Equational Logic and Categorical Semantics for Multi-Languages

Samuele Buro\textsuperscript{a} and Roy Crole\textsuperscript{b} and Isabella Mastroeni\textsuperscript{a,1,2,3}

\textsuperscript{a}University of Verona, Italy and \textsuperscript{b}University of Leicester, UK

Abstract

Programming language interoperability is the capability of two programming languages to interact as parts of a single system. Each language may be optimized for specific tasks, and a programmer can take advantage of this. HTML, CSS, and JavaScript yield a form of interoperability, working in conjunction to render webpages. Some object oriented languages have interoperability via a virtual machine host (.NET CLI compliant languages in the Common Language Runtime, and JVM compliant languages in the Java Virtual Machine). A high-level language can interact with a lower level one (Apple’s Swift and Objective-C). While there has been some research exploring the interoperability mechanisms (Section 1) there is little development of theoretical foundations. This paper presents an approach to interoperability based around theories of equational logic, and categorical semantics.

We give ways in which two languages can be blended, and interoperability reasoned about using equations over the blended language. Formally, \textit{multi-language equational logic} is defined within which one may deduce valid equations starting from a collection of axioms that postulate properties of the combined language. Thus we have the notion of a \textit{multi-language theory} and much of the paper is devoted to exploring the properties of these theories. This is accomplished by way of \textit{category theory}, giving us a very general and flexible semantics, and hence a nice collection of models. Classifying categories are constructed, and hence equational theories furnish each categorical model with an internal language; from this we can also establish soundness and completeness. A set-theoretic semantics follows as an instance, itself sound and complete. The categorical semantics is based on some pre-existing research, but we give a presentation that we feel is easier and simpler to work with, improves and mildly extends current research, and in particular is well suited to computer scientists. Throughout the paper we prove some interesting properties of the new semantic machinery. We provide a small running example throughout the paper to illustrate our ideas, and a more complex example in conclusion.

Keywords: categorical logic, equational logic, interoperability, multi-languages, order-sorted signatures and theories, programming languages, subsort polymorphism.

1 Introduction

The theory of equational algebra has been a compelling topic since the early days of \textit{universal algebra} [33,2]. Research on \textit{equational logic}, addressing the problem of \textit{reasoning by deduction} about term equality, has been prolific (see [34,17] for surveys). In particular, many \textit{sound} and \textit{complete} deduction systems have arisen. For instance, if Sg is a \textit{one-sorted}, \textit{many-sorted}, or \textit{order-sorted} signature (sorts and function symbols), such systems appear in [2], [10], and [13], respectively. These developments have had a remarkable impact on \textit{operational semantics} and \textit{automatic theorem proving}. In particular, the pioneering works of [8,18] to operationalize equational deduction led to the theory of \textit{term rewriting systems} [15,35] which has extensive applications.

\textit{Multi-languages} are programming languages arising from the combination of already existing languages [27,1,32,9,14,26,24,21]. Intuitively, terms of multi-languages are obtained by performing cross-language substitutions. For instance, the multi-language designed in [21] allows programmers to replace ML expressions with Scheme expressions and vice versa. Potential benefits are code reuse and software interoperability. In order to provide a semantics, [21] introduces new constructs to regulate the flow of values between the underlying languages, the so-called \textit{boundary functions}.
But what are the formal properties and semantics of multi-languages? Indeed, how general can we be? And in any case, what is a good formal definition of a multi-language in the first place? Buro and Mastroeni [3] extended the approach of [21] to the broader class of order-sorted algebras, providing a systematic and more general way to define multi-languages, but they did not address equational reasoning. We do so here. In more detail, a multi-language is specified by combining two order-sorted signatures $S_1$ and $S_2$. But how exactly should the signatures be combined? And what is the formal equational theory of “term interoperability”? More precisely, since equations will not only be defined between two terms both of which are over just one of the $S_i$, but also defined between terms containing symbols from both $S_1$ and $S_2$, what is a good deduction system for such multi-language equations? We tackle these questions by specifying multi-language equational logic which is shown to be sound and complete.

Contributions in Detail: Conceptually we lift the basic syntactic theories of order-sorted equational logic [13], and models of the theories, to the algebraic multi-language framework defined by [3]. The models in [13] are built from sets, but we adapt the categorical approach in [20]. The main contribution is a deductive system for multi-languages with a sound and complete categorical semantics. We also prove some interesting semantic properties. There is a running example application throughout the paper, and an outline of a more elaborate application in Section 3.3 where we combine an imperative language and a lambda calculus. Our account of order-sorted equational theories builds on and refines [20], with all our deductive systems presented with uniform and clear inductive rules. Further, we include explicit type information in equality judgements, and include axioms that may be conditional equations. We give a simplified categorical semantics along with categorical type-theory correspondence and classifying category, and also give an explicit connection to free set-algebra semantics.

Remarks and Intuitions: We work with order-sorted signatures [13]. As such, our languages enjoy subsort polymorphism; there is no provision for parametric polymorphism. Such signatures may satisfy criteria known as monotonicity and regularity. Since the intuitions of these criteria are usually omitted from technical papers, we make some remarks. Language terms $t$ are built inductively by applying function symbols $f$ to existing terms (which begin as constants). Such symbols $f$ can be polymorphically sorted. As such, if one input sort of $f$ is a subsort of another input sort, we would like the term $t$ to be subsort polymorphic with respect to the output sorts. Monotonicity formalises this requirement. One needs to place some control over such polymorphism: one way of doing so is to apply a requirement whereby every term $t$ has a least sort. Regularity [13] is such a condition. To ensure every term $t$ to have a least sort, we might naively achieve this by requiring each subsort-polymorphic function symbol $f$ to have a least input and least output sort, hoping that $t$ would do so inductively. We would need to give all constants a least sort to start the induction; and we could simply stipulate that each constant has just one sort. In initially building terms, if there were any function symbol $f$ with a (polymorphic) input sort that is a strict subsort of the sort of a constant, such $f$ could not be applied to that constant. We are led to refine our naive idea: Fix any $f$. Consider only those constants that this $f$ can be polymorphically applied to. Now fix one such constant of sort $s$; this imposes a lower bound for the input sort of $f$, namely $s$. We then require that for all such polymorphic instances of $f$, there is one instance with a least input sort $s$, where of course $s \leq s$, and a least output sort. This requirement, formalised, is Regularity.

Paper Structure: In Section 2 we present a transparent rule based deduction system for order-sorted equational logic with conditional axioms, together with a categorical semantics which is proved sound and complete. In Section 3 we present a similar set of results for multi-languages. In Section 4 we give some further examples of multi-languages. In Section 3.3 we present an outline of an extended example in which a traditional lambda calculus and an imperative while language are blended as a single multi-language.

2 Order-Sorted Equational Logic

We review order-sorted equational theories (see for example [13,20]). Here we give an improved presentation that is syntactically simpler than in loc cit, and further we extend theories to include conditional equational axioms. We also present a detailed but stylistically improved summary of the categorical models from [20], along with a simpler construction of the classifying category (up to equivalence). We then prove a result relating the classifying category to free order-sorted algebras.

2.1 Order-Sorted Algebras

A set $S$ is usually regarded as a set of sorts or set of ground types. Often $S$ is partially ordered by $\leq$, and then $S^n \triangleq S \times \cdots \times S$ ($n$-times cartesian product for $n \geq 1$) inherits the pointwise order, with typical instances written $w \leq w'$. If $w \in S^n$, we usually make explicit its components by writing $w \triangleq s_1, \ldots, s_n$, sometimes referring to a sequence of sorts.

We write $(A_s \mid s \in S)$ for a family indexed by $S$ where each $A_s$ is sometimes a set, but more generally an object in a category $C$. We sometimes refer to the family as an $S$-sorted set (or, an $S$-sorted object). Such indexed families are simply functors $A$ in the presheaf category $Set^S(C^S)$. As such, an $S$-sorted function
(S-sorted morphism) \( h : A \to B \) is simply a morphism (that is, natural transformation) in \( 
abla = \text{Set}^S (C^2) \) where \( S \) is a set or poset.

In this paper all categories have finite products, and functors preserve them up to isomorphism. If \( A, B, \) and \( A_i (1 \leq i \leq n) \) are objects in a category \( C \), we write \( A \times B \) for the binary product of \( A \) and \( B, A_1 \times \cdots \times A_n \) or \( \prod f \) for the finite product of the \( A_i \), and \( I \) for the terminal object. Mediating morphisms for binary product are written \( (f, f') \), and as usual \( f \times f' \equiv (f \circ \pi, f' \circ \pi') \) (for suitable \( f \) and \( f' \) and the usual projections). We adopt the obvious extension of notation for finite products. If \( A \) is an S-sorted object and \( w \equiv \{ s_1, \ldots, s_n \} \), we denote \( A_w \) the product \( A_{s_1} \times \cdots \times A_{s_n} \). Likewise, if \( f \) is an S-sorted morphism, then the morphism \( f_w \) is defined by \( f_{s_1} \times \cdots \times f_{s_n} \). The coproduct object of \( A \) and \( B \) is written \( A + B \). We write \( l_1 \ldots l_n \) or \( l_1 \ldots , l_n \) for a typical finite list, and we may abbreviate just to \( l \). In the special case of a list of sorts \( s_1, s_2, \ldots, s_n \) we usually abbreviate to \( w \). When we define order-sorted signatures \( S_{g1}, S_{g2}, S_g, S'_g \), etc., we shall implicitly assume that their posets of sorts are denoted by \( (S_1, \leq_1), (S_2, \leq_2), (S', \leq'), (S', \leq'), \) etc., respectively.

