Modular construction of multi-sorted free extensions (short paper)

Guillaume Allais
guillaume.allais@ens-lyon.org
University of St. Andrews
St. Andrews, Fife, Scotland, UK

Nathan Corbyn
nathan.corbyn@cs.ox.ac.uk
University of Oxford
Oxford, England, UK

Ohad Kammar
ohad.kammar@ed.ac.uk
University of Edinburgh
Edinburgh, Scotland, UK

Nachiappan Valliappan
nachivpn@gmail.com
Chalmers University
Gothenburg, Sweden

Sam Lindley
Sam.Lindley@ed.ac.uk
University of Edinburgh
Edinburgh, Scotland, UK

Jeremy Yallop
jeremy.yallop@cl.cam.ac.uk
University of Cambridge
Cambridge, England, UK

Abstract

The free extension (frex) offers a uniform theoretical foundation for the collection of partial-evaluation techniques known as partially-static data structures. In the frex methodology, proposed by Yallop et al. (2018), to define the partially static representation of a concrete semantic domain, we: (1) identify an algebraic signature for expressions in this semantic domain; (2) identify a collection of semantic equations with respect to which we wish to optimise expressions in this domain; and (3) design a data-structure, the partially-static representation, satisfying the specification given by the universal property of the free extension of the concrete semantic domain with a set of variables. Since universal properties are given up-to a unique canonical isomorphism, such specifications allow us to formulate the way in which different partially-static representations compare or differ. This uniform identification of the partially-static representations allows Yallop et al. to treat them uniformly in a single library. We report about ongoing work concerning using existing free extensions (frexes) for component theories to construct frexes for combined theories.

We begin with a brief recourse of multi-sorted universal algebra and its application, through frex, to partial evaluation. Next, we describe a modular construction of an involutive algebra frex out of the frex of the underlying algebra. This construction encompasses situations such as: string concatenation and reversal; complex number addition/multiplication and conjugation; matrix addition/multiplication and transpose. The second modular construction concerns the free multi-sorted extension of a graph of algebras and homomorphisms between them, using the single-sorted frex for each algebra. Examples here cover monoids with (concrete) fold operators on them, as well as calculations that come up when mechanising the calculations involved in establishing the frex universal property (Allais et al. 2023). The key insight here is using the set of nodes from which we can reach each node in the graph. For the final construction, we describe ongoing work on Hyland and Power's distributive combination of theories. These can account for the combinations of non-deterministic choice with probabilistic choice, and we offer a generalisation that accounts for non-affine theories. We conclude by outlining prospects and further directions.

Keywords: free extension, algebra, partial evaluation

1 Introduction

Representing code fragments modulo semantic equivalences is a recurring problem in partial evaluation and code generation [e.g. Carette and Kiselyov 2011; Carette et al. 2009; Kiselyov et al. 2004; Rompf et al. 2013; Yallop 2017]. Consider the arithmetic expression $2 + (x - 2 + x)$, where x will take integer values. One common optimisation technique first calculates a representation of the expression modulo the semantic equivalences — associativity, commutativity, etc. — and evaluation of constants. In this case, we can represent the expression as the pair $\langle 0, 2\rangle \in \mathbb{Z} \times \mathbb{N}$, where the first component represents the sum of the statically known summands, and the second component represents the linear coefficient of the dynamically known summand. From this representation, we can generate the more efficient expression $2x$. These kind of representations generalize *partially-static data structures* [Mogensen 1988] — i.e. values that contain both static fields and fields whose values are known only dynamically. As the collection of syntactic constructs and associated semantic equivalences grows, the partially-static representation changes, potentially substantially. Here, we demonstrate modular construction of these data structures, combining representations for simpler collections of syntactic constructs and associated equations to form a representation for the combined operations and equations.

More recently, Yallop et al. [2018] proposed a unifying theory for specifying (generalized) partially-static data-structures by recourse to universal algebra. We specify:

- expressions by algebraic terms over an algebraic signatures of n-ary operators over a carrier set, e.g. a binary operator $(*) : \mathsf{carrier}^2 \longrightarrow \mathsf{carrier}$ and nullary operator $e : \mathsf{carrier}^0 \longrightarrow \mathsf{carrier}$;
- semantic equations by algebraic axioms between such terms with free variables, e.g. $x*y = y*x$ and $e*x = x$;

- static values are elements of an algebra for this signature validating the equations, e.g. the commutative monoid $(\mathbb{Z}, (+), 0)$ interpreting carrier $:= \mathbb{Z}$, $(*) := (+)$ and $e := 0$; and
- the partially-static data structure represents the *free extension (frex)* of this algebra by a set of variables, e.g. we represent the extension with a single variable x by $\mathbb{Z} \times \mathbb{N}$ through the identification:

$$\langle a, n \rangle \qquad \leftrightarrow \qquad a + n \cdot x$$

The frex offers a uniform theoretical foundation for these diverse data structures. Yallop et al. developed libraries for staging-based optimisations with a nonetheless uniform, frex-based, interface. We report about ongoing development where we use frexes for component algebraic theories to obtain frexes for combined theories.

To keep this manuscript self-contained, we start with a tutorial on universal algebra and free extensions. To nonetheless provide novelty still, we present multi-frex in S2: free extensions of multi-sorted theories. These results are standard and the theoretical development is near-identical for the multi-sorted case, but the current sequence of frex papers has not yet presented the multi-sorted development, which we hope strikes a good balance as an introduction in a research manuscript. In the accompanying talk, we plan to stay closer to the examples and intuition, and refer interested readers to this manuscript for details.

