{R\}}, g \in Z_{\{R\}} \to M\}$. We will prove

$$\text{(b) } \Gamma \cup \Delta \models \Big(\prod_{i=m}^{n} g\,i\Big) \cdot \Big(\prod_{i=m}^{n} h\,i\Big) \simeq \prod_{i=m}^{n} (g\,i) \cdot (h\,i)$$

for all n by induction on the n.

Base case: $n < m$. Then:

$$\Gamma \cup \Delta \models (n : R)\!\downarrow \wedge (g : R \to M)\!\downarrow \tag{1}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n} g\,i\Big) \cdot \Big(\prod_{i=m}^{n} h\,i\Big) \simeq \mathsf{e} \cdot \mathsf{e} \tag{2}$$

$$\Gamma \cup \Delta \models \prod_{i=m}^{n} (g\,i) \cdot (h\,i) \simeq \mathsf{e} \tag{3}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; and (2) and (3) follow from $n < m$, (1), and Def9. Thus (b) holds by Axiom 20 of T and the Quasi-Equality Rules [20, Lemma A.4] when $n < m$.

Induction step: $n \geq m$ *and assume*

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot \Big(\prod_{i=m}^{n-1} h\,i\Big) \simeq \prod_{i=m}^{n-1} (g\,i) \cdot (h\,i).$$

Then:

$$\Gamma \cup \Delta \models (n : R)\!\downarrow \wedge (g : R \to M)\!\downarrow \tag{1}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n} g\,i\Big) \cdot \Big(\prod_{i=m}^{n} h\,i\Big) \simeq \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot g\,n \cdot \Big(\prod_{i=m}^{n-1} h\,i\Big) \cdot h\,n \tag{2}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot g\,n \cdot \Big(\prod_{i=m}^{n-1} h\,i\Big) \cdot h\,n \simeq \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot \Big(\prod_{i=m}^{n-1} h\,i\Big) \cdot g\,n \cdot h\,n \tag{3}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n-1} g\,i\Big) \cdot \Big(\prod_{i=m}^{n-1} h\,i\Big) \cdot g\,n \cdot h\,n \simeq \Big(\prod_{i=m}^{n-1} (g\,i) \cdot (h\,i)\Big) \cdot (g\,n \cdot h\,n) \tag{4}$$

$$\Gamma \cup \Delta \models \Big(\prod_{i=m}^{n-1} (g\,i) \cdot (h\,i)\Big) \cdot (g\,n \cdot h\,n) \simeq \prod_{i=m}^{n} (g\,i) \cdot (h\,i) \tag{5}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (5) follow from (1) and Def9; (3) follows from Axiom 21 of T; and (4) follows from the induction hypothesis. Thus (b) holds by the Quasi-Equality Rules [20, Lemma A.4] when $n \geq m$.

Therefore, (b) holds for all n, and (a) follows from this by the Deduction Theorem [20, Theorem A.50] and by Universal Generalization [20, Theorem A.30] using the fact that n and g are not free in Γ since Γ is a set of sentences. □

A.10 Development of COM-MON-ACT-over-COF

1. Thm32: $\forall\, x, y : M,\ s : S\ .\ x \mathsf{\ act\ } (y \mathsf{\ act\ } s) = y \mathsf{\ act\ } (x \mathsf{\ act\ } s)$
(act has commutative-like property).

Proof of the theorem. Let $\mathbf{A}_o$ be the theorem and $T = (L, \Gamma)$ be COM-MON-ACT-over- COF. We must show $(\star)$ $T \vDash \mathbf{A}_o$.

