Equational Logic and Set-Theoretic Models for Multi-Languages*

Samuele Buro[1], Roy L. Crole[2], and Isabella Mastroeni[1]

[1] Department of Computer Science, University of Verona
Strada le Grazie 15, 37134 Verona, Italy
{samuele.buro,isabella.mastroeni}@univr.it
[2] Department of Computer Science, University of Leicester,
University Road, Leicester LE1 7RH, United Kingdom
rlc3@le.ac.uk

Abstract. Interoperability is the capability of two languages to interact within a single system: HTML, CSS, and JavaScript can work together to render webpages. Some object oriented languages have interoperability via a virtual machine host (.NET CLI compliant languages in the Common Language Runtime). A high-level language can be interoperable with a low-level one (Apple's Swift and Objective-C). While there has been some research in the foundations of interoperability there is little supporting theory.
This paper is based upon our existing work on combining languages to produce so-called *multi-languages*. Here, we define an *equational logic* for deducing valid equations, from axioms that postulate properties of the multi-language. We define *set-theoretic multi-language algebras* as models, and provide algebraic constructions such as *congruences* and *quotient algebras*. Such models, and the constructions, provide the ingredients for the main deliverable, *soundness* and *completeness* for the equational logic. We illustrate the basic ideas with a running example.

Keywords: Multi-languages · Equational Logic · Set-Algebras.

1 Introduction

Multi-languages arise by combining existing languages [23, 1, 28, 10, 15, 22, 21, 19]. For instance, the multi-language in [19] allows programmers to interchange ML expressions and Scheme expressions. Benefits are code reuse and software interoperability. Unfortunately, they come at the price of a lack of clarity of (formal) properties of the new multi-language, mainly semantic specifications. Developing such properties is a key focus of this paper.

But what exactly *are* multi-languages? What is a good abstract definition of a multi-language? How should we be thinking conceptually? The problem was originally addressed in [19]. The authors introduced *boundary functions*, new

constructs able to regulate the flow of values between the underlying languages. Buro, Crole, and Mastroeni [8, 6] extended their approach to the broader class of *order-sorted signatures*: rather than combining two fixed languages, they combine two such signatures Sg_1 and Sg_2. The result is a notion of *multi-language term*. Such a term contains symbols from both signatures and also *boundary function symbols* to formally specify the interchange of *multi-language terms*.

Here we focus on equational algebras [29, 5, 11, 17, 12] and we provide and study suitable algebras for modelling multi-language terms, which we call *multi-language algebras*. Inspired by the classic results of [5], we develop in detail a number of analogues of standard concepts such as algebra congruences and quotients. These are put to use in a completeness proof for a system of equational reasoning in the multi-language setting, which we now describe.

Research on *equational reasoning and logic* has been prolific (see [30, 18]). There are many *sound* and *complete* derivation systems if Sg is a *many* or *order*-sorted signature: see [5], [11], and [12]. So another challenge lies in devising deduction systems that allow for sound reasoning between multi-language terms. We address this challenge by defining a *multi-language equational logic*. In more detail we give a deduction system for equations between multi-language terms. The main result is that our new system is sound and complete for deductions starting from a set of axiom equations, relative to the *multi-language algebras*.

Contributions and Paper Structure: In Section 2 we review the theory of order-sorted equational logic [12] and the multi-language framework of [6]. We adopt the Birkhoff approach to proving soundness and completeness, that is, we formulate multi-language algebras which are free, or quotients of other algebras. In Section 3 we describe the *free multi-language algebra* construction that yields the notion of *multi-language terms with variables*. Then we extend constructions of universal algebra to the multi-language context. In particular, we define the notion of *multi-language quotient algebra*. In Section 4, we give definitions of *multi-language equation* and *equation satisfaction*, and we sketch a proof of soundness and completeness. We need to assume familiarity with ordered structures for reasons of space (we redirect the reader to [12] for an introduction).

Running Example: Throughout the paper we develop a small illustrative application example that lays the foundations for combining a core imperative language with a functional one (the simply-typed lambda-calculus). In particular, we show how we can provide a formal semantics to multi-language terms accompanied by an equational theory. For instance, we formally define the semantics of the multi-language term `x = (λy:int . z * y) 5` obtained by blending a lambda-term into an imperative program.

2 Background

2.1 Order-Sorted Algebras

The basic notions of *order-sorted algebras* [12] are *signature* and *algebra*. We assume readers are familiar with these ideas, but include definitions to set up notation.

Definition 1 (Order-Sorted Signature). *An **order-sorted signature** is a triple* $Sg \triangleq (S, \leq, \Sigma)$*, where* $(S, \leq)$ *is a **poset of sorts** and* Σ *is an* $(S^* \times S)$*-sorted **set*** $\Sigma \triangleq \{ \Sigma_{w,s} \mid (w,s) \in S^* \times S \}$ ***of operators**. If* $\sigma \in \Sigma_{w,s}$ *we call* σ *an **operator** (**symbol**) and* (w,s) *the **rank** of* σ*. If* $w = \varepsilon$*, we call the operator a **constant**, and otherwise a **function symbol**. Given* Σ*, we write* $f\colon w \to s$*, for function symbols, and* $k\colon s$ *for constants, as shorthands for the set memberships. These data satisfy:*

(1os) each constant has a unique rank (ε, s)*: we abbreviate the rank to* s *and say* sort *as a synonym; and*
*(2os) function symbols satisfy the following **monotonicity condition**: if we have* $f\colon w_1 \to s_1$ *and* $f\colon w_2 \to s_2$ *with* $w_1 \leq w_2$*, then* $s_1 \leq s_2$*.*