Our contributions:

- S3: We re-use a frex for $\mathcal{T}$-algebras to construct a frex for *involutive* $\mathcal{T}$-algebras: those with an involution. E.g.: lists with reversal, matrices with transposition, complex numbers and quaternions with conjugation. This reuse is *modular* in the following sense. We define a frex partial evaluator for an involutive $\mathcal{T}$-algebra by re-using the frex for its underlying $\mathcal{T}$-algebra. Specifically, we define the evaluator for the additional involution operation solely through recourse to the frex interface. Therefore, with every implementation of a frex partial evaluator for a theory $\mathcal{T}$, we can also generically derive implementations for partial evaluators to its involutive $\mathcal{T}$-algebras.
- S4: We similarly re-use a frex for $\mathcal{T}$-algebras to construct a frex for a family of $\mathcal{T}$-algebras related by a graph of $\mathcal{T}$-homomorphisms. E.g.: logarithms and exponentials, map-fusion for lists, map-reduce for lists.
- In ongoing work, we consider the distributive combination of two theories [Hyland and Power 2006], which we hope to use to construct ring simplifiers out of monoid simplifiers. Covering this work in detail is beyond the scope of this manuscript, and we will mention it briefly in the concluding S5.

We conclude (S5) with other prospects and further directions. Most importantly, the work to date is theoretical in nature, and establishes the correctness of the proposed data-structures and normalisation procedures. We hope to begin the implementation phase in the near future.

2 Multi-sorted free extensions

To be able to deal with algebras generically, we need generic descriptions of the syntax and semantics of algebraic structures, and use multi-sorted universal algebra.

2.1 Syntax

The data we need for the syntax is a *multi-sorted signature*

$$\Sigma = \left\langle \mathbf{sort}_\Sigma, \mathbf{operator}_\Sigma, \mathbf{arity}_\Sigma \right\rangle$$

consisting of:

- a set $\mathbf{sort}_\Sigma$: we call its elements s *sorts*;
- a set $\mathbf{operator}_\Sigma$: we call its elements op *operators*;
- a function $\mathbf{arity}_\Sigma : \mathbf{operator}_\Sigma \to (\text{List}\,\mathbf{sort}_\Sigma) \times \mathbf{sort}_\Sigma$: when $\mathbf{arity}_\Sigma(\text{op}) = \left\langle \langle s_1, \ldots, s_n \rangle, s \right\rangle$, we:
  - write $\vdash_\Sigma \text{op} : s_1 \times \cdots \times s_n \to s$ and call:
  - s the *sort* of the operator,
  - $\langle s_1, \ldots, s_n \rangle$ its arity, and
  - say that the operator is n-ary.

Example 2.1 (multiplicative and additive signatures)**.** We define the signature **multiplicative** by:

- a single sort $\mathbf{sort} := \{\text{carrier}\}$;
- two operators $\mathbf{operator} := \{(\cdot), 1\}$
- their arities $\vdash (\cdot) : \mathsf{carrier} \times \mathsf{carrier} \to \mathsf{carrier}$ and $\vdash 1 : () \to \mathsf{carrier}$;

We use standard syntactic conventions to present such data briefly, e.g. we define the similar signature **additive** with the same, single, sort carrier and operators:

$$\vdash (+) : \mathsf{carrier}^2 \to \mathsf{carrier} \qquad \vdash 0 : \mathsf{carrier}$$

Example 2.2. The signature **FinDim** of *finite dimensional transformations*, with which we describe linear transformations / matrix multiplication has as sort $\mathbf{Hom}(m, n)$ for each pair of natural numbers $m, n \in \mathbb{N}$ and two operators:
$\vdash (\circ_{m,n,k}) : \mathbf{Hom}(n, k) \times \mathbf{Hom}(m, n) \to \mathbf{Hom}(m, k)$
$\vdash \text{Id}_n : \mathbf{Hom}(n, n)$.
This example makes essential use of multiple sorts.

Example 2.3 (involution)**.** Given a single-sorted signature Σ, i.e. $\mathbf{sort}_\Sigma = \{s\}$, we form its associated *involutive* signature $\mathbf{inv}\Sigma$ by formally adding a unary operator $\vdash \overline{(-)} : s \to s$. For **FinDim**, we define two involutive signatures by adding one of the following sequences of unary operators:

- $\vdash_{\mathbf{inv}_{\mathbf{rev}}\Sigma} \overline{(-)}_{m,n} : \mathbf{Hom}(n, m) \to \mathbf{Hom}(m, n)$: reversing
- $\vdash_{\mathbf{inv}_{\mathbf{id}}\Sigma} \overline{(-)}_{m,n} : \mathbf{Hom}(m, n) \to \mathbf{Hom}(m, n)$: direct

Given a signature Σ, a Σ-*algebra* A consists of:

- an assignment of a set $A\llbracket s \rrbracket_{\mathbf{sort}}$ to every $s \in \mathbf{sort}_\Sigma$;

- an assignment of a function:

$$A\llbracket \mathrm{op} \rrbracket_{\mathbf{op}} : (A\llbracket s_1 \rrbracket_{\mathbf{sort}} \times \cdots \times A\llbracket s_n \rrbracket_{\mathbf{sort}}) \to A\llbracket s \rrbracket_{\mathbf{sort}}$$

  to every operator $\vdash_\Sigma \mathrm{op} : s_1 \times \cdots \times s_n \to s$ in Σ.

Example 2.4. A **multiplicative**-algebra A then amounts to a triple $\left\langle A\llbracket \mathsf{carrier} \rrbracket_{\mathbf{sort}}, A\llbracket (\cdot) \rrbracket_{\mathbf{op}}, A\llbracket 1 \rrbracket_{\mathbf{op}} \right\rangle$ consisting of:

- a set $A\llbracket \mathsf{carrier} \rrbracket_{\mathbf{sort}}$ called the *carrier*;
- a binary operation $A\llbracket