$$\Gamma \vDash (x : M)\!\downarrow \wedge\, (y : M)\!\downarrow \wedge\, (s : S)\!\downarrow \tag{1}$$
$$\Gamma \vDash x \mathsf{\ act\ } (y \mathsf{\ act\ } s) = (x \cdot y) \mathsf{\ act\ } s \tag{2}$$
$$\Gamma \vDash y \mathsf{\ act\ } (x \mathsf{\ act\ } s) = (y \cdot x) \mathsf{\ act\ } s \tag{3}$$
$$\Gamma \vDash x \cdot y = y \cdot x \tag{4}$$
$$\Gamma \vDash y \mathsf{\ act\ } (x \mathsf{\ act\ } s) = (x \cdot y) \mathsf{\ act\ } s \tag{5}$$
$$\Gamma \vDash x \mathsf{\ act\ } (y \mathsf{\ act\ } s) = y \mathsf{\ act\ } (x \mathsf{\ act\ } s) \tag{6}$$
$$\Gamma \vDash \mathbf{A}_o \tag{7}$$

(1) follows from variables always being defined by [20, Axiom A5.1]; (2) and (3) follow from (1) and Axiom 22 of T by Universal Instantiation [20, Theorem A.14]; (4) follows from (1) and Axiom 21 of T by Universal Instantiation; (5) follows from (4) and (3) by Quasi-Equality Substitution [20, Lemma A.2]; (6) follows from (2) and (5) by the Equality Rules [20, Lemma A.13]; (7) follows from (6) by Universal Generalization [20, Theorem A.30] using the fact that x, y, and s are not free in Γ since Γ is a set of sentences. Therefore, $(\star)$ holds. □

A.11 Development of STR

1. Def10: $\mathsf{str}_{\{R \to A\}} = [A]$ (string quasitype).

Proof that RHS is defined. Let T be the top theory of STR-1. We must show $(\star)$ $T \vDash [A]\!\downarrow$. Now $[A]$ stands for

$$\{s : \langle\!\langle A \rangle\!\rangle \mid \exists\, n : \mathbf{C}^N_{\{R\}}\ .\ \forall\, m : \mathbf{C}^N_{\{R\}}\ .\ (s\ m)\!\downarrow \Leftrightarrow \mathbf{C}_{A \to A \to o} m\ n\}$$

based on the notational definitions in Table 12. Thus $(\star)$ holds because function abstractions are always defined by [20, Axiom A5.11]. □

2. Def11: $\epsilon_{R\to A} = [\,]_{R\to A}$ (empty string).

   Proof that RHS is defined. Let T be the top theory of STR-1. We must show ($\star$) $T \vDash [\,]_{R\to A}\!\downarrow$. Now $[\,]_{R\to A}$ stands for

   $$\lambda\, x : R \,.\, \bot_A$$

   based on the notational definitions in Tables 4 and 12. Thus ($\star$) holds because function abstractions are always defined by [20, Axiom A5.11]. □

   Def12: $\mathsf{cat}_{((R\to A)\times(R\to A))\to(R\to A)} = {+\!+}_{(R\to A)\to(R\to A)\to(R\to A)}$ (concatenation).

   Proof that RHS is defined. Let T be the top theory of STR-1. We must show

   $$(\star)\ T \vDash {+\!+}_{(R\to A)\to(R\to A)\to(R\to A)}\!\downarrow.$$

   The pseudoconstant ${+\!+}_{(\alpha\to\beta)\to(\alpha\to\beta)\to(\alpha\to\beta)}$ is defined in Table 12. For all α and β, ${+\!+}_{(\alpha\to\beta)\to(\alpha\to\beta)\to(\alpha\to\beta)}$ denotes the concatenation function for finite sequences over the denotation of β. Therefore, ($\star$) holds. □

3. Thm33: $\forall\, x : \mathsf{str} \,.\, \epsilon x = x\epsilon = x$ (ϵ is an identity element).

   Proof of the theorem. Let $T = (L, \Gamma)$ be the top theory of STR-1 extended by Def10–Def12. We must show:

   (a) $T \vDash \forall\, x : \mathsf{str} \,.\, \epsilon x = x$.

   (b) $T \vDash \forall\, x : \mathsf{str} \,.\, x\epsilon = x$.