Definition 2 (Order-Sorted Algebra). *An **order-sorted** Sg**-algebra** $\mathbf{A}$ for an order-sorted signature* $Sg \triangleq (S, \leq, \Sigma)$ *is specified by an* S*-sorted set* $A \triangleq (\, [\![s]\!]_{\mathbf{A}} \triangleq A_s \mid s \in S \,)$ *of **carrier sets**, together with **interpretation functions** $[\![\sigma\colon w \to s]\!]_{\mathbf{A}}\colon [\![w]\!]_{\mathbf{A}} \to [\![s]\!]_{\mathbf{A}}$ for each operator* $\sigma\colon w \to s$*, where* $w \triangleq s_1 \ldots s_n$ *and* $[\![w]\!]_{\mathbf{A}} \triangleq [\![s_1]\!]_{\mathbf{A}} \times \cdots \times [\![s_n]\!]_{\mathbf{A}}$*. In the case that* $w = \varepsilon$*, so that* σ *is a constant* k*, we define* $[\![\varepsilon]\!]_{\mathbf{A}} \triangleq \{\bullet\}$*; then, instead of* $[\![k\colon s]\!]_{\mathbf{A}}\colon \{\bullet\} \to [\![s]\!]_{\mathbf{A}}$*, we write* $[\![k\colon s]\!]_{\mathbf{A}} \in [\![s]\!]_{\mathbf{A}}$*. These data satisfy:*

(1oa) $s \leq s'$ *implies* $[\![s]\!]_{\mathbf{A}} \subseteq [\![s']\!]_{\mathbf{A}}$*; and*
(2oa) if $f\colon w_1 \to s_1$ *and* $f\colon w_2 \to s_2$ *with* $w_1 \leq w_2$*, then* $[\![f\colon w_1 \to s_1]\!]_{\mathbf{A}}(a) = [\![f\colon w_2 \to s_2]\!]_{\mathbf{A}}(a)$ *for all* $a \in [\![w_1]\!]_{\mathbf{A}}$*.*

From now on, we drop the algebra subscript and the ranks of operator symbols within the semantic brackets whenever they are clear from the context.

Definition 3 (Order-Sorted Homomorphism). *Let* $Sg \triangleq (S, \leq, \Sigma)$ *be a signature and let* $\mathbf{A}$ *and* $\mathbf{B}$ *be* Sg*-algebras. An **order-sorted** Sg**-homomorphism** $h\colon \mathbf{A} \to \mathbf{B}$ is an* S*-sorted function* $h\colon A \to B$ *such that*

(1oh) $h_s([\![k]\!]_{\mathbf{A}}) = [\![k]\!]_{\mathbf{B}}$ *for each* $k\colon s$*;*
(2oh) $h_s \circ [\![f\colon w \to s]\!]_{\mathbf{A}} = [\![f\colon w \to s]\!]_{\mathbf{B}} \circ h_w$ *for each* $f\colon w \to s$*; and*
(3oh) $s \leq s'$ *implies* $h_s(a) = h_{s'}(a)$ *for each* $a \in [\![s]\!]_{\mathbf{A}}$*.*

The class of all the order-sorted Sg*-algebras and the class of all the order-sorted* Sg*-homomorphisms form a category denoted by* $\mathcal{OSA}lg_{Sg}$*.*

Among the algebras in $\mathcal{OSA}lg_{Sg}$, the **term** Sg**-algebra** $\mathbf{T}_{Sg}$ is freely generated from Sg in the usual way. The carrier set $[\![s]\!]_{\mathbf{T}_{Sg}}$ consists of the **(ground) terms** t **of sort** s. Function symbols f are modelled as interpretation functions which build new terms from old ones by "applying f"; for instance, $[\![f]\!]_{\mathbf{T}_{Sg}}(t) \triangleq f(t)$. Note that any term t can appear in several carrier sets of the term algebra: terms are *polymorphic*. A key property of signatures, related to polymorphism, is **regularity** (see [12] for the formal definition). Regularity ensures each term t has a unique **least sort** s_t.

Proposition 1. *If Sg is a regular signature, the term Sg-algebra $\mathbf{T}_{Sg}$ is an* initial object *in $\mathcal{OSA}lg_{Sg}$, that is, there is a unique homomorphism $h\colon \mathbf{T}_{Sg} \to \mathbf{A}$ for any Sg-algebra $\mathbf{A}$. The components $(\mathbf{T}_{Sg})_s$ are defined by (mutual, structural) induction, hence h_s can be defined by structural recursion. Since we are seeking a homomorphism, we* must have *$h_s(f(t_1,\ldots,t_n)) = [\![f]\!]_{\mathbf{A}}(h_w(t_1,\ldots,t_n))$ and $h_s(k) = [\![s]\!]_{\mathbf{A}}$. But this is exactly a structural recursion, and the unique one yielding a homomorphism. As such, it makes sense to define the (unique)* ***interpretation of a term*** *t, by way of its least sort, as*

$$\boxed{[\![t]\!]_{\mathbf{A}} \triangleq h_{\mathbf{s}_t}(t)}$$

Since our paper focuses in entirety on how the syntax of languages is combined, term algebras such as this one, and others, will play a major role.

2.2 Multi-Language Signatures and Algebras

A key role of a multi-language specification is to specify which terms of one language can be deployed in the other. As such, showing exactly how this is done is a key milestone in the paper. Such a choice amounts, formally, to a mapping between terms of one language and terms of the other, and it is usually defined between syntactic categories (*i.e.*, collection of terms) rather than single terms [22, 19, 21]. Therefore, in the algebraic context, it is tantamount to a relation specifying pairs of sorts with one sort in each language. We call such specifications *interoperability relations*. A *multi-language signature* is specified by two order sorted signatures together with an interoperability relation. The signature is used to determine the terms of the multi-language. Note that a multi-language signature *explicitly* provides users with the *original two* language specifications; but it is also used to define a "multi-language", that is, a new language of terms which is a combination of the two originals. Please note our notation: If S_1 and S_2 are two sets of sorts we denote by $S_1 + S_2$ their coproduct. If $s \in S_i$ with $i \triangleq 1, 2$, we write s_i for $(s, i) \in S_1 \times \{1\} \cup S_2 \times \{2\}$.

Definition 4 (Multi-Language Signature). *A* ***multi-language signature*** *$SG \triangleq (Sg_1, Sg_2, \ltimes)$ is defined by a pair of signatures $Sg_1 \triangleq (S_1, \leq_1, \Sigma_1)$ and $Sg_2 \triangleq (S_2, \leq_2, \Sigma_2)$ and a binary* ***interoperability relation*** *$\ltimes$ over $S_1 + S_2$ such that $s_i \ltimes s'_j$ with $i, j \in \{1, 2\}$ and $i \neq j$ (thus, due to our notation, in relationships $s_i \ltimes s'_j$ we have $s \in S_i$ and $s' \in S_j$).*

The idea is that if $s_i \ltimes s'_j$ and t is a term of sort s in Sg_i, then t can be used "as a term with with sort s' in the language generated by Sg_j". A *multi-language algebra* provides meaning to the underlying languages Sg_1 and Sg_2 and it specifies *exactly how* terms of sort s may be converted to terms of sort s' whenever $s_i \ltimes s'_j$. These specifications are called *boundary functions* [19].