Key ingredients of order-sorted equational theories are the definitions of signature and algebra. The former defines the symbols from which the terms of a language are built, and the latter provides terms with a meaning. This meaning can be both set-theoretic and category-theoretic [13,20].

**Definition 2.1 (Order-Sorted Signature)** An order-sorted signature \( S_g \) is specified by

- a poset \( (S, \leq) \) of sorts;
- a collection of function symbols \( f : s_1, \ldots, s_n \to s \) each with an arity \( a \geq 1 \) and \( (w, s) \in S^a \times S \) the rank of \( w \) where \( w \equiv s_1, \ldots, s_n \);
- a collection of constants \( k : s \), each of a unique rank \( s \) (just a single sort); and
- a monotonicity requirement that whenever \( f : w_1 \to s \) and \( f : w_2 \to r \) with \( w_1 \leq w_2 \), then \( s \leq r \).

By an operator we mean either a function symbol or a constant.

A key property of such signatures \( S_g \), related to polymorphism, is regularity. We shall shortly show how to build a set of terms out of \( S_g \), and regularity ensures that each term has a unique least sort [13, Proposition 2.10]. (All signatures in this paper are assumed regular.)

**Definition 2.2 (Regularity of an Order-Sorted Signature)** An order-sorted signature \( S_g \) is regular if for each \( f : w \to s \) and for each lower bound \( w_1 \leq w \) the set \( \{ (w', s') \mid f : w' \to s' \land w_1 \leq w' \} \subseteq S^a \times S \) has a minimum, called the least rank of \( f \) with respect to \( w_1 \).

**Raw terms** over a signature \( S_g \) are defined by \( t \equiv x | k | f(t_1, \ldots, t_n) \) with \( x \in \text{Var} \) (a countably infinite set of variables), \( k \) a constant, and \( f \) a function symbol with arity \( a \). A context is a finite list of ordered pairs \( x: s \) formed by a variable \( x \) and a sort \( s \) in \( S_g \). We usually define a context by writing \( \Gamma \equiv [x_1 : s_1, \ldots, x_n : s_n] \). We work with sorting judgements of the form \( \Gamma \vdash t : s \) and regularity ensures that each term has a unique least sort. Those that are generated by the sorting rules in Figure 1 are called proved terms. Note that a term \( t \) may have more than one sort \( s \) for which \( \Gamma \vdash t : s \) is a proved term. However there is always a unique least sort.

**Lemma 2.3 (Terms Have A Least Sort)** Suppose that \( \Gamma \) is a context and \( t \) a raw term for a given regular signature \( Th \). If there is any sort \( s \) for which \( \Gamma \vdash t : s \) is a proved term, then there is a least such sort, \( s \).

**Proof.** One uses rule induction. The proof is easy, though in the literature a key step is often omitted. By induction, for the rule \( F_n \), one easily uses regularity to obtain a sort, say \( s \), such that \( \Gamma \vdash f(t_1, \ldots, t_n) : s \). Now \( s \) is a candidate for the least sort of \( f(t_1, \ldots, t_n) \). Most authors state that such a sort \( s \) is least. This is true, but proving it so requires a separate (though trivial) rule induction.

We denote by \( t[u/x] \) the substitution of the raw term \( u \) for the variable \( x \) in \( t \), and by \( t[u/x] \) the simultaneous substitution of raw terms \( u \equiv u_1, \ldots, u_n \) for variables \( x \equiv x_1, \ldots, x_n \).

**Definition 2.4 (Inclusion Structure and FPI-category)** An inclusion structure \( 1 \) in a category \( C \) is specified by a subposet (subcategory) \( 1 \) of \( C \) such that

- for any two objects \( A \) and \( B \) of \( C \), the unique morphism \( A \to B \) in \( 1 \), if any, is a monic in \( C \);
- for any object \( A \) in \( C \), the identity \( id_A \) is in \( 1 \) (so \( 1 \) is a huff subcategory: it has the same objects as \( C \)).
Given an order-sorted signature $S_g$, an $S_g$-algebra $A$ is specified by

- an object $[s]_A$ in $C$ for each sort $s$ and object $[w]_A ≜ [s_1]_A × \cdots × [s_n]_A$ for each $w ≜ s_1, \ldots, s_n ∈ S^n$;
- morphisms $[f : w → s]_A : [w]_A → [s]_A$ for each $f : w → s$ and $k : s$; and
- a morphism $[s ≤ r]_A : [s]_A → [r]_A$ in $1$ for each $s ≤ r$ in $S$, where we set $[s ≤ s]_A ≜ id_{[s]}$ such that if the function symbol $f$ appears with more than one rank $f : w_1 → s$ and $f : w_2 → r$ in $S_g$ with $s_1, \ldots, s_a ≜ w_1 ≤ w_2 ≤ r_1, \ldots, r_a$, then the following diagram commutes:

\[
\begin{array}{ccc}
[s_1]_A × \cdots × [s_n]_A & \xrightarrow{[f : w_1 → s]_A} & [s]_A \\
[s_1 ≤ r_1]_A × \cdots × [s_n ≤ r_n]_A & \downarrow & [s ≤ r]_A \\
[r_1]_A × \cdots × [r_a]_A & \xrightarrow{[f : w_2 → r]_A} & [r]_A
\end{array}
\]

From now on, we drop the algebra subscript and the ranks of function symbols in the semantic brackets whenever they are clear by context.

**Definition 2.6 (Order-Sorted Homomorphism)** Let $S_g$ be an order-sorted signature and let $A$ and $B$ be $S_g$-algebras. An $S_g$-homomorphism $h : A → B$ is an $S$-sorted morphism $(h_s : [s]_A → [s]_B | s ∈ S)$ such that given $f : s_1, \ldots, s_a → s$, $k : s$, and $s ≤ r$ in $S_g$ the following diagrams commute:

\[
\begin{array}{ccc}
[s_1]_A × \cdots × [s_n]_A & \xrightarrow{[f]_A} & [s]_A \\
h_{s_1} × \cdots × h_{s_n} & \downarrow & h_s \\
[s_1]_B × \cdots × [s_n]_B & \xrightarrow{[f]_B} & [s]_B
\end{array}
\]
\[
\begin{array}{ccc}
1 & \xrightarrow{[k]_A} & [s]_A \\
h_s & \downarrow & h_s \\
[s]_A & \xrightarrow{[s ≤ r]_A} & [r]_A
\end{array}
\]
\[
\begin{array}{ccc}
[s]_B & \xrightarrow{[s ≤ r]_B} & [r]_B
\end{array}
\]

We define $h_w ≜ h_{s_1} × \cdots × h_{s_n}$ provided that $w ≜ s_1, \ldots, s_n$.

Given an order-sorted signature $S_g$, the class of all the order-sorted $S_g$-algebras and the class of all the order-sorted $S_g$-homomorphisms form a category denoted by $OSAlg(C, Σ)_S$.

If $Γ ⊢ t : s$ is a proved term in a regular signature $S_g$ and $Γ ≜ [x_1 : s_1, \ldots, x_n : s_n]$, any $S_g$-algebra $A$ induces a (unique) morphism from $[Γ]_A ≜ [s_1]_A × \cdots × [s_n]_A$ to $[s]_A$ in $C$ according to the inductive definition that appears in Figure 2. We denote such an arrow by $[Γ ⊢ t : s]_A$ and we refer to it as the semantics of $[Γ ⊢ t : s]$. Since terms can be assigned different types in one given context, we should consider whether the definition in Figure 2 is a sensible one. As such, we have the following lemma, where one sees that semantics of substitutions of terms is given as usual by morphism composition.

**Lemma 2.7 (Well-Defined Semantics)** Given a proved term $Γ ⊢ t : s$ and an algebra $A$:

- The semantic morphism $[Γ ⊢ t : s]_A$ is unique; that is, the assignment $ξ → [ξ]$ is a total function.
Let $\Gamma \vdash [x_1 : s_1', \ldots, x_n : s_n']$ be a context in (AcSub) and (AcSub).

\[
\begin{align*}
\Gamma \vdash t = t' : s & \quad \text{in } Ax \\
(\forall i \leq i \leq n) \quad \Gamma \vdash u_i : s_i' & \quad \text{(Ref)} \\
\Gamma' \vdash t[u/x] = t'[u/x] & \quad \text{(Sym)} \\
(\forall i \leq i \leq n) \quad \Gamma' \vdash u_i : s_i' & \quad \text{(Subsort)} \\
\Gamma \vdash t = t' : s & \quad \text{(AxSub)} \\
\Gamma \vdash t = t' & \quad \text{(Sym)} \\
\Gamma + t = t' & \quad \text{(Trans)} \\
\Gamma \vdash [x_1 : s_1', \ldots, x_n : s_n'] & \quad \text{(AxSub)}
\end{align*}
\]

Lemma 2.9 (Generalised Substitution) The following rule is admissible by a routine rule induction

\[
(\text{Sub}) \\
\begin{align*}
\Gamma \vdash t = t' : s \\
\Gamma' \vdash t[u/x] = t'[u/x] & \quad \text{(Sub)} \\
\Gamma \vdash u_i : s_i & \quad \text{(Subsort)} \\
\Gamma \vdash [x_1 : s_1', \ldots, x_n : s_n'] & \quad \text{(Subsort)}
\end{align*}
\]

Let $\Gamma \vdash [x_1 : s_1', \ldots, x_n : s_n']$ be a coherent signature. An equation if

\[
(\forall i \leq i \leq n) \Gamma' \vdash u_i : s_i' \\
(\forall i \leq i \leq n) \Gamma' \vdash t_i : s_i' \\
\Gamma' \vdash [x_1 : s_1', \ldots, x_n : s_n']
\]

Lemma 2.10 (Satisfaction is Well-Defined) As a consequence of Lemma 2.7, satisfaction is well-defined up to subsort-polyorphic equality, as follows: Suppose that we have a theorem $\Gamma \vdash t = t' : s$ satisfied in a model $\mathfrak{A}$. If $\Gamma \vdash t = t'$, $s$ is also a theorem, then it too is satisfied.