   Let Δ be the set $\{x \in \mathsf{str}\}$. Then:

   $$\Gamma \cup \Delta \vDash \epsilon x = x \tag{1}$$
   $$\Gamma \vDash \forall\, x : \mathsf{str} \,.\, \epsilon x = x \tag{2}$$

   (1) follows from $x \in \mathsf{str}$ and Def12; and (2) follows from (1) by the Deduction Theorem [20, Theorem A.50] and then by Universal Generalization [20, Theorem A.30] using the fact that $(x : R \to A)$ is not free in Γ since Γ is a set of sentences. Therefore, (a) holds.

   We will prove (c) $\Gamma \cup \Delta \vDash x\epsilon = x$ by induction on the length of x.

   Base case: x is ϵ. Then $\Gamma \cup \Delta \vDash \epsilon\epsilon = \epsilon$ is an instance of (1) above.

Induction step: x *is* $(a :: y)$ *and assume* $\Gamma \cup \Delta \models y\epsilon = y$. Then:

$$\Gamma \cup \Delta \models (a :: y)\epsilon = (a :: y\epsilon) \tag{1}$$
$$\Gamma \cup \Delta \models (a :: y\epsilon) = (a :: y) \tag{2}$$

(1) follows from $x \in \mathsf{str}$ and Def12; and (2) follows from the induction hypothesis and (1) by Quasi-Equality Substitution [20, Lemma A.2]. Thus $\Gamma \cup \Delta \models (a :: y)\epsilon = (a :: y)$ holds by the Equality Rules [20, Lemma A.13].

Therefore (c) holds, and (b) follows from (c) by the Deduction Theorem [20, Theorem A.50] and then by Universal Generalization [20, Theorem A.30] using the fact that $(x : R \to A)$ is not free in Γ since Γ is a set of sentences. □

4. Thm34: $\forall\, x, y, z : \mathsf{str}\ .\ x(yz) = (xy)z$ (cat is associative).

Proof of the theorem. Let $T = (L, \Gamma)$ be the top theory of STR-1 extended by Def10–Def12. We must show

(a) $T \models \forall\, x, y, z : \mathsf{str}\ .\ x(yz) = (xy)z$.

Let Δ be the set $\{x \in \mathsf{str}, y \in \mathsf{str}, z \in \mathsf{str}\}$. We will prove

(b) $\Gamma \cup \Delta \models x(yz) = (xy)z$

by induction on the length of x.

Base case: x *is* ϵ. Then:

$$\Gamma \cup \Delta \models \epsilon(yz) = (yz) \tag{1}$$
$$\Gamma \cup \Delta \models (yz) = (\epsilon y)z \tag{2}$$

(1) and (2) follow from Thm33. Thus $\Gamma \cup \Delta \models \epsilon(yz) = (\epsilon y)z$ holds by the Equality Rules [20, Lemma A.13].

Induction step: x *is* $(a :: w)$ *and assume* $\Gamma \cup \Delta \models w(yz) = (wy)z$. Then:

$$\Gamma \cup \Delta \models (a :: w)(yz) = a :: w(yz) \tag{1}$$
$$\Gamma \cup \Delta \models a :: w(yz) = a :: (wy)z \tag{2}$$
$$\Gamma \cup \Delta \models a :: (wy)z = (a :: wy)z \tag{3}$$
$$\Gamma \cup \Delta \models (a :: wy)z = ((a :: w)y)z \tag{4}$$

(1), (3), and (4) follow from $x \in \mathsf{str}$, $y \in \mathsf{str}$, and $z \in \mathsf{str}$ and Def12; and (2) follows from the induction hypothesis. Thus

$$\Gamma \cup \Delta \models (a :: w)(yz) = ((a :: w)y)z$$

holds by the Equality Rules [20, Lemma A.13].

Therefore (b) holds, and (a) follows from (b) by the Deduction Theorem [20, Theorem A.50] and then by Universal Generalization [20, Theorem A.30] using the fact that $(x : R \to A)$, $(y : R \to A)$, and $(z : R \to A)$ are not free in Γ since Γ is a set of sentences. □

5. Thm35 (Thm1-via-MON-over-COF-to-STR-2):
   $\mathsf{MONOID}(\mathsf{str}_{\{R\to A\}}, \mathsf{cat}_{((R\to A)\times(R\to A))\to(R\to A)}, \epsilon_{R\to A})$
   (strings form a monoid).