Definition 5 (Multi-Language Algebra). *Let $SG \triangleq (Sg_1, Sg_2, \ltimes)$ be a multi-language signature. An SG-****algebra*** *$\mathbf{A}$ is given by*

– *a pair of order-sorted algebras* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *over* Sg_1 *and* Sg_2*; and*
– *a* ***boundary function*** $[\![s_i \ltimes s'_j]\!]_{\mathbf{A}} : [\![s]\!]_{\mathbf{A}_i} \to [\![s']\!]_{\mathbf{A}_j}$ *for each constraint* $s_i \ltimes s'_j$*.*

Example 1. Let $Sg(\mathsf{Imp})$ and $Sg(\lambda^{\to})$ be the signatures of a small imperative language and the simply-typed lambda-calculus. We may want to define the interoperability relation between these two languages in order to use $\lambda^{\to}$-terms as expressions in Imp and vice versa. As a result, we can achieve multi-language programs such as `x = (λy:int . z * y) 5` (the multi-language terms are formally introduced in Definition 8). Note that, despite the simplicity, the evaluation of such a term requires a careful translation of meanings between the underlying languages: In order to obtain the value to assign to `x`, we first need to compute the $\lambda^{\to}$ application, which in turn needs the interpretation of `5` as a $\lambda^{\to}$-term. Such conversions are specified by the boundary functions.

Homomorphisms between multi-language algebras are ordinary algebraic homomorphisms that also commute with the boundary functions.

Definition 6 (Multi-Language Homomorphism). *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two multi-language* $SG \triangleq (Sg_1, Sg_2, \ltimes)$*-algebras. A* SG***-homomorphism*** $h : \mathbf{A} \to \mathbf{B}$ *is given by a pair of order-sorted homomorphisms* $h_1 : \mathbf{A}_1 \to \mathbf{B}_1$ *and* $h_2 : \mathbf{A}_2 \to \mathbf{B}_2$ *such that they commute with boundary functions, namely, if* $s_i \ltimes s'_j$*, then* $(h_j)_{s'} \circ [\![s_i \ltimes s'_j]\!]_{\mathbf{A}} = [\![s_i \ltimes s'_j]\!]_{\mathbf{B}} \circ (h_i)_s$*.*

We can define an S*-sorted set* A *by setting* $A_{s_i} \triangleq [\![s]\!]_{\mathbf{A}_i}$*; and given any such* SG*-homomorphism* h*, there is an* S*-sorted homomorphism* $\overline{h} : A \to B$ *given by* $\overline{h}_{s_i} \triangleq (h_i)_s : A_{s_i} \to B_{s_i}$ *(which commutes with boundary functions).*

Lemma 1. *The mapping* $h \mapsto \overline{h}$ *is well-defined, a bijection, and functorial in that* $\overline{h \circ h'} = \overline{h} \circ \overline{h'}$*. This is immediate from the definitions. Note that we will usually write* h *for* $\overline{h}$*, thus identifying the two concepts, and regard* $h : \mathbf{A} \to \mathbf{B}$ *and* $h : A \to B$ *as inter-changeable throughout the paper.*

A multi-language signature gives rise to an order-sorted one by adding **conversion operators** $\hookrightarrow_{s_i, s'_j}$ for each interoperability relation $s_i \ltimes s'_j$: each conversion operator maps a term of type s_i (informally, a term built from Sg_i) into a term of type s'_j (informally, a term built from Sg_j).

Definition 7 (Associated Signature). *Let* $SG \triangleq (Sg_1, Sg_2, \ltimes)$ *be a multi-language signature. The* ***associated signature*** $\overline{SG} \triangleq (S, \leq, \Sigma_{\overline{SG}})$ *of* SG *is the order-sorted signature defined as follows:*

– *the poset* $(S, \leq)$ *of sorts is given by* $S \triangleq S_1 + S_2$ *and by defining* $s_i \leq r_j$ *if and only if* $i = j$ *and* $s \leq_i r$*; and*
– *the order-sorted set of operators* $\Sigma_{\overline{SG}}$ *is defined by the clauses*
  - *if* $f : w \to s$ *is a function symbol in* Σ_i *for some* $i \triangleq 1, 2$*, then* $f_i : w_i \to s_i$ *is a function symbol in* $\Sigma_{\overline{SG}}$*;*
  - *if* $k : s$ *is a constant in* Σ_i *for some* $i \triangleq 1, 2$*, then* $k_i : s_i$ *is in* $\Sigma_{\overline{SG}}$*; and*
  - *a* ***conversion operator*** $\hookrightarrow_{s_i, s'_j} : s_i \to s'_j$ *in* $\Sigma_{\overline{SG}}$ *for each* $s_i \ltimes s'_j$*.*

A multi-language signature is defined **regular** if and only if the underlying signatures are regular. An immediate consequence is that if SG is regular, then so too is $\overline{SG}$. In the multi-language context, the **term SG-algebra $\mathbf{T}_{SG}$** defines the terms resulting from the combination of the underlying languages:

Definition 8 (Term SG-Algebra $\mathbf{T}_{SG}$). *Let $SG \triangleq (Sg_1, Sg_2, \ltimes)$ be a multi-language signature. The **term** SG-algebra $\mathbf{T}_{SG}$ is defined as follows:*