Proof. The existence of least sorts $s_i$ and $s_i'$, means that $s$ and $s'$ are connected, and so have a super-type $s'$. Thus each term has this type $s'$, and the result follows by using factorisation from Lemma 2.7 and the left-cancellation properties of monomorphisms. \qed

The category $\text{Pres}_{X_i, \ldots}((\mathcal{C}, 1), (\mathcal{D}, 0))$ is defined by having objects functors $F : \mathcal{C} \to \mathcal{D}$ such that finite
products are preserved and $F$ restricts to a functor $F^*_B: \emptyset \to J$ (that is, monics are also preserved). Suppose that we have a model $A$ in $\text{OSMod}([C,1])_{Th}$. Then there is a model $F.A$ in $\text{OSMod}([D,J])_{Th}$ that is, roughly speaking, defined by “taking the image of $Sg$ in $(C,J)$ induced by the model $A$, and applying $F$”. Equally one may “apply $F$ to homomorphisms of models” and this process (which is absolutely standard in categorical type theory/logic; see for example [6,16]) leads to a functor $A_{\text{Pred}}: \text{Pres}_{x\rightarrow y}((C,I),(D,J)) \to \text{OSMod}([D,J])_{Th}$. A **classifying category** $Cl(Th)$ for a theory $Th$ is an FPI-category such that there is an equivalence of categories

$$A_{\text{Pred}}: \text{Pres}_{x\rightarrow y}((C,I),(D,J)) \simeq \text{OSMod}([D,J])_{Th}$$

and thus models of $Th$ in $(D,J)$ correspond to such structure preserving functors with source $Cl(Th)$.

**Theorem 2.11 (Existence of Classifying Category)** There is an FPI-category $Cl(Th)$ constructed out of the syntax of $Th$, in which there is a **generic** model of $Th$ with the property that equality of morphisms corresponds to derivability of term equations. At an abstract level this notion is standard in categorical type theory/logic; see for example [6,16]. We feel that our concrete construction is simpler than that found in [20], and regard this as a small contribution. Such existence proofs are notoriously tricky to get completely correct, and there are notable errors in the literature. We use matching contexts and permutation invariance [7,29] to replace the usual substitutions that rename variables, and we think this makes our proofs simpler to state and more general, below, the metavariables $x, y, z$ etc, possibly subscripted, range over $\text{Var}$. Thus $\Gamma \models [x_1: s_1, \ldots, x_n: s_n]$ is a typical context as before, and we say that any such context matches $s \triangleq s_1, \ldots, s_n$.

**Proof.** Let $\text{Var}$ be a (countable) fixed set of variables $\{V_1, V_2, \ldots\}$. We call the context $\Gamma_s \triangleq \{V_1: s_1, \ldots, V_n: s_n\}$ the **primary** context for any sorts $s_1, \ldots, s_n$. In general, below, the metavariables $x, y, z$ etc, possibly subscripted, range over $\text{Var}$. Thus $\Gamma \models [x_1: s_1, \ldots, x_n: s_n]$ is a typical context as before, and we say that any such context matches $s \triangleq s_1, \ldots, s_n$.

First we define the objects $T(\Gamma)$ of $\text{FOSAlg}$. Of course $T(\Gamma)$ must be an $S$-sorted set, and the components are sets of equivalence classes

$$T(\Gamma)_s \triangleq \text{Term}(\Gamma)_s / \sim \quad \text{where} \quad \text{Term}(\Gamma)_s \triangleq \{t \mid \Gamma \vdash t: s \text{ in } Th\}$$

where we define the equivalence relation $t \sim t'$ just in case we can derive $\Gamma \vdash t = t': s$ in $Th$, and write $[t]$ for a typical equivalence class.

Let $\Gamma$ and $\Gamma'$ match $s_1, \ldots, s_n$ and $s'_1, \ldots, s'_m$ respectively. The morphisms $h : T(\Gamma) \to T(\Gamma')$ must be $S$-sorted functions $(h_s : T(\Gamma)_s \to T(\Gamma')_s \mid s \in S)$. These are specified by lists $h_s \triangleq \{t' \vdash t_1, \ldots, t_n \}$ where $t_i \in \text{Term}(\Gamma')_s$, and where

$$h_s([\bar{t}] \in T(\Gamma)_s) \triangleq [\bar{t}[t_1, \ldots, t_n/x_1, \ldots, x_m]] \in T(\Gamma')_s$$

It is easy to check this is well defined. Note that if

$$h \triangleq \{\Gamma \vdash t_1, \ldots, t_n\} : T(\Gamma) \to T(\Gamma')$$

and

$$h' \triangleq \{\Gamma' \vdash t'_1, \ldots, t'_m\} : T(\Gamma') \to T(\Gamma'')$$

then we have $h \circ h'$ defined by

$$\{\Gamma'' \vdash t_1[t'_1, \ldots, t'_m/x_1, \ldots, x_m], \ldots, t_n[t'_1, \ldots, t'_m/x_1, \ldots, x_m]\}$$

It is tedious but routine to verify that this gives rise to a category, relying crucially on the substitution rules for equation derivation. $\square$

**Theorem 2.12 (Soundness and Completeness)** Let $Th$ be an order-sorted theory. $\Gamma \vdash t = t': s$ is a theorem of $Th$ if and only if $\Gamma \vdash t = t': s$ is satisfied by every model of $Th$.

**Proof.** Soundness follows by rule induction for Figure 3. For completeness, suppose that $\Gamma \vdash t = t': s$ is satisfied in any model. Then in particular it is satisfied in the generic model $G$ in the classifying category $Cl(Th)$. Thus we have $[\Gamma \vdash t: s]_G = [\Gamma \vdash t': s]_G$ and so we have $(\Gamma \mid t) = (\Gamma \mid t')$ which holds precisely when $\Gamma \vdash t = t': s$ is a theorem. $\square$

We conclude this section with a new result, although it is motivated by analogous theorems [28]. The proof also makes use of matching contexts and permutations of variables.

**Theorem 2.13 (Relationship to Free Algebras)** There is an equivalence between FPI-categories $Cl(Th)$ and $(\text{FOSAlg}^{op}, J)$ where $\text{FOSAlg}$ is the category of free order-sorted algebras over finite sets of
variables, and order-sorted homomorphisms. Moreover the equivalence is given by an FPI-functor \( \Phi : \text{Cl}(\text{Th}) \simeq (\mathcal{FOSAlg}^{op}, J) \).

**Proof.** With a view to showing that \((\mathcal{FOSAlg}^{op}, J)\) is an FPI-category, we shall show that \(\mathcal{FOSAlg}\) is has finite coproducts, and then define \( J \). Given objects \( T(\Gamma) \) and \( T(\Gamma') \) then the binary coproduct object is given by \( T(\Delta \triangleleft \Gamma, \Gamma') \) where the sorts match \( \Gamma \) and \( \Gamma' \) respectively. The coproduct insertions are given by \( \{ \Delta \vdash v_1, \ldots, v_n \} \) and \( \{ \Delta \vdash v_{n+1}, \ldots, v_{n+m} \} \). Given morphisms

\[
\{ \Gamma'' \vdash t_1, \ldots, t_n \} : T(\Gamma) \rightarrow T(\Gamma'') \\
\{ \Gamma'' \vdash t'_1, \ldots, t'_m \} : T(\Gamma') \rightarrow T(\Gamma'')
\]

then the mediating morphism is \( \{ \Gamma'' \vdash t_1, \ldots, t_n, t'_1, \ldots, t'_m \} \). Note that \( T(\emptyset) \) is the initial object.

Suppose that \( s_i \leq r_i \) for each \( 1 \leq i \leq n \). Then there is an epic morphism \( i \triangleleft \{ \Gamma_s \vdash x_1, \ldots, x_n \} : T(\Gamma_r) \rightarrow T(\Gamma_s) \); it’s easy to verify that this is an epimorphism, and hence yields a monomorphism in \( \mathcal{FOSAlg}^{op} \). The luff subcategory \( \mathcal{J} \) has all of its morphisms the monomorphisms \( i^{op} \triangleleft \{ \Gamma_s \vdash x_1, \ldots, x_n \} : T(\Gamma_s) \rightarrow T(\Gamma_r) \). This is certainly an inclusion category.