   Proof of the theorem. Similar to the proof of Thm11. □

6. Def13 (Def3-via-MON-over-COF-to-STR-2):
   $\mathsf{set\text{-}cat}_{(\{R\to A\}\times\{R\to A\})\to\{R\to A\}} =$
   $\mathsf{set\text{-}op}_{(((R\to A)\times(R\to A))\to(R\to A))\to((\{R\to A\}\times\{R\to A\})\to\{R\to A\})}\,\mathsf{cat}$
   (set concatenation).

   Proof that RHS is defined. Similar to the proof that the RHS of Def6 is defined. □

7. Def14 (Def4-via-MON-over-COF-to-STR-2):
   $\mathsf{E}_{\{R\to A\}} = \{\epsilon_{R\to A}\}$ (set identity element).

   Proof that RHS is defined. Similar to the proof that the RHS of Def6 is defined. □

8. Thm36 (Thm12-via-MON-over-COF-1-to-STR-2):
   $\mathsf{MONOID}(\mathcal{P}(\mathsf{str}_{\{R\to A\}}), \mathsf{set\text{-}cat}_{(\{R\to A\}\times\{R\to A\})\to\{R\to A\}}, \mathsf{E}_{\{R\to A\}})$
   (string sets form a monoid).

   Proof of the theorem. Similar to the proof of Thm11. □

9. Def15 (Def9-via-MON-over-COF-1-to-STR-2):
   $\mathsf{iter\text{-}cat}_{R\to R\to(R\to(R\to A))\to(R\to A)} =$
   $\mathrm{I}\, f : Z_{\{R\}} \to Z_{\{R\}} \to (Z_{\{R\}} \to (R \to A)) \to (R \to A)\ .$
   $\forall\, m, n : Z_{\{R\}},\ g : Z_{\{R\}} \to (R \to A)\ .\ f\, m\, n\, g \simeq$
   $(m > n \mapsto \epsilon \mid (f\, m\, (n-1)\, g)\ \mathsf{cat}\ (g\, n))$
   (iterated concatenation).

   Proof that RHS is defined. Similar to the proof that the RHS of Def6 is defined. □

B Miscellaneous theorems

Lemma B.1 (Universal Sets)**.** *The following formulas are valid:*

1. $U_{\{\alpha\}}{\downarrow}$.
2. $U_{\{\alpha\}} \neq \emptyset_{\{\alpha\}}$.
3. $\forall\, x : \alpha \;.\; x \in U_{\{\alpha\}}$.
4. $(\lambda\, \mathbf{x} : \alpha \;.\; \mathbf{B}_\beta) = (\lambda\, \mathbf{x} : U_{\{\alpha\}} \;.\; \mathbf{B}_\beta)$.
5. $(\forall\, \mathbf{x} : \alpha \;.\; \mathbf{B}_o) \Leftrightarrow (\forall\, \mathbf{x} : U_{\{\alpha\}} \;.\; \mathbf{B}_o)$.
6. $(\exists\, \mathbf{x} : \alpha \;.\; \mathbf{B}_o) \Leftrightarrow (\exists\, \mathbf{x} : U_{\{\alpha\}} \;.\; \mathbf{B}_o)$.
7. $(\mathrm{I}\, \mathbf{x} : \alpha \;.\; \mathbf{B}_o) \simeq (\mathrm{I}\, \mathbf{x} : U_{\{\alpha\}} \;.\; \mathbf{B}_o)$.
8. $\mathbf{A}_\alpha{\downarrow} \Leftrightarrow (\mathbf{A}_\alpha \downarrow U_{\{\alpha\}})$
9. $U_{\{\alpha\to\beta\}} = (U_{\{\alpha\}} \to U_{\{\beta\}})$.
10. $U_{\{\alpha\times\beta\}} = (U_{\{\alpha\}} \times U_{\{\beta\}})$.
11. $U_{\{\{\alpha\}\}} = \mathcal{P}(U_{\{\alpha\}})$.
12. $\boldsymbol{A}_{(\alpha\times\beta)\to\gamma} = \boldsymbol{A}_{(\alpha\times\beta)\to\gamma}|_{U_{\{\alpha\}}\times U_{\{\beta\}}}$.