- *the order-sorted Sg_i-algebra $(\mathbf{T}_{SG})_i$ is specified by:*
  - $[\![s]\!]_{(\mathbf{T}_{SG})_i} \triangleq [\![s_i]\!]_{\mathbf{T}_{\overline{SG}}}$ *for each $s \in S_i$;*
  - $[\![k]\!]_{(\mathbf{T}_{SG})_i} \triangleq [\![k_i]\!]_{\mathbf{T}_{\overline{SG}}}$ *for each $k\colon s$ in Σ_i; and*
  - $[\![f\colon w \to s]\!]_{(\mathbf{T}_{SG})_i} \triangleq [\![f_i\colon w_i \to s_i]\!]_{\mathbf{T}_{\overline{SG}}}$ *for each $f\colon w \to s$ in Σ_i;*
- *boundary functions are defined by $[\![s_i \ltimes s'_j]\!]_{\mathbf{T}_{SG}} \triangleq [\![\hookrightarrow_{s_i,s'_j}]\!]_{\mathbf{T}_{\overline{SG}}}$ for each $s_i \ltimes s'_j$.*

*A **multi-language ground term** t is any element of $(T_{SG})_{s_i} \triangleq [\![s]\!]_{(\mathbf{T}_{SG})_i} \triangleq [\![s_i]\!]_{\mathbf{T}_{\overline{SG}}}$ for some s_i; $\mathbf{T}_{\overline{SG}}$ is of course the "standard" term order-sorted algebra over order-sorted $\overline{SG}$, and we will say that t **has sort** s_i.*

It follows that each multi-language ground term t has a unique least sort s_t when the signature is regular, being the least sort of t over the associated signature (and of course t may inhabit many other carrier sets).

The category whose objects are multi-language SG-algebras and morphisms are multi-language SG-homomorphisms is denoted by $\mathcal{MLA}lg_{SG}$. Moreover, as in the order-sorted case, the multi-language term algebra $\mathbf{T}_{SG}$ over a regular multi-language signature SG is initial in $\mathcal{MLA}lg_{SG}$ [6]. Therefore, given a multi-language SG-algebra $\mathbf{A}$, there is a unique (semantic) function $h\colon \mathbf{T}_{SG} \to \mathbf{A}$ defined by recursion over the syntax of terms. Analogous to the order-sorted case, the multi-language semantics of a ground term t is defined by $[\![t]\!]_{\mathbf{A}} \triangleq h_{\mathsf{s}_t}(t)$.

Example 2. Multi-language ground terms of our running example are exactly those that can be obtained by performing cross-language substitutions of an Imp-expression for a $\lambda^{\to}$-term and vice versa. The initiality of the term algebra ensures a unique meaning for each of these multi-language terms. For instance, if we compute the semantics of `x = (λy:int . z * y) 5`, we obtain exactly the expected function $\rho \mapsto \rho[\mathtt{x} \leftarrow\!\shortmid \rho(\mathtt{z}) * 5]$, where ρ is the environment.

3 Universal Constructions in a Multi-Language Context

3.1 Multi-Language Free Algebra

To formally define the terms of our multi-languages we shall make use of the concept of *free algebras*. First we review the abstract concept, and then we show that "terms built using variables" are a concrete instance of a free algebra. The universal property of a free algebra provides a convenient tool for defining and working with term substitutions.

A *free algebra* is the loosest algebra generated from an S-sorted X (whose elements are understood as *variables*). As usual [27, 25], the free algebra yields

terms with variables (as opposed to *ground terms* in the term algebra). First recall the notation $\overline{h}$ in Definition 6 and Lemma 1; after the next definition we always write h for any $\overline{h}$.

Definition 9 (Free SG-Algebra $\mathbf{F}(X)$ over X). *Let X be an S-sorted set of variables. A **free** SG-algebra $\mathbf{F}(X)$ **over** X is an SG-algebra $\mathbf{F}(X)$ together with an S-sorted function $\eta_X \colon X \to F(X)$ such that the following universal property is satisfied:*

- *For each SG-algebra $\mathbf{A}$, if $a \colon X \to A$ is an S-sorted function, then there exists a unique multi-language SG-homomorphism $a^* \colon \mathbf{F}(X) \to \mathbf{A}$ making the following diagram commute:*

$$\begin{array}{ccc} X & \xrightarrow{\eta_X} & F(X) \\ & \searrow^{a} & \downarrow{\scriptstyle \overline{a^*}} \\ & & A \end{array} \qquad\qquad \begin{array}{c} \mathbf{F}(X) \\ \downarrow{\scriptstyle a^*} \\ \mathbf{A} \end{array} \tag{1}$$

The previous definition does not guarantee the existence of such a free SG-algebra over X. However, if it does exist it is unique up to a unique isomorphism. We now provide a two-step syntactic construction, establishing existence. See, for instance, [9, Definition 1.3.2], [13, p. 72], and [12, p. 14] for a free algebra construction in the one-sorted, many-sorted, and order-sorted worlds. We now define a specific instance of $\mathbf{F}(X)$, denoted by $\mathbf{T}(X)$, and constructed out of syntax.

Definition 10 (Syntactic Free SG-Algebra $\mathbf{T}(X)$ over X).

1. *Let SG be a multi-language signature and X an S-sorted family of variables $(X_{s_i} \mid s_i \in S)$, each component also disjoint from the symbols in SG. By setting $X_i \triangleq ((X_i)_s \triangleq X_{s_i} \mid s \in S_i)$ we obtain an S_i-sorted set of variables for $i = 1, 2$. The **multi-language signature** $SG(X) \triangleq (Sg_1(X_1), Sg_2(X_2), \ltimes)$ **over** X has the same interoperability relation $\ltimes$ as SG, along with order sorted signatures $Sg_i(X_i) \triangleq (S_i, \leq_i, \Sigma_i(X_i))$. These are defined with the same poset $(S_i, \leq_i)$ of sorts of Sg_i, along with $(S_i^* \times S_i)$-sorted sets $\Sigma_i(X_i)$ of operator symbols specified by*
   - *if operator $\sigma \colon w \to s$ is in Σ_i, then $\sigma \colon w \to s$ is also in $\Sigma_i(X_i)$; and*
   - *if $x \in (X_i)_s$, then $x \colon s$ is in $\Sigma_i(X_i)$.*