Now we prove the equivalence. We define a functor \( \Phi : \text{Cl}(\text{Th}) \rightarrow (\mathcal{FOSAlg}^{op}, J) \) as follows. Given a morphism \( (\Gamma \mid t_1) \ldots (\Gamma \mid t_m) : s \rightarrow r \) then \( \Phi \) sends this to

\[
\{ \Gamma_s \vdash \pi t_1, \ldots, \pi t_m \} : T(\Gamma_r) \rightarrow T(\Gamma_s)
\]

where permutation \( \pi \) is specified by \( \pi : x_i \mapsto V_i \). We check this is well defined. Suppose that

\[
(\Gamma \mid t_1) \ldots (\Gamma \mid t_m) = (\Gamma' \mid t'_1) \ldots (\Gamma' \mid t'_m).
\]

We need to check that

\[
\{ \Gamma_s \vdash \pi t_1, \ldots, \pi t_m \} \triangleleft \Phi((\Gamma \mid t_1) \ldots (\Gamma \mid t_m)) = \Phi((\Gamma' \mid t'_1) \ldots (\Gamma \mid t'_m)) \triangleleft \{ \Gamma_s \vdash \pi' t'_1, \ldots, \pi' t'_m \}
\]

By definition we have \( \Gamma \vdash t_j = \rho t'_j : r_j \) where \( \rho \) is specified by \( \rho : x'_i \mapsto x_i \). We can deduce, using substitution rules for equations, that \( \pi \Gamma \vdash \pi t_j = \pi(\rho t'_j) : r_j \) and this is exactly \( \Gamma_s \vdash \pi t_j = \pi' t'_j : r_j \) as required, since \( \pi' = \pi \circ \rho \). We feel that the use of permutations, while equivalent to the use of simultaneous variable renamings by substitution, improves readability and more importantly simplifies calculations by making use of judgements that are permutation invariant.

(\( \Phi \) is essentially surjective): For any object \( s \) in \( \text{Cl}(\text{Th}) \) we have \( \Phi(s) = T(\Gamma_s) \). But one easily shows that for any \( T(\Gamma) \) where \( \Gamma \) matches \( s \), we have \( T(\Gamma) \cong T(\Gamma_s) \) where the inverse homomorphisms ‘swap variables’ \( x_i \) and \( V_i \).\( \Phi \) is faithful: Let

\[
\Phi((\Gamma \mid t_1) \ldots (\Gamma \mid t_m)) = \Phi((\Gamma' \mid t'_1) \ldots (\Gamma \mid t'_m)).
\]

We need to check that \( \Gamma \vdash t_j = \rho t'_j : r_j \). By the assumption we have

\[
\{ \Gamma_s \vdash \pi t_1, \ldots, \pi t_m \} = \{ \Gamma_s \vdash \pi' t'_1, \ldots, \pi' t'_m \}
\]

Hence \( \Gamma_s \vdash \pi t_j = \pi' t'_j : r_j \). Therefore we can deduce that \( \Gamma \vdash t_j = (\pi^{-1} \circ \pi') t'_j : r_j \); we are done since \( \pi^{-1} \circ \pi' = \rho \).\( \Phi \) is full: Let

\[
\{ \Gamma_s \vdash t_1, \ldots, t_m \} : \Phi(r) = T(\Gamma_r) \rightarrow \Phi(s) = T(\Gamma_s)
\]

Then \( \{ \Gamma_s \mid t_1, \ldots, t_m \} \) is the appropriate \( \text{Cl}(\text{Th}) \) morphism.

(\( \Phi \) is an object of \( \text{Pres}_{\mathcal{X}, \mathcal{J}}(\text{Cl}(\text{Th}), (\mathcal{FOSAlg}^{op}, J)) \)): We need to check that \( \Phi_\emptyset : \emptyset \rightarrow \mathcal{J} \) where \( \emptyset \) is the inclusion category of \( \text{Cl}(\text{Th}) \). Let \( s \leq r \). Note that \( \Phi((\Gamma \mid x_1) \ldots (\Gamma \mid x_n) : s \rightarrow r) \) is the monomorphism \( \{ \Gamma_s \vdash x_1, \ldots, x_n \} : \Gamma_r \rightarrow \Gamma_s \).

3 Multi-Language Equational Logic

3.1 Fundamentals of Multi-Languages

Throughout this section we often refer to a Running Example, introduced below and subsequently extended, to illustrate how the theory works in a concrete setting (see Section 3.3 for the outline of a more complex example).
Running Example. Our example is defined using the following order-sorted signatures:

- The signature $S_{g1}$ defines the symbols of a language for constructing simple mathematical expressions over natural numbers in Peano’s notation. Let the poset of sorts $(S_1, \leq_1)$ of $S_{g1}$ be a poset with a single sort $\mathbb{N}$ denoting the type of natural numbers, and let the operators be those in Figure 4a.

- Let $c \in \text{Char} \triangleq \{a, b, \ldots, z\}$ be the metavariable ranging over a finite set Char of characters. The signature $S_{g2}$ defines a language to build strings over Char. The set of sorts $S_2$ of $S_{g2}$ carries the sort $\text{str}$ for strings and the sort $\text{chr}$ for characters. The subset relation $\leq$ is the reflexive relation on $S_2$ plus $\text{chr} \leq \text{str}$ (i.e., characters are one-symbol strings), and the operator symbols in $S_{g2}$ appear in Figure 4b.

We model $S_{g1}$ and $S_{g2}$ by the order-sorted algebras $A_1$ and $A_2$ (see Figure 5) in $\langle \text{Set, Incl} \rangle$, the FPIcategory of sets with inclusion functions forming the inclusion structure. The symbol $c'$ in the definition of $\text{chr} \leq_2 \text{str}$ denotes the character that follows $c$ in Char (assuming the standard alphabetical order).

Remark 3.1 The forthcoming definitions and results gradually define and illustrate multi-languages, and give relationships between multi-languages and order-sorted languages. A multi-language signature 3.2 is specified as two order-sorted signatures (as in the Running Example) together with an interoperability relation between the two signatures. This determines the terms of the multi-language. Note that the relation is not a universal property of the underlying signatures; and also note a multi-language signature explicitly provides users with the original two language specifications. We show in 3.3 that there are nice notions of categories of signatures, both order-sorted and multi-language. We shall also see that we can exhibit a functor that maps a multi-language signature to an order-sorted signature (the associated signature 3.4), blending the two original signatures into one. After defining multi-language algebras 3.5 and homomorphisms 3.6, Theorem 3.7 will provide, via associated signatures, a clear semantic relationship between multi and order-sorted languages. In particular, it suggests how to give the definitions of (multi-language) terms, equations, and satisfaction using the associated signature. We can then reason equationally about interoperability of the two given languages. This takes us to Section 3.2 where we study equational reasoning in detail.

In order to define multi-language signatures we introduce some crucial notation. We denote by $+$ the disjoint union of two sets: the insertion morphisms that form a coproduct in the category of sets are injective functions, thus they have left inverses (and one has a model of disjoint union). In the following, if $S_1$ and $S_2$ are two sets of sorts and $s \in S_i$ with $i \triangleq 1, 2$, we write

$$s_i \quad \text{for the element} \quad t_i(s) \in S_1 + S_2 \quad \text{where} \quad t_i(s) \triangleq (s, i) \in S_1 \times \{1\} \cup S_2 \times \{2\}$$

Thus in relationships $s_i \times s_j'$ we have $s \in S_i$ and $s' \in S_j$. This is a very useful notation but perhaps requires care on first reading. Moreover, if $w \triangleq s_1, \ldots, s_n \in S^*_i$, then we write $w_i$ for $(s_1)_i, \ldots, (s_n)_i$. 

Buro, Crole, and Mastroeni
Definition 3.2 (Multi-Language Signature) A multi-language signature $SG \triangleq (Sg_1, Sg_2, \kappa)$ is

- a pair of order-sorted signatures $Sg_1$ and $Sg_2$ with posets of sorts $(S_1, \leq_1)$ and $(S_2, \leq_2)$, respectively; and
- a (binary) relation $\text{join} \kappa$ over $S_1 + S_2$ such that $s_i \kappa s_j'$ with $i, j \in \{1, 2\}$ and $i \neq j$.

The idea is that if $s_i \kappa s_j'$, and $\Gamma \vdash t: s$ is a proved term in one language, then $t$ can be used in place of a term $t'$ such that $\Gamma' \vdash t': s'$ in the other language: as in [21], “ML code can be used in place of Scheme code”. This is made precise in due course.

Definition 3.3 $\text{OSSg}$ is the category of order-sorted signatures with morphisms $h: Sg_1 \rightarrow Sg_2$ given by

- a monotone function $h: (S_1, \leq_1) \rightarrow (S_2, \leq_2)$ (where we will write $h(w) \triangleq h(s_1), \ldots, h(s_n)$ for $w \triangleq s_1, \ldots, s_n \in S_1^n$), and
- a mapping $h$ from the operators in $Sg_1$ to those in $Sg_2$ that preserves rank: given $k: s \in Sg_1$, then $h(k): h(s) \in Sg_2$; and given $f: w \rightarrow s$ in $Sg_1$, then $h(f): h(w) \rightarrow h(s)$ in $Sg_2$.