Proof The proof is left to the reader as an exercise. □

Lemma B.2. *Let T be* MON *extended by the definition* Def2*. The formula*

$$\boldsymbol{A}_M \cdot^{\mathrm{op}} \boldsymbol{B}_M \simeq \boldsymbol{B}_M \cdot \boldsymbol{A}_M$$

is valid in T.

Proof Let $\mathbf{X}_o$ be

$$\mathbf{A}_M \cdot^{\mathrm{op}} \mathbf{B}_M \simeq \mathbf{B}_M \cdot \mathbf{A}_M,$$

N be a model of T, and $\varphi \in \mathsf{assign}(N)$. Suppose that $V_\varphi^N(\mathbf{A}_M)$ or $V_\varphi^N(\mathbf{B}_M)$ is undefined. Then clearly $V_\varphi^N(\mathbf{X}_o) = \mathrm{T}$. Now suppose that $V_\varphi^N(\mathbf{A}_M)$ and $V_\varphi^N(\mathbf{B}_M)$ are defined. Then $V_\varphi^N(\mathbf{X}_o) = \mathrm{T}$ by Def2. □

References

[1] P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof*. Springer, second edition, 2002.

[2] P. B. Andrews, M. Bishop, S. Issar, D. Nesmith, F. Pfennig, and H. Xi. TPS: A theorem-proving system for classical type theory. *Journal of Automated Reasoning*, 16:321–353, 1996.

[3] E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P. D. Mosses, D. Sannella, and A. Tarlecki. CASL: the Common Algebraic Specification Language. *Theoretical Computer Science*, 286:153–196, 2002.

[4] S. Autexier, D. Hutter, H. Mantel, and A. Schairer. Towards an evolutionary formal software-development using CASL. In D. Bert, C. Choppy, and P. D. Mosses, editors, *Recent Trends in Algebraic Development Techniques (WADT 1999)*, volume 1827 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 1999.

[5] C. Ballarin. Locales: A module system for mathematical theories. *Journal of Automated Reasoning*, 52:123–153, 2014.

[6] A. Bordg, L. Paulson, and W. Li. Simple type theory is not too simple: Grothendieck's schemes without dependent types. *Experimental Mathematics*, 31:364–382, 2022.

[7] A. Bove, P. Dybjer, and U. Norell. A brief overview of Agda — A functional language with dependent types. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2009)*, volume 5674 of *Lecture Notes in Computer Science*, pages 73–78. Springer, 2009.

[8] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[9] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, 1986.

[10] L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean theorem prover (system description). In A. P. Felty and A. Middeldorp, editors, *Automated Deduction — CADE-25*, volume 9195, pages 378–388, 2015.

[11] Epigram: Marking Dependent Types matter. http://www.e-pig.org/. Accessed on 3 January 2025.

[12] F*. https://www.fstar-lang.org/. Accessed on 3 January 2025.

[13] W. M. Farmer. A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–91, 1990.

[14] W. M. Farmer. A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211–240, 1993.

[15] W. M. Farmer. Theory interpretation in simple type theory. In J. Heering, K. Meinke, B. Möller, and T. Nipkow, editors, *Higher-Order Algebra, Logic, and Term Rewriting (HOA 1993)*, volume 816 of *Lecture Notes in Computer Science*, pages 96–123. Springer, 1994.

[16] W. M. Farmer. Formalizing undefinedness arising in calculus. In D. A. Basin and M. Rusinowitch, editors, *Automated Reasoning — IJCAR 2004*, volume 3097 of *Lecture Notes in Computer Science*, pages 475–489. Springer, 2004.