   Informally, $\Sigma_i(X_i)$ consists of all Σ_i operators and, for all $s \in S_i$, all variables of sort s_i.
2. *The **free** SG-algebra $\mathbf{T}(X)$ **over** X is defined by making direct use of the term $SG(X)$-algebra $\mathbf{T}_{SG(X)}$ (use an instance of Definition 8). The order-sorted Sg_i-algebras $\mathbf{T}(X)_i$, for $i \triangleq 1, 2$, are defined by*
   - *$[\![s]\!]_{\mathbf{T}(X)_i} \triangleq [\![s]\!]_{(\mathbf{T}_{SG(X)})_i}$ for each $s \in S_i$;*
   - *$[\![k]\!]_{\mathbf{T}(X)_i} \triangleq [\![k]\!]_{(\mathbf{T}_{SG(X)})_i}$ for each $k \colon s$ in Σ_i; and*
   - *$[\![f]\!]_{\mathbf{T}(X)_i} \triangleq [\![f]\!]_{(\mathbf{T}_{SG(X)})_i}$ for each $f \colon w \to s$ in Σ_i.*

And $[\![s_i \ltimes s'_j]\!]_{\mathbf{T}(X)} : [\![s]\!]_{\mathbf{T}(X)_i} \to [\![s']\!]_{\mathbf{T}(X)_j}$ *is defined by*

$$[\![s_i \ltimes s'_j]\!]_{\mathbf{T}_{SG(X)}} : [\![s]\!]_{(\mathbf{T}_{SG(X)})_i} \to [\![s']\!]_{(\mathbf{T}_{SG(X)})_j}$$

Given X *and* SG*, a* ***multi-language term*** t *is any element of the set* $T(X)_{s_i} \triangleq [\![s]\!]_{\mathbf{T}(X)_i} \triangleq [\![s_i]\!]_{\mathbf{T}_{\overline{SG(X)}}}$ *for some* s_i*;* $\mathbf{T}_{\overline{SG(X)}}$ *is of course the "standard" term order-sorted algebra over order-sorted* $\overline{SG(X)}$*, and we will say that* t ***has sort*** s_i*.*

Example 3. Terms in the multi-language free algebra can be thought as ordinary multi-language terms with "holes", where a hole is given by an algebraic variable. For instance, `x = (λy:int . z * y)` v is not ground since any Imp-expression can be plugged into v. Terms with free algebraic variables are the basis for axiomatizing signatures by equations, leading to fully-fledged algebraic theories.

Theorem 1. *Let* SG *be a regular multi-language signature. Then, the* SG*-algebra* $\mathbf{T}(X)$ *is free over* X *in* $\mathcal{MLAlg}_{SG}$*.*

3.2 Multi-Language Quotient Algebra

The aim of this section is to introduce and extend well-known constructions of universal algebra to multi-languages. Apart from this being interesting in its own right, quotient algebras are central to our completeness proof. Indeed, we follow the Birkhoff's approach [5] and we establish the mathematical machinery needed to prove the completeness theorem in the classical way. We define the notions of *congruence* and *quotient algebra*. A multi-language signature SG is **locally filtered** (resp., **coherent**) if its associated signature $\overline{SG}$ is locally filtered (resp., coherent). We shall sometimes need these properties for results to hold: and we will require coherence for a well-defined notion of equation later in the paper. The reader may want to consult [12] for details.

A *congruence (relation)* is an equivalence relation on the carrier sets of a given algebra which is compatible with its structure.

Definition 11 (Congruence). *Let* $\mathbf{A}$ *be a multi-language* SG*-algebra. A* ***multi-language*** SG***-congruence*** $\equiv \triangleq (\equiv_{s_i} \subseteq [\![s_i]\!]_{\mathbf{A}} \times [\![s_i]\!]_{\mathbf{A}} \mid s_i \in S)$ *is an* S*-sorted family of equivalence relations such that*

(1co) $\equiv|_{S_i} \triangleq (\equiv_s \triangleq \equiv_{s_i} \mid s \in S_i)$ *is an* Sg_i-congruence *[12] on* $\mathbf{A}_i$ *for* $i \triangleq 1, 2$*; and*

(2co) $s_i \ltimes s'_j$ *and* $a \equiv_{s_i} a'$ *implies* $[\![s_i \ltimes s'_j]\!]_{\mathbf{A}}(a) \equiv_{s'_j} [\![s_i \ltimes s'_j]\!]_{\mathbf{A}}(a')$ *for each* $a, a' \in [\![s_i]\!]_{\mathbf{A}}$*.*

The next definition lifts the notion of *quotient algebra* to multi-languages. Intuitively, a quotient algebra is the outcome of the partitioning of an algebra through a congruence relation.

Definition 12 (Quotient Algebra). *Let* $\mathbf{A}$ *be a multi-language algebra over a locally filtered multi-language signature* SG*. Given an* SG*-congruence* $\equiv$*, the* ***quotient*** *of* $\mathbf{A}$ *induced by* $\equiv$ *is the* SG*-algebra* $\mathbf{A}/\equiv$ *defined as follows:*

(1q) $(\mathbf{A}/\equiv)_i$ *is the* order-sorted quotient Sg_i-algebra *[12]* $\mathbf{A}_i/(\equiv|_{S_i})$*; and*
(2q) $s \ltimes s'$ *implies* $[\![s_i \ltimes s'_j]\!]_{\mathbf{A}/\equiv}([a]) = [[\![s_i \ltimes s'_j]\!]_{\mathbf{A}}(a)]$.

For each quotient algebra $\mathbf{A}/\equiv$ of the multi-language algebra $\mathbf{A}$, the homomorphic *quotient map* $q\colon \mathbf{A} \to \mathbf{A}/\equiv$ is defined by $q(a) = [a]$. Moreover, the kernel of q is exactly $\equiv$. Following Definition 6 and Lemma 1 we shall also write q for the order-sorted homomorphism $q\colon A \to A/\equiv$.

4 Equational Logic

We now give an equational logic for multi-language terms. We define equations between multi-language terms and show that our system is sound and complete for SG-algebras; we also show the existence of an initial algebra. Through conversion operators $\hookrightarrow$, multi-language equations can usefully axiomatize boundary functions.