Moreover, we denote by $\text{MLSg}$ the category of multi-language signatures in which a morphism

$$H \triangleq (h_1, h_2): (Sg_1, Sg_2, \kappa) \rightarrow (Sg_1', Sg_2', \kappa')$$

is defined by two morphisms $h_1: Sg_1 \rightarrow Sg_1'$ and $h_2: Sg_2 \rightarrow Sg_2'$ in $\text{OSSg}$ such that they preserve the join relation, namely $s_i \kappa s_j'$ in $Sg_1, Sg_2, \kappa$ implies $(h_i(s)_i, \kappa')(h_j(s)'_j)$ in $(Sg_1', Sg_2', \kappa')$.

Definition 3.4 (Associated Signature) Let $SG \triangleq (Sg_1, Sg_2, \kappa)$ be a multi-language signature. The associated signature $\overrightarrow{SG}$ of $SG$ is the order-sorted signature defined as follows:

- the poset of sorts is given by $(S_1 + S_2, \leq)$, where $s_i \leq s_j$ if $i = j$ and $s_i \leq r$;
- if $f: w \rightarrow s$ is an operator in $Sg_2$, for some $i \triangleq 1, 2$, then $f_i: w_i \rightarrow s_i$ is a function symbol in $\overrightarrow{SG}$;
- if $k: s$ is a constant in $Sg_i$ for some $i \triangleq 1, 2$, then $k_i: s_i$ is a constant in $\overrightarrow{SG}$;
- and a conversion operator $\overset{s_i, s_j'}{\Rightarrow}: s_i \rightarrow s_j'$ for each constraint $s_i \kappa s_j'$.

The associated signature functor $(\overrightarrow{-}): \text{MLSg} \rightarrow \text{OSSg}$ maps each multi-language signature $SG \triangleq (Sg_1, Sg_2, \kappa)$ to its associated signature $\overrightarrow{SG}$, and each multi-language signature morphism $H: SG \rightarrow SG'$ to the order-sorted signature morphism $\overrightarrow{H}: \overrightarrow{SG} \rightarrow \overrightarrow{SG'}$ given by $\overrightarrow{H}(s_i) \triangleq (h_i(s)_i)$ for each $s \in S_i$ (hence $s_i \in S_1 + S_2$) and $\overrightarrow{H}(f_i) \triangleq (h_i(f))_i$ for each $f \in Sg_i$ (hence $f_i$ in $Sg_i$).

Running Example. $(\overrightarrow{-})$ embeds the multi-language signature $SG$ into $\text{OSSg}$, providing the order-sorted version $\overrightarrow{SG}$ of the multi-language. $\overrightarrow{SG}$ generates $SG$-terms (see Section 3.2) as well as hybrid multi-language terms involving conversion operators such as $[e: \text{int}, n: \text{nat}] \vdash 4 \ (\overrightarrow{\text{concat}}, \text{nat}, \text{int}): \text{nat}$. From now on, we use colours in the examples for disambiguating the left and the right inclusion in place of subscripts 1 and 2. Moreover, we use an infix notation whenever the operators lend themselves well to do so. That is, the previous term is represented by $[e: \text{int}, n: \text{nat}] \vdash \text{concat}, \text{nat}, \text{int} :: \text{nat}$. Such a functor outlines an embedding of multi-language signatures into order-sorted signatures, enabling us to see a multi-language as an ordinary language. Indeed, it is easy to see that $(\overrightarrow{-})$ is both injective on objects and a faithful functor.

Definition 3.5 (Multi-Language Algebra) Let $SG \triangleq (Sg_1, Sg_2, \kappa)$ be a multi-language signature. An $SG$-algebra $A$ in an FPI-category $(C, \emptyset)$ is given by

- a pair of order-sorted algebras $A_1$ and $A_2$ in $(C, \emptyset)$ over $Sg_1$ and $Sg_2$, respectively; and
- a boundary morphism $[s_i, s_j']_A: [s_i]_A \rightarrow [s_j']_A$ in $C$ for each constraint $s_i \kappa s_j'$.

An algebra sets out the meaning of a multi-language: The meaning of the underlying languages, and how terms of sort $s \in S_i$ can be interpreted as terms of sort $s' \in S_j$. Put differently, “boundary morphisms regulate the flow of values across $A_1$ and $A_2$” [22].

Definition 3.6 (Multi-Language Homomorphism) Let $SG \triangleq (Sg_1, Sg_2, \kappa)$ be a multi-language signature, and let $A$ and $B$ be two $SG$-algebras. An $SG$-homomorphism $h: A \rightarrow B$ is given by a pair of order-sorted homomorphisms $h_1: A_1 \rightarrow B_1$ and $h_2: A_2 \rightarrow B_2$ such that they commute with boundary functions, namely, if $s_i \kappa s_j'$, then the following diagram commutes:

9
Given a multi-language signature $SG$, the class of all $SG$-algebras and $SG$-homomorphisms form a category denoted by $\mathcal{MLAlg}(C, I)_SG$. We have a simple connection between this category and $\mathcal{OSAlg}(C, I)$, outlined in Theorem 3.7, after more of the Running Example.

**Running Example.** Suppose we are interested in a multi-language $SG \triangleq (Sg_1, Sg_2, \kappa)$ according to the following specifications:

- terms denoting natural numbers can be used in place of characters according to the function $\text{chr}: \mathbb{N} \to \text{Char}$; and
- terms denoting strings can be used in place of natural numbers according to the function $\text{len}: \text{Char}^* \to \mathbb{N}$, namely the length of the string.

In order to get such a multi-language, we provide (1) the join relation $\sqsupset$ on $S_1 + S_2$ and (2) a boundary morphism $[s_i \times s'_j]_A : [s'_j]_A \to [s'_j]_A$, for each constraint $s_i \times s'_j$ introduced by $\kappa$:

- $\sqsupset s_1 \times \sqsupset s_2$ and $\sqsupset s_1 \times \sqsupset s_2$ with boundaries $[\sqsupset s_1 \times \sqsupset s_2]_A(n) \triangleq [\sqsupset s_1 \times \sqsupset s_2]_A(n) \triangleq \text{chr}(n)$; and
- $\sqsupset s_1 \times \sqsupset s_2$ and $\sqsupset s_1 \times \sqsupset s_2$ with boundaries $[\sqsupset s_1 \times \sqsupset s_2]_A(c) \triangleq \text{len}(c) = 1$ and $[\sqsupset s_1 \times \sqsupset s_2]_A(s) \triangleq \text{len}(s)$.

The next theorem yields a formal correspondence between multi-languages and order-sorted languages: We can make a multi-language signature $SG$ into an order-sorted one by applying the functor $[\_]$, and thus blending the underlying languages. Nevertheless, we do not lose any semantical information if we consider the category of algebras over $SG$ and $SG$.

**Theorem 3.7** There is a natural isomorphism between the category of multi-language algebras over $SG$ and the category of order-sorted algebras over the associated signature $SG$.

\[ \eta: \mathcal{MLAlg}(C, I) \to \mathcal{OSAlg}(C, I) \circ [\_] \quad \text{inducing} \quad \mathcal{MLAlg}(C, I)_SG \cong \mathcal{OSAlg}(C, I)_SG \]

where there are functors $\mathcal{MLAlg}(C, I)_1: \mathcal{MLSg} \to \mathcal{Cat}^{\text{op}}$ and $\mathcal{OSAlg}(C, I)_1: \mathcal{OSSG} \to \mathcal{Cat}^{\text{op}}$ that map signatures to their category of algebras in $(C, I)$.

**Proof.** The functors are defined on objects by $\mathcal{MLAlg}(C, I)(SG) \triangleq \mathcal{MLAlg}(C, I)_SG$ and $\mathcal{OSAlg}(C, I)(SG) \triangleq \mathcal{OSAlg}(C, I)_SG$. Now, let $SG \triangleq (Sg_1, Sg_2, \kappa)$ and $SG' \triangleq (Sg'_1, Sg'_2, \kappa')$, and let $H \triangleq (h_1, h_2): SG \to SG'$ be a signature morphism between them. We shall define a functor $H^*: \mathcal{MLAlg}(C, I)_SG \to \mathcal{MLAlg}(C, I)_SG'$ and let $\mathcal{MLAlg}(C, I)(H) \triangleq H^*$. Let $A$ be an $SG'$-algebra in $\mathcal{MLAlg}(C, I)_SG'$. Then, the multi-language $SG$-algebra $H^*A$ is defined as follows:

- We define its order-sorted components $(H^*A)_1$ and $(H^*A)_2$. Let $i \triangleq 1, 2$:
  - the interpretation of sorts is given by $[s_i]_{(H^*A)} \triangleq [h_i(s)]_A$, for each $s \in S_i$;
  - the constant symbol $k$: $s \to s$ in $Sg_i$, we define $[f: w \to s_i]_{(H^*A)} \triangleq [h_i(f): h_i(w) \to h_i(s)]_A$;
  - the constant symbol $k$: $s \leq r$ in $Sg_i$, is interpreted by letting $[k]_{(H^*A)} \triangleq [h_i(k)]_A$; and
  - $[s \leq r]_{(H^*A)} \triangleq [h_i(s) \leq h_i(r)]_A$ for each sort constraint $s \leq r$ in $Sg_i$.