[17] W. M. Farmer. Andrews' type system with undefinedness. In C. Benzmüller, C. E. Brown, J. Siekmann, and R. Statman, editors, *Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on his 70th Birthday*, volume 17 of *Studies in Logic*, pages 223–242. College Publications, 2008.

[18] W. M. Farmer. The seven virtues of simple type theory. *Journal of Applied Logic*, 6:267–286, 2008.

[19] W. M. Farmer. LaTeX for Alonzo. https://imps.mcmaster.ca/doc/latex-for-alonzo.pdf, 2023 (revised 2024).

[20] W. M. Farmer. *Simple Type Theory: A Practical Logic for Expressing and Reasoning About Mathematical Ideas*. Computer Science Foundations and Applied Logic. Birkhäuser/Springer, second edition, 348 pp., 2025.

[21] W. M. Farmer. Formal mathematics for the masses. In J. Blanchette, Davenport J, P. Koepke, M. Kohlhase, A. Kohlhase, A. Naumowicz, D. Müller, Y. Sharoda, and C. Sacerdoti Coen, editors, *Workshop Papers of the 14th Conference on Intelligent Computer Mathematics (CICM 2021)*, volume 3377 of *CEUR Workshop Proceedings*. CEUR-WS.org, 8 pp., 2023.

[22] W. M. Farmer, J. D. Guttman, and F. J. Thayer. Little theories. In D. Kapur, editor, *Automated Deduction — CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 567–581. Springer, 1992.

[23] W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.

[24] W. M. Farmer, J. D. Guttman, and F. J. Thayer. *The IMPS 2.0 User's Manual*. The MITRE Corporation, 1998. Available at https://imps.mcmaster.ca/doc/imps-2.0-manual.pdf.

[25] Formal Abstracts. https://formalabstracts.github.io, 2022. Accessed on 3 January 2025.

[26] F. Garillot. *Generic Proof Tools and Finite Group Theory*. PhD thesis, Ecole Polytechnique X, 2011.

[27] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: Algebraic Specification in Action*, volume 2 of *Advances in Formal Methods*, pages 3–167. Springer, 2000.

[28] G. Gonthier, A. Mahboubi, L. Rideau, E. Tassi, and L. Théry. A modular formalisation of finite group theory. In K. Schneider and J. Brandt, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2007)*, volume 4732 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2007.

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

[30] T. Hales. Formal abstracts in mathematics. In F. Rabe, W. M. Farmer, G. O. Passmore, and Y. Abdou, editors, *Intelligent Computer Mathematics (CICM 2018)*, volume 11006 of *Lecture Notes in Computer Science*, page xiii. Springer, 2018.

[31] J. Harrison. HOL Light: An overview. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2009)*, volume 5674 of *Lecture Notes in Computer Science*, pages 60–66. Springer, 2009.

[32] L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15:81–91, 1950.

[33] M. Iancu. *Towards Flexiformal Mathematics*. PhD thesis, Jacobs University Bremen, 2017.

[34] Idris: A Language for Type-Driven Development. https://www.idris-lang.org/. Accessed on 3 January 2025.

[35] F. Kachapova. Formalizing groups in type theory. *Computing Research Repository (CoRR)*, abs/2102.09125, 2021.

[36] M. Kohlhase. The Flexiformalist Manifesto. In A. Voronkov, V. Negru, T. Ida, T. Jebelean, D. Petcu, S. M. Watt, and D. Zaharie, editors, *14th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*, pages 30–35. IEEE Computer Society, 2012.

[37] M. Kohlhase, T. Koprucki, D. Müller, and K. Tabelow. Mathematical models as research data via flexiformal theory graphs. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics (CICM 2017)*, volume 10383 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2017.

[38] M. Kohlhase, F. Rabe, and V. Zholudev. Towards MKM in the large: Modular representation and scalable software architecture. In S. Autexier, J. Calmet, D. Delahaye, P. D. F. Ion, L. Rideau, R. Rioboo, and A. P. Sexton, editors, *Intelligent Computer Mathematics (CICM 2010)*, volume 6167 of *Lecture Notes in Computer Science*, pages 370–384. Springer, 2010.