Definition 13 (Multi-Language Equations). *Let* $SG \triangleq (Sg_1, Sg_2, \ltimes)$ *be a coherent multi-language signature,* X *be an* S*-sorted set of variables, and* $\mathbf{T}(X)$ *the free* SG*-algebra over* X*.*

1. *An* SG-equation *over* X *is a judgement of the form* $t =_X t'$*, such that* $t \in T(X)_{s_i}$ *and* $t' \in T(X)_{s'_i}$ *for some* $s, s' \in S_i$*, where* s *and* s' *are connected via* $\leq_i$ *(equivalently, least sorts* s_t *and* $\mathsf{s}_{t'}$ *are connected in* S_i*). Note that* $t =_X t$ *is an* SG*-equation. The connectedness of* s_t *and* $\mathsf{s}_{t'}$*, and coherence which ensures both terms have a super-sort, is necessary to ensure that forthcoming definitions of equation satisfaction are well-defined.*
2. *A* conditional SG-equation *over* X *is a judgement of the form* $\mathscr{E} \Rightarrow t =_X t'$*, such that* $t =_X t'$ *is an* SG*-equation and* $\mathscr{E}$ *is a set of* SG*-equations over* X*. An unconditional equation* $t =_X t'$ *is* $\varnothing \Rightarrow t =_X t'$*.*

We write AX *for any set of conditional and/or unconditional equations.*

Example 4. In order to axiomatize the behaviour of boundary functions that allow the use of Imp-expressions in place of $\lambda^{\to}$-terms and vice versa, we provide equations which define the conversion of values. Intuitively, we want to move the meaning of terms between the languages only when no more computational steps are possible. Let $\mathbb{e}$ be the sort of $\lambda^{\to}$-terms and $\mathbb{exp}$ the sort of Imp-expressions:

$$\hookrightarrow_{\mathbb{e},\mathbb{exp}}(i) = i = \hookrightarrow_{\mathbb{exp},\mathbb{e}}(i) \qquad \forall i \in \{\ldots, \texttt{-1}, \texttt{0}, \texttt{1}, \ldots\}$$
$$\hookrightarrow_{\mathbb{e},\mathbb{exp}}(x) = x = \hookrightarrow_{\mathbb{exp},\mathbb{e}}(x) \qquad \forall x \in \mathit{Var}$$
$$\hookrightarrow_{\mathbb{exp},\mathbb{e}}(\texttt{true}) = \texttt{1}$$
$$\hookrightarrow_{\mathbb{exp},\mathbb{e}}(\texttt{false}) = \texttt{0}$$

The first equation allows the flow of integers between the imperative language and the simply-typed lambda-calculus. This is possible because both languages have the notion of integer values, but in more realistic cases, such a conversion should

take into consideration different machine-integer implementations, overflow, etc. A variation on theme can be obtained by assuming a lambda-calculus that does not provide a primitive notion of integer numbers. In that case, the boundary functions can move Imp-integers to a suitable numeral encoding in the lambda-calculus, *e.g.*, the Church encoding. The second equation establishes a correspondence between variable names of the two languages. Such a correspondence enables an implicit flow of values between Imp- and $\lambda^{\rightarrow}$-environments. For instance, in Example 2, the free $\lambda^{\rightarrow}$-variable z "becomes" an Imp-variable when we compute the semantics of the lambda term. Finally, the last two equations define the conversion of booleans to integers.

The next step is to show how new equations can be inductively generated (derived) from a set of axioms. The rules require the concept of substitution, namely a syntactic transformation that replaces variables with terms. Let X and Y be two sets of variables. A *variable substitution* is an S-sorted function $\theta\colon X \to T(Y)$: for each variable $x \in X_{s_i}$ we supply a term $\theta_{s_i}(x)$, of sort s_i, involving variables from Y. Then by the freeness of $T(X)$ (that is, freeness of $\mathbf{T}(X)$), there is a unique *substitution homomorphism* $\theta^*\colon T(X) \to T(Y)$ induced by θ such that $\theta^* \circ \eta_X = \theta$: the idea is that for any $t \in T(X)_{s_i}$, θ^* recurses over t and then finally substitutes $\theta_{s_i}(x)$ for each x occuring in t.

We shall adopt some further notation, used in the rules. If $a\colon X \to A$ where X is a set of variables, then any $x \in X_{s_i}$ has the unique sort s_i and we write $a(x) \triangleq a_{s_i}(x)$. Further, for any S-sorted homomorphism $h\colon T(X) \to A$, if $t \in T(X)_{s_i}$ then of course $h_{s_i}(t) \in A_{s_i}$. But, since there is a least sort s_t, also $t \in T(X)_{(\mathsf{s}_t)_i} \subseteq T(X)_{s_i}$, and since h is a homomorphism we have $h_{(\mathsf{s}_t)_i}(t) = h_{s_i}(t) \in A_{(\mathsf{s}_t)_i} \subseteq A_{s_i}$. From now on we write $h(t) \triangleq h_{(\mathsf{s}_t)_i}(t)$ (that is, $h(t) = h_{s_i}(t) \in A_{s_i}$) to reduce subscript notation.

Definition 14 (Equations Derived from Axioms). *Given a set AX of axioms, the inductive rules of equational logic allow the derivation of new* unconditional *equations. Our rules in Figure 1 are a modification of those in [12], and they make use of the following notation: t, t', and t'' are multi-language terms in $T(X)$, and $\theta, \theta'\colon X \to T(Y)$ are two variable substitutions from X to $T(Y)$ over a set Y of variables. Thus one also has $\theta^*, \theta'^*\colon T(X) \to T(Y)$ by freeness.*

If $t =_X t'$ is generated from AX then we write $AX \vdash t =_X t'$. The rule (CONG) intuitively says that replacing equals for equals yields new equalities, that is, term formation is a congruence. Thus we may build new equations using the "usual/informal" rules of equational reasoning. One easily proves admissible rule (SUB), namely that if $t =_X t'$ then $\theta^*(t) =_Y \theta^*(t')$; but note that the proof requires the following fact. The composition of two substitution homomorphisms $\theta_1^*\colon T(X) \to T(Y)$ and $\theta_2^*\colon T(Y) \to T(Z)$ is still a substitution homomorphism, since it is induced by $\theta_2^* \circ \theta_1$. To see that $\theta_2^* \circ \theta_1^* = (\theta_2^* \circ \theta_1)^*$ note that

$$(\theta_2^* \circ \theta_1)^* \circ \eta_X = \theta_2^* \circ \theta_1 = (\theta_2^* \circ \theta_1^*) \circ \eta_X$$

$$\frac{-}{t =_X t}\ (\textsc{ref}) \qquad \frac{t =_X t'}{t' =_X t}\ (\textsc{sym}) \qquad \frac{t =_X t' \quad t' =_X t''}{t =_X t''}\ (\textsc{trans})$$

$$\frac{\forall x \in \bigcup_{s_i \in S} X_{s_i} \,.\, \theta(x) =_Y \theta'(x)}{\theta^*(t) =_Y \theta'^*(t)}\ (\textsc{cong})$$

$$\frac{\forall u =_X u' \in \mathscr{E} \,.\, \theta^*(u) =_Y \theta^*(u')}{\theta^*(t) =_Y \theta^*(t')}\ [\mathscr{E} \Rightarrow t =_X t' \in AX] \quad (\textsc{axsub})$$

Fig. 1. Inference Rules for Inductively Defining Equational Logic.