The fact that $(H^*A)_i$ is a proper order-sorted $Sg_i$-algebra is ensured by the properties of the (multi-language) signature morphism $H$.

- Boundary morphisms are defined by $[s_i \times s'_j]_{H^*A} \triangleq [h_i(s) \times h'_j(s')]_A$ for each constraint $s_i \times s'_j$ in $SG$.

In order to define the action of $H^*$ on homomorphisms, suppose that $g: A \to B$ is a multi-language $SG'$-homomorphism in $\mathcal{MLAlg}(C, I)_SG'$. Then, $(H^*g)_i: (H^*A)_i \to (H^*B)_i$ is defined by the $S_i$-sorted morphisms

- $((H^*g)_i)_s \triangleq (g_i)_{h_i(s)}$, which is well-defined since $g: A \to B$ is an order-sorted $Sg'_i$-homomorphism and $[h_i(s)]_A = [s]_{(H^*A)_i}$.

The commutativity of the diagram in Definition 3.6 is given by a tedious but simple diagram chase.
We next define a functor $h^* : \mathcal{O}S\mathcal{A}l\mathcal{G}(C, \varnothing)_S \to \mathcal{O}S\mathcal{A}l\mathcal{G}(\{\varnothing\}, 1)_S$, and set $\mathcal{O}S\mathcal{A}l\mathcal{G}(C, \varnothing)(h) \triangleq h^*$, where $h : S\varnothing_1 \to S\varnothing_2$ is an order-sorted signature morphism. This is similar to the definition of $H^*$ above. First pick any (order-sorted) $S\varnothing_2$-algebra $A$. We need to define the order-sorted $S\varnothing_1$-algebra $h^*A$. We define

- objects $\{s \in S\varnothing_1\} h^*A \triangleq \{\|h(s)\|_A\}$ in $C$ and hence $\|w\|_{h^*A} \triangleq \{h(s_1))\|A \times \cdots \times \{h(s_n))\|A$ for $w \triangleq s_1, \ldots, s_n \in S\varnothing_1$;
- morphisms $[f] : w \to s \in S\varnothing_2^I h^*A \triangleq \{h(f) : h(w) \to h(s) \in S\varnothing_2^A\}$: $\|w\|_{h^*A} \to \{\|s\|_{h^*A}$ and morphisms $\{k\in S\varnothing_2\} h^*A \triangleq \{k\}_{A} : 1 \to \{\|s\|_{h^*A}$;
- a morphism $\{s \leq r \in S\varnothing_1\} h^*A \triangleq \{h(s) \leq h(r)\} A$ : $\{\|s\|_{h^*A} \to \{\|r\|_{h^*A}$ in $0$.

We omit the verification that semantics of operators commutes with the semantics of subsorting, although this is essentially immediate since $A$ is an $S\varnothing_2$-algebra. Now let $g : A \to B$ be an order-sorted $S\varnothing_2$-homomorphism. We define the $S\varnothing_1$-homomorphism $h^*(g) : h^*A \to h^*B$ by setting the components to be $(h^*(g))_{s \in S\varnothing_1} \triangleq g_{\|h(s)\|_A} : \{\|h(s)\|_A \to \{\|h(s)\|_B$.

Now we define the natural transformation $\eta$ by specifying the components $\eta_{SG} : \mathcal{M}L\mathcal{A}l\mathcal{G}(C, \varnothing)_S \to \mathcal{O}S\mathcal{A}l\mathcal{G}(\{\varnothing\}, 0)_{SG}$. Pick any $SG$-homomorphism $h : A \to B$. First we define the (order-sorted) $SG$-algebra $\eta_{SG}A$ by setting

- $\{s_i \in S\varnothing_1 + S\varnothing_2\} \eta_{SG}A \triangleq \{\|s\|_A\}$ in $C$ and hence $\|w\|_{\eta_{SG}A} \triangleq \{[s_i]_{A} \times \cdots \times [s_n]_{A}$; for each $w \triangleq s_1, \ldots, s_n \in (S\varnothing_1 + S\varnothing_2)^n$;
- morphisms $\{f : w \to s \in \eta_{SG}A} \triangleq \{f : w \to s\} \|_{A}$, and $\{k\}_{\eta_{SG}A} \triangleq \{k\}_{A} : 1 \to \{\|s\|_{A}$; and
- $\{r \leq s_{i,j} \in \eta_{SG}A} \triangleq \{s \leq r\}_{\|_{A}}$ ; $\{\|s\|_{A} \to \{\|r\|_{A}$ in $0$ for each $s_i \leq r_j$ in $S\varnothing_1 + S\varnothing_2$ (don’t forget that $i = j$).

Then we need to show that the diagram below commutes.

$$\begin{align*}
\text{MLAlg}(C, \varnothing)_S \xrightarrow{\eta_{SG}} \text{OSAlg}(C, \varnothing)_S^SG \\
\text{MLAlg}(C, \varnothing)_S \xrightarrow{\eta_{SG}} \text{OSAlg}(C, \varnothing)_S^SG
\end{align*}$$

Pick any morphism $g : A \to B$ in $\text{MLAlg}(C, \varnothing)_S$. First we need to show that $\eta_{SG}(H^*A) = \overline{T}^*(\eta_{SG}A)$. Let us check only that these $SG$-algebras provide equal meaning to sorts $s_i \in S\varnothing_1 + S\varnothing_2$ where we have $(h_i(s))_{i} \in S\varnothing_1 + S\varnothing_2$.

$$\{s_i\}_{\eta_{SG}(H^*A)} = \{\|s\|_{\{H^*A\}_i} = \{[h_i(s)]_{i} \in \{[h_i(s)]_{i} \in S\varnothing_1 + S\varnothing_2\}_{\eta_{SG}A} = \{\overline{T}(s_i)\}_{\eta_{SG}A} = \{s_i\}_{\overline{T}^*(\eta_{SG}A)}$$
Now $g$ furnishes us with $SG'$-homomorphisms $g_i: A_i \to B_i$, and we need to show that

$$\eta_{SG}(H^*g) = \overline{H^*}(\eta_{SG'}g): \llbracket h_i(s) \in S_i \rrbracket_{A_i} \to \llbracket h_i(s) \in S_i \rrbracket_{B_i}$$

is an equality of $SG$-homomorphisms. But this follows from the following calculation on components of $S_1 + S_2$-sorted morphisms

$$(\eta_{SG}(H^*g))_s = ((H^*g)_s)_s = (g_i)h_i(s) = (\eta_{SG'}g)(h_i(s)), = (\eta_{SG'}g)(\overline{H}(s)) = (\overline{H^*}(\eta_{SG'}g))_s,$$

and the proof is completed. \hfill $\square$

**Running Example.** The multi-language semantics of the term introduced in the previous example is given by the algebra $\eta_{SG}A$ which leads to $\llbracket [c: \text{Char}, n: \text{Nat}] \vdash c \cdot n \vdash [c: \text{Char}, n: \text{Nat}] \eta_{SG}A = (c, n) \mapsto n + 1: \text{Char} \times \text{Nat} \to \text{Nat}$. 

### 3.2 Equational Reasoning in a Multi-Language Context

In this section we define multi-language proved terms, and give them a semantics. Then we define multi-language equations and semantic satisfaction. From this we can define theories and models, and hence prove soundness and completeness.

Let $SG \triangleq (Sg_1, Sg_2, \times)$ be a multi-language signature. A (multi-language) proved term $\Gamma \vdash t: s_i$ is a proved term over the associated signature $SG$. It follows that if $\Gamma \vdash t: s$ is a proved term over $Sg_1$, then $\Gamma \vdash t: s$ is a proved term in $SG$, where $\Gamma \vdash t: s \triangleq \Gamma \vdash t: s$ and

- $s \triangleq s_i$ for each $s \in S_i$; and $\Gamma \vdash [x_1; \ldots; x_n; s]$ for each context $\Gamma \triangleq [x_1; \ldots; x_n; s]$ over $Sg_i$;
- $t$ is recursively defined over the syntax of raw terms generated by $Sg_i$; $\mathcal{A} \triangleq x; \mathcal{B} \triangleq k$; and $f(t_1, \ldots, t_o) \triangleq f_i(t_1, \ldots, t_o)$.

Due to the injectivity of this construction, we shall refer to it as the inclusion of an order-sorted term into the multi-language, and we informally say that a multi-language “contains” the underlying languages. Furthermore, the definition of multi-language terms also includes hybrid terms that are not the result of the inclusion of an order-sorted term but which are constructed using the conversion operators in the associated signature.