[39] Metamath Proof Explorer Home Page. http://us.metamath.org/mpeuni/mmset.html. Accessed on 3 January 2025.

[40] T. Mossakowski, C. Maeder, and K. Lüttich. The heterogeneous tool set. In O. Grumberg and M. Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007)*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer, 2007.

[41] R. Nakajima and T. Yuasa, editors. *The IOTA Programming System*, volume 160 of *Lecture Notes in Computer Science*. Springer, 1983.

[42] A. Naumowicz and A. Korniłowicz. A brief overview of Mizar. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2009)*, volume 5674 of *Lecture Notes in Computer Science*, pages 67–72. Springer, 2009.

[43] 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*. North

Holland, 1994.

[44] R. Nickson, O. Traynor, and M. Utting. Cogito Ergo Sum — Providing structured theorem prover support for specification formalisms. In K. Ramamohanarao, editor, *Proceedings of the Nineteenth Australasian Computer Science Conference (ACSC 1996)*, volume 18 of *Australian Computer Science Communications*, pages 149–158, 1996.

[45] U. Norell. *Towards a Practical Programming Language based on Dependent Type Theory*. PhD thesis, Chalmers University of Technology, 2007.

[46] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *Computer Aided Verification (CAV 1996)*, volume 1102 of *Lecture Notes in Computer Science*, pages 411–414. Springer, 1996.

[47] S. Owre and N. Shankar. Theory interpretations in PVS. Technical Report NASA/CR-2001-211024, NASA, 2001.

[48] L. C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer, 1994.

[49] L. C. Paulson. Formalising mathematics in simple type theory. In S. Centrone, D. Kant, and D. Sarikaya, editors, *Reflections on the Foundations of Mathematics*, volume 407 of *Synthese Library*, pages 437–453. Springer, 2019.

[50] L. C. Paulson. Large-scale formal proof for the working mathematician — lessons learnt from the ALEXANDRIA Project. *Computing Research Repository (CoRR)*, abs/2305.14407, 2023.

[51] Proof Power. http://www.lemma-one.com/ProofPower/index/index.html. Accessed on 3 January 2025.

[52] F. Rabe and M. Kohlhase. A scalable module system. *Information and Computation*, 230:1–54, 2013.

[53] F. Rabe and C. Schürmann. A practical module system for LF. In J. Cheney and A. Felty, editors, *Proceedings of the Fourth International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP 2009)*, pages 40–48. ACM Press, 2009.

[54] The Rocq Prover. https://rocq-prover.org/. Formerly the Coq Proof Assistant; accessed on 9 May 2025.

[55] J. Rushby, F. von Henke, and S. Owre. An introduction to formal specification and verification using EHDM. Technical Report SRI-CSL-91-02, SRI International, 1991.

[56] D. M. Russinoff. A formalization of finite group theory. *Computing Research Repository (CoRR)*, abs/2205.13347, 2022.

[57] D. Sannella and M. Wirsing. A kernel language for algebraic specification and implementation. In M. Karpinski, editor, *Fundamentals of Computation Theory (FCS 1983)*, volume 158 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 1983.

[58] D. R. Smith. KIDS: A knowledge-based software development system. *IEEE Transactions on Software Engineering*, 16:483–514, 1991.

[59] Y. V. Srinivas and R. Jüllig. Specware: Formal support for composing software. In

B. Möller, editor, *Mathematics of Program Construction (MPC 1995)*, volume 947 of *Lecture Notes in Computer Science*, pages 399–422. Springer, 1995.

[60] X. Yu, A. Nogin, A. Kopylov, and J. Hickey. Formalizing abstract algebra in type theory with dependent records. In *16th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2003)*, Emerging Trends Proceedings, pages 13–27, 2013.

[61] A. Zipperer. *A Formalization of Elementary Group Theory in the Proof Assistant Lean*. PhD thesis, Carnegie Mellon University, 2016.

Received 13 December 2024