4.1 Soundness, Completeness, and Freeness Results

Definition 15 (Satisfaction). *Let* $\mathbf{A}$ *be a multi-language algebra over a regular multi-language signature* SG*. We refer to* SG*-equations as* equations*.* $\mathbf{A}$ satisfies $\mathscr{E} \Rightarrow t =_X t'$*, denoted* $\mathbf{A} \models \mathscr{E} \Rightarrow t =_X t'$*, if for each assignment function* $a\colon X \to A$ *such that* $a^*\colon T(X) \to A$ *equalizes each equation in* $\mathscr{E}$*, a also equalizes* t *and* t'*,* i.e.*,* $a^*(t) = a^*(t')$*. Also,* $\mathbf{A} \models t =_X t'$ *just in case each* $a\colon X \to A$ *equalizes* t *and* t'*. Let* AX *be a set of conditional and/or unconditional equations. If an* SG*-algebra* $\mathbf{A}$ *satisfies each equation in* AX *we write* $\mathbf{A} \models AX$*. Think of* AX *as a set of* ***axioms*** *satisfied (modelled) by the* SG*-algebra* $\mathbf{A}$*.*

One may generalize the above fact about substitution homomorphisms to arbitrary assignment functions and SG-homomorphisms. We do so in the next lemma, which is used in the proof of soundeness and completeness.

Lemma 2. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two* SG*-algebras over a regular multi-language signature* $SG \triangleq (Sg_1, Sg_2, \ltimes)$*. If* $a\colon X \to A$ *is an assignment function and* $h\colon \mathbf{A} \to \mathbf{B}$ *is an* SG*-homomorphism, then* $(h \circ a)^* = h \circ a^*\colon \mathbf{F}(X) \to \mathbf{B}$ *(or equivalently* $(h \circ a)^* = h \circ a^*\colon F(X) \to B$*).*

Theorem 2 (Soundness and Completeness). *Let* SG *be a coherent signature and* AX *a set of axioms and* $t =_X t'$ *any equation. Then* $AX \vdash t =_X t'$*, just in case, for every* SG*-algebra* $\mathbf{A}$ *we have* $\mathbf{A} \models AX$ *implies* $\mathbf{A} \models t =_X t'$*.*

Soundness is straightforward to prove. In order to prove completeness, we build a quotient algebra (see [5]) $\mathbf{T}_{AX}(X)$ on $\mathbf{T}(X)$ such that given terms t and t' in the same equivalence class, the equation $t =_X t'$ can be derived from AX. The S-sorted binary relation $\sim_{AX(X)}$ is defined on the carrier sets of $\mathbf{T}(X)$ as $t \sim_{AX(X),s} t'$ if $AX \vdash t =_X t'$ where $t, t' \in T(X)_s$. Trivially $\sim_{AX(X)}$ is an equivalence relation and a congruence on $\mathbf{T}(X)$. Hence we can build the quotient algebra $\mathbf{T}_{AX}(X) \triangleq \mathbf{T}(X)/\sim_{AX(X)}$. The elements of $\mathbf{T}_{AX}(X)$ are exactly equivalence classes of terms, provably equal from AX. Indeed, it is easy to prove that $[t] = [t']$ if and only if $AX \vdash t =_X t'$. Moreover we have the following proposition

Proposition 2. $\mathbf{T}_{AX}(X) \models t =_x t'$ *implies* $AX \vdash t =_x t'$.

The proof of completeness is then immediate if we can show that $\mathbf{T}_{AX}(X) \models AX$, for then $\mathbf{T}_{AX}(X) \models t =_x t'$ follows by the assumption in the Theorem.

The class of the multi-language SG-algebras satisfying a set AX of axioms is denoted by $\mathcal{MLM}od_{AX}$, namely the class of *models* satisfying AX. If we take all the SG-homomorphisms between them, we have that $\mathcal{MLM}od_{AX}$ is a *full subcategory* of $\mathcal{MLA}lg_{SG}$. In fact $\mathbf{T}_{AX}(X)$ is free in $\mathcal{MLM}od_{AX}$, where the freeness is defined along the lines of Definition 9.

Theorem 3. *Let SG be a coherent signature and AX a set of axioms. Then, the quotient algebra $\mathbf{T}_{AX}(X)$ is the free algebra over X in $\mathcal{MLM}od_{AX}$.*

Since initiality is a special case of the freeness property and $\mathbf{T}(\varnothing) = \mathbf{T}$, then $\mathbf{T}_{AX} = \mathbf{T}_{AX}(\varnothing)$ is an initial object in $\mathcal{MLA}lg_{AX}$.

5 Concluding Remarks

Others have investigated issues that arise when combining languages: [26, 4, 15, 31, 22] explore the combination of typed and untyped languages (Lua and ML [26], Java and PLT Scheme [15], or Assembly and a typed functional language [22]), focusing on typing issues and values exchanging techniques. Several works have focused on multi-language implementations: [14] provides a type system for a fragment of Microsoft Intermediate Language (IL) used by the .NET framework, that allows programmers to write components in several languages (C#, Visual Basic, VBScript, ...) which are then translated to IL. [16] proposes a virtual machine that can execute the composition of dynamically typed programming languages (Ruby and JavaScript) and a statically typed one (C). [3, 2] describes a multi-language runtime mechanism achieved by combining single-language interpreters of (different versions of) Python and Prolog. Finally, [7] investigates the mathematical link between signatures and grammars.