Given a multi-language $SG$-algebra $A$, the categorical semantics of a (multi-language) term $\Gamma \vdash t: s_i$ is the order-sorted semantics of $\Gamma \vdash t: s_i$ induced by $\eta_{SG}A$, namely $\llbracket \Gamma \vdash t: s_i \rrbracket_A \triangleq \llbracket \Gamma \vdash t: s_i \rrbracket_{\eta_{SG}A}$. As expected, a multi-language preserves the semantics of the underlying terms:

**Proposition 3.8** Let $A$ be a multi-language $SG$-algebra over $SG \triangleq (Sg_1, Sg_2, \times)$. If $\Gamma \vdash t: s$ is a proved term over $Sg_2$, then $\llbracket \Gamma \vdash t: s \rrbracket_A = \llbracket \Gamma \vdash t: s \rrbracket_{\eta_{SG}A}$. 

Regularity and coherence for a multi-language signature $SG \triangleq (Sg_1, Sg_2, \times)$ are defined with respect to its associated signature. That is, $SG$ is regular (resp., coherent) if $Sg$ is regular (resp., coherent). It is immediate that $SG$ is regular (resp., coherent) if and only if $Sg_1$ and $Sg_2$ are regular (resp., coherent).

**Definition 3.9 (Multi-Language Equation and Satisfaction)** Let $SG$ be a coherent multi-language signature. A (conditional) equation for $SG$ is an order-sorted (conditional) equation over $SG$. A multi-language algebra $A$ satisfies any such (conditional) equation if the (conditional) equation is satisfied by $\eta_{SG}A$.

An immediate consequence of Proposition 3.8 is that every $Sg$-equation satisfied by $A$, is also satisfied by the multi-language algebra $A$ (in its inclusion form provided by the mapping $(-)$).

A multi-language theory $TH \triangleq (SG, AX)$ is a pair of a multi-language signature $SG$ and a set of (conditional) multi-language equations $AX$ over $SG$, namely the axioms of the theory. The theorems of $TH$ are the equations $\Gamma \vdash t = t'$, $s_i$ derivable from $(SG, AX)$. A multi-language $SG$-algebra that satisfies all the axioms in $AX$ is said a model of $TH$, and $\text{MLAlg}(C, 1)_{TH}$ denotes the full subcategory of models of $\text{MLAlg}(C, 1)_{TH}$. We now introduce the categories of theories in order to define the associated theory of a multi-language theory. From now on, when we write order-sorted theories $Th_1$, $Th_2$, $Th$, $Th'$, etc., we assume they are defined as $Th_1 \triangleq (Sg_1, AX_1)$, $Th_2 \triangleq (Sg_2, AX_2)$, $Th \triangleq (Sg, AX)$, $Th' \triangleq (Sg', AX')$, etc., respectively.

**Running Example.** Let $Th_1 \triangleq (Sg_1, AX_1)$ and $Th_2 \triangleq (Sg_2, AX_2)$ be the order-sorted theories over $Sg_1$ and $Sg_2$ axiomatized by the equations provided in Figure 6. We can generate from $AX_1$ and $AX_2$ a set $AX$ of multi-language equations by applying $(-)$ to each equation. For instance, $(eq_{1,1}) \triangleq [n: \text{Nat}] \vdash 0 + n = n: \text{Nat}$.
becomes \((eq_{1,1}) \triangleq [n: \text{nat}] \vdash 0 + n = n: \text{nat}\). Note that a substantial change occurs when mapping an order-sorted equation to a multi-language one. Consider again \((eq_{1,1})\). A substitution in the order-sorted world can only plug \(t\), where \(\Gamma \vdash t: \text{nat} \) in \(SG_1\), into the variable \(n: \text{nat}\). However, a multi-language substitution can substitute any \(t'\), where \(\Gamma' \vdash t': \text{nat} \) in \(SG\), for \(n: \text{nat}\) in the lifted equation \((eq_{1,1})\)—including, crucially, the possibility that \(t'\) is a hybrid multi-language term.

The behaviour of boundary morphisms can be axiomatized by adding the following equations to \(AX\):

\[
\begin{align*}
(EQ_1) & \quad \vdash \text{eq}_{\text{chf}} : (0) = a: \text{chr} \\
(EQ_2) & \quad \vdash \text{eq}_{\text{str}} : (0) = a: \text{str} \\
(EQ_3) & \quad [c: \text{chr}] \vdash \text{eq}_{\text{chf}, \text{rel}} (c) = s(0) : \text{nat} \\
(EQ_4) & \quad [c: \text{chr}] \vdash \text{eq}_{\text{str}, \text{rel}} (c) = s(0) : \text{nat} \\
(EQ_5) & \quad [n: \text{nat}] \vdash \text{eq}_{\text{chf}} : \text{next}(\text{eq}_{\text{chf}}(n)) = \text{chf} \\
(EQ_6) & \quad [n: \text{nat}] \vdash \text{eq}_{\text{str}} : \text{next}(\text{eq}_{\text{str}}(n)) = \text{str} \\
(EQ_7) & \quad [s: \text{str}, v: \text{str}] \vdash \text{eq}_{\text{str}, \text{rel}}(s + v) = \text{eq}_{\text{str}, \text{rel}}(s) \leftarrow \text{eq}_{\text{str}, \text{rel}}(v): \text{nat}
\end{align*}
\]

**Definition 3.10** Let \(OStH\) be the category of order-sorted theories whose morphisms \(h: Th_1 \rightarrow Th_2\) are signature morphisms \(h: SG_1 \rightarrow SG_2\) in \(OSSg\) that preserve theorems, that is, if \(\Gamma \vdash t = t'\); \(s_i\) is a theorem of \(Th_1\) with \(\Gamma \triangleq [x_1:s_1, \ldots, x_n: s_n]\), then \(\Gamma' \vdash h(t) = h(t')\); \(h(s_i)\) is a theorem of \(Th_2\), where \(\Gamma' \triangleq [x_1:h(s_1), \ldots, x_n: h(s_n)]\) and \(h(t)\) and \(h(t')\) are inductively defined over the syntax according to the action of \(h\) on function symbols and constants.

The category of multi-language theories is denoted by \(MLTh\) and a theory morphism \(H: (SG_1, AX_1) \rightarrow (SG_2, AX_2)\) is a signature morphism \(H: SG_1 \rightarrow SG_2\) in \(MLSg\) such that if \(\Gamma \vdash t = t'\); \(s_i\) is a theorem of \((SG_1, AX_1)\) with \(\Gamma \triangleq [x_1:s_1, \ldots, x_n: s_n]\), then \(\Gamma' \vdash \overline{H}(t) = \overline{H}(t')\); \(\overline{H}(s_i)\) is a theorem of \((SG_2, AX_2)\), where \(\Gamma' \triangleq [x_1:\overline{H}(s_1), \ldots, x_n: \overline{H}(s_n)]\).

Functors \(MALg(C, \iota)\) and \(OSAlg(C, \iota)\) can be easily extended to \(MLMod(C, \iota)\) and \(OSMod(C, \iota)\) respectively, such that they associate to each signature its corresponding category of models. Then, \((-)\): \(MLTh \rightarrow OStH\) is defined by \(\overline{TH} \triangleq (SG, AX)\) on objects and by \(\overline{TH}\) on morphisms \(H: TH_1 \rightarrow TH_2\).

**Proposition 3.11** \(MLMod(C, \iota)\) and \(OSMod(C, \iota)\) are isomorphic functors. Let \(\eta\) be the natural isomorphism between them and \(TH\) a multi-language theory. Then, \(\eta\overline{TH}\) is the isomorphism between categories \(MLMod(C, \iota)\overline{TTH}\) and \(OSMod(C, \iota)\overline{TTH}\).

**Theorem 3.12** (Soundness and Completeness) Let \(TH\) be a multi-language theory. \(\Gamma \vdash t = t'\); \(s\) is a theorem of \(TH\) if and only if \(\Gamma' \vdash t = t'\); \(s\) is satisfied by every model of \(TH\).

### 3.3 An Extended Example

The Running Example has the sole purpose of illustrating our theory in an elementary way: we are very much aware of its limitations. Here we give a taste of a more realistic example. For space reasons, we can convey only the main ideas: full details are in an extended version of the paper citearxiv-version-of-this-paper.

We define a new multi-language by blending a simple functional core with a minimal imperative language. The former is of course suited to writing programs that are easier to reason about, whereas the latter provides a more straightforward procedural and low-level approach to software development. We formalise the simply-typed lambda calculus and a simple imperative language as two equational theories, and we blend them together in order to provide the gist of an interoperability between the functional and imperative paradigms. More complex examples can be built along the lines of the one presented here.

We assume \(\text{Imp}\) and \(\lambda^\iota\) to be the signatures of a small imperative language and the simply-typed lambda-calculus, respectively. In the following, we use colours blue for denoting \(\text{Imp}\) code and red for \(\lambda^\iota\)-terms. The
interoperability we wish to provide should allow the use of $\lambda^-$-terms as $\Imp$-expressions and vice versa. For instance, we would like to write multi-language programs such as $x = (\lambda y: \text{int} . y + y)1$, which encodes the assignment to the variable $x$ of the value obtained by applying the $\lambda^-$-function $(\lambda y: \text{int} . y + y)$ to the $\Imp$-numeral $1$. Although minimal, its interpretation requires several applications of the boundary functions: First, we need to compute the result of the function application, which in turn needs the evaluation of $1$ as a $\lambda^-$-term. Then, the resulting term has to be converted back to an $\Imp$-numeral in order to be assigned to $x$.