There is considerable work to explore in future. One might study language combinations where the sort (type) and term structure is much richer. Here we have looked only at equational logic, so the whole area of *operational semantics* for multi-languages should be explored. And of course as well as the syntactic languages we need also to explore suitable semantics (given as algebras in this paper). Since equational theories give rise to *free algebra monads* [24], further studies should investigate the possibility of extending/generalizing the results in this paper to a more abstract approach based on the notion of *monad* [20]. Finally, in recent work we have produced a sound and complete categorical semantics for multi-languages [8].

As a final remark, we have further results about the categories of set-theoretic models mentioned within our paper, which are not presented here due to space limitations. However they would hopefully appear in a journal version.

References

1. Ahmed, A., Blume, M.: An equivalence-preserving cps translation via multi-language semantics. SIGPLAN Not. **46**(9), 431–444 (Sep 2011)
2. Barrett, E., Bolz, C.F., Tratt, L.: Unipycation: A case study in cross-language tracing. In: Proceedings of the 7th ACM workshop on Virtual machines and intermediate languages. pp. 31–40. ACM, New York, NY, USA (2013)
3. Barrett, E., Bolz, C.F., Tratt, L.: Approaches to interpreter composition. Computer Languages, Systems & Structures **44**, 199–217 (2015)
4. Benton, N.: Embedded interpreters. Journal of functional programming **15**(4), 503–542 (2005)
5. Birkhoff, G.: On the structure of abstract algebras. In: Mathematical Proceedings of the Cambridge Philosophical Society. vol. 31, pp. 433–454. Cambridge University Press, Cambrdige, UK (Oct 1935)
6. Buro, S., Mastroeni, I.: On the multi-language construction. In: Caires, L. (ed.) Programming Languages and Systems. pp. 293–321. Springer, Berlin/Heidelberg, DE (2019)
7. Buro, S., Mastroeni, I.: On the semantic equivalence of language syntax formalisms. In: Proceedings of the 20th Italian Conference on Theoretical Computer Science. pp. 34–51. CEUR-WS.org (2019)
8. Buro, S., Mastroeni, I., Crole, R.L.: Equational logic and categorical semantics for multi-languages. In: In-press (accepted for publication at 36th International Conference on Mathematical Foundations of Programming Semantics — MFPS 2020) (2020)
9. Denecke, K., Wismath, S.L.: Universal Algebra and Coalgebra. World Scientific Publishing, Singapore (2009)
10. Furr, M., Foster, J.S.: Checking type safety of foreign function calls. SIGPLAN Not. **40**(6), 62–72 (Jun 2005)
11. Goguen, J., Meseguer, J.: Completeness of many-sorted equational logic. Houston Journal of Mathematics **11**(3), 307–334 (1985)
12. Goguen, J.A., Meseguer, J.: Order-sorted algebra i: equational deduction for multiple inheritance, overloading, exceptions and partial operations. Theoretical Computer Science **105**(2), 217–273 (1992)
13. Goguen, J.A., Thatcher, J.W., Wagner, E.G., Wright, J.B.: Initial algebra semantics and continuous algebras. Journal of the ACM **24**(1), 68–95 (1977)
14. Gordon, A.D., Syme, D.: Typing a multi-language intermediate code. In: Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001. pp. 248–260. ACM, New York, NY, USA (2001)
15. Gray, K.E.: Safe cross-language inheritance. In: Vitek, J. (ed.) ECOOP 2008 – Object-Oriented Programming. pp. 52–75. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
16. Grimmer, M., Schatz, R., Seaton, C., Würthinger, T., Luján, M.: Cross-language interoperability in a multi-language runtime. ACM Trans. Program. Lang. Syst. **40**(2), 8:1–8:43 (May 2018)
17. Higgins, P.J.: Algebras with a scheme of operators. Mathematische Nachrichten **27**(1-2), 115–132 (1963)
18. Kirchner, C., Kirchner, H.: Equational logic and rewriting. In: Computational Logic. pp. 255–282 (2014)

19. Matthews, J., Findler, R.B.: Operational semantics for multi-language programs. ACM Transactions on Programming Languages and Systems **31**(3), 12:1–12:44 (2009)
20. Moggi, E.: Notions of computation and monads. Information and computation **93**(1), 55–92 (1991)
21. Osera, P.M., Sjöberg, V., Zdancewic, S.: Dependent interoperability. In: Claessen, K., Swamy, N. (eds.) Proceedings of the Sixth Workshop on Programming Languages Meets Program Verification. pp. 3–14. ACM, New York, NY, USA (2012)
22. Patterson, D., Perconti, J., Dimoulas, C., Ahmed, A.: Funtal: Reasonably mixing a functional language with assembly. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 495–509. ACM, New York, NY, USA (2017)
23. Perconti, J.T., Ahmed, A.: Verifying an open compiler using multi-language semantics. In: Proceedings of the 23rd European Symposium on Programming Languages and Systems. pp. 128–148. Springer, Berlin, DE (2014)
24. Plotkin, G.: Some varieties of equational logic. In: Algebra, Meaning, and Computation, pp. 150–156. Springer (2006)
25. Powell, W.B.: Ordered Algebraic Structures. CRC Press, Boca Raton, FL, USA (1985)
26. Ramsey, N.: Embedding an interpreted language using higher-order functions and types. In: Proceedings of the 2003 Workshop on Interpreters, Virtual Machines and Emulators. pp. 6–14. IVME '03, ACM, New York, NY, USA (2003)
27. Reiterman, J., Trnkovà, V.: Free structures. In: Herrlich, H., Porst, H.E. (eds.) Category Theory at Work. pp. 277–288. Heldermann Verlag, Berlin, DE (1991)
28. Tan, G., Morrisett, G.: Ilea: Inter-language analysis across Java and C. SIGPLAN Not. **42**(10), 39–56 (Oct 2007)
29. Tarski, A.: Equational logic and equational theories of algebras. In: Studies in Logic and the Foundations of Mathematics, vol. 50, pp. 275–288. Elsevier, Amsterdam, NL (1968)
30. Taylor, W.: Equational logic. Houston Journal of Mathematics **47**(2) (1979)
31. Wrigstad, T., Zappa Nardelli, F., Lebresne, S., Östlund, J., Vitek, J.: Integrating typed and untyped code in a scripting language. In: Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, January 17-23, 2010. pp. 377–388 (2010)