The multi-language signature $\lambda^- \Imp$ providing the desired interoperability is given by coupling the signatures $\Imp$ and $\lambda^-$ with the join relation specified by $e \times \text{exp}$ and $\text{exp} \times e$, where $e$ is the sort of $\Imp$-expressions and $\text{exp}$ the sort of $\lambda^-$-terms. The semantics of the generated multi-language programs is obtained by introducing a boundary function for each $\times$-constraint. For instance, given a standard denotational semantics for both the underlying languages, the boundary function $[e \times \text{exp}]$ can provide each $\Imp$-expression with a $\lambda^-$-meaning in the following way: Let $e$ be the semantics of such an $\Imp$-expression. We can first transform a $\lambda^-$-environment to an $\Imp$-environment, run $e$ on its conversion, and then move the resulting $\Imp$-values to suitable $\lambda^-$-values.

The equational axiomatization of such a boundary function can be specified by the following multi-language equations: $\text{(1)} \iff_{e, \text{exp}} (i) = i$ and $\text{(2)} \iff_{e, \text{exp}} (x) = x$. The first equation allows $\lambda^-$-integers to be converted to $\Imp$-numerals of the same form. In more realistic examples, the conversion of values across languages should take into consideration different machine representations (for instance, if the $\lambda^-$ language does not admit an explicit representation of integers, we may convert the integer $i$ to its corresponding Church-numeral).

Equation (2) provides a match between $\Imp$ and $\lambda^-$ variables with the same name. This enables a natural way for moving stored values across the two languages. For instance, the multi-language program $x = (\lambda y: \text{int} . y + y)z$ acts in the same way of the previously described one but applying the $\lambda^-$-function to the value stored in the $\Imp$-variable $z$.

On the other hand, the boundary function $[\text{exp} \times e]$ works in a dual manner for providing $\lambda^-$-terms with an $\Imp$-meaning. Given all these specifications, the equational logic provides the following chain of equalities:

$$x = (\lambda y: \text{int} . y + y)1 \iff x = 1 + 1 \iff x = 2 \iff x = 2$$

### 4 Further Multi-Language Constructions

Buro and Mastroeni [3] provides three different multi-language constructions based on boundary morphism properties (although in their work, morphisms are only set-theoretic functions). In Section 3, we studied a categorical equational logic for the simplest construction. Here we briefly discuss the other two, each a refinement of the first.

The first refinement of multi-language signatures is accomplished by allowing all conversion operators $\rightarrow_{s_i, s'_j}: s_i \rightarrow s'_j$ in the associated signature to be replaced by subsort polymorphic operators $\rightarrow: s_i \rightarrow s'_j$ that do not carry any sort information. One can check that any associated signature $SG$ defined in this way remains an order-sorted signature if and only if the following additional constraint holds for $SG$:

$$s_i \times s'_j, r_i \times r'_j, \text{ and } s_i \leq r_i \text{ imply } s'_j \leq r'_j$$

Multi-language algebras are then restricted by the following monotonicity requirement:

$$s_i \times s'_j, r_i \times r'_j, \text{ and } s_i \leq r_i \text{ imply } [s'_j \leq r'_j]_A \circ [s_i \times s'_j]_A = [r_i \times r'_j]_A \circ [s'_j \leq r'_j]_A$$

This new multi-language construction, we can prove the following version of Theorem 3.7:

**Theorem 4.1** Assume (1) and (2) for multi-language signatures and algebras, respectively. There is a natural isomorphism $\eta: \text{MLAlg}(C, \mid) \cong \text{OSAlg}(C, \mid)_{SG} \cong \text{OSAlg}(C, \mid)^{\text{op}}$, where there are functors $\text{MLAlg}(C, \mid): \text{MLAlg} \rightarrow \text{Cat}^{\text{op}}$ and $\text{OSAlg}(C, \mid): \text{OSAlg} \rightarrow \text{Cat}^{\text{op}}$ that map signatures to their category of algebras in $(C, \mid)$.

**Proof.** The proof is almost identical to the proof of Theorem 3.7. That each $\eta_{SG}A$ is a proper order-sorted algebra boils down to the fact that each $[\rightarrow: s_i \rightarrow s'_j]_{SG}A$ commutes with the desired morphisms in $\mid$; but this commutativity follows immediately from (2).

The second refinement of multi-language signatures aims to achieve a multi-language construction which consists only of the union of the underlying languages, that is no conversion operator is added to the associated signature and single-language operators are not tagged. Such a construction is particularly useful when modeling the extension of a language rather than the union of two already existing languages.
The notion of multi-language signature is refined by assuming that
- \((S_1 + S_2, \kappa)\) is a poset; and
- \(f : w \to s\) in \(S_{g_1}\) and \(f : w' \to s'\) in \(S_{g_j}\) with \(w_i \kappa w'_j\), then \(s_i \kappa s'_j\).

and the associated signature \(\mathcal{S}\mathcal{G}\) is defined as follows:
- the poset of sorts is given by \((S_1 + S_2, \leq)\), where \(s_i \leq r_j\) if \(i = j\) and \(s \leq r\) or \(i \neq j\) and \(s_i \kappa r_j\);
- if \(f : w \to s\) is a function symbol in \(S_{g_i}\), then \(f : w_1 \to s_i\) is a function symbol in \(\mathcal{S}\mathcal{G}\), and similarly for constants.

Multi-language algebras now force boundary morphisms to act as subset morphisms. This means that if the function symbol \(f\) appears with more than one rank \(f : w \to s\) and \(f : w' \to r\) in \(S_{g_i}\), \(S_{g_j}\), etc., respectively, and \(s_i \kappa r_j\), then the following diagram commutes:

\[
\begin{array}{ccc}
\left[s_1\right]_{A_i} \times \cdots \times \left[s_n\right]_{A_i} & \xrightarrow{[f : w_1 \to s_1]_{A_i}} & \left[s\right]_{A_i} \\
\left[(s_1)_{i} \times (r_1)_j\right]_{A_i} \times \cdots \times \left[(s_n)_{i} \times (r_n)_j\right]_{A_j} & \xrightarrow{[f : w_2 \to r]_{A_j}} & \left[r\right]_{A_j}
\end{array}
\]

**Theorem 4.2** Assume these new hypotheses for multi-language signatures and algebras, respectively. There is a natural isomorphism \(\eta : \text{MLAlg}(C, \mathcal{L}) \Rightarrow \text{OSAlg}(\mathcal{L}, \mathcal{B})\) inducing \(\text{MLAlg}(C, \mathcal{L})_{\mathcal{S}\mathcal{G}} \cong \text{OSAlg}(\mathcal{L}, \mathcal{B})_{\mathcal{S}\mathcal{G}}\), where (as before) there are functors \(\text{MLAlg}(C, \mathcal{L}) : \text{MLAlg}(\mathcal{L}) \to \text{Cat}^{\mathcal{S}}\) and \(\text{OSAlg}(\mathcal{L}, \mathcal{B}) : \text{OSAlg}(\mathcal{L}, \mathcal{B}) \to \text{Cat}^{\mathcal{S}}\).

**Conclusions and Future Research**

Equational logic is a simple fragment of first-order logic with several applications to computer science [12,34,17]. In this paper, we have addressed the problem of equational deduction in a multi-language context. We have lifted the order-sorted equational logic of [13] to the algebraic framework of multi-languages introduced by [3], and we have proved the soundness and the completeness of the resulting deduction system. The main benefit of the theoretical development in this paper is a solid mathematical foundation for reasoning about equalities in a multi-language context.

Among all the applications, one motivation for extending the theory of equational logic to a multi-language context resides in the possibility of providing operational semantics to multi-languages, in a similar way to [11]. In future work, we plan to investigate this in the context of rewriting logic [31], where axioms might be partitioned into a set \(R\) of rewriting rules and a set \(E\) of equations in order to perform rewriting modulo \(E\).

We know that there is considerable practical interest in understanding how real languages and systems may be integrated to exploit advantages of each individual system. To make real progress, we believe that practical advances need to be made in synchrony with theoretical developments, with each approach supporting and informing the other. To this end, we are pursuing practical developments of the work presented in this paper. As a side note, we have also begun to look at the implementation of our examples within Maude [5].

We deduce unconditional equations but allow conditional axioms. This approach has merit from the point of view of practical specifications, and reasoning about them. That said, one could be rather more expressive if one allows conditional equations as primary judgements of a deduction system. In such a case, the semantics of judgements could be given in an internal manner by making use of categories with equalisers [19]. We are currently working on such a system, with a view to giving a sound and complete semantics, and the results will appear in a future paper. There are interesting questions concerning the appropriate category theory; and the answers will have connections to work such as [25]. And further, since equational theories give rise to free algebra monads [30], further studies should investigate the possibility of extending/generalizing the results in this paper to the notion of monad [23]. Here, however, our intention has been to provide an account that is very general (categorical) but not so abstract that applications become obfuscated.

Finally, we wish to note that the approach presented in this paper generalises to the combination of an arbitrary number \(n\) of languages by recursively combining the (associated theory of the) first \(\text{Th}_1, \ldots, \text{Th}_{n-1}\) theories with \(\text{Th}_n\). Such a modularity property strengthens the framework both from a theoretical and practical perspective, enabling the construction of complex theories on the basis of more elementary ones.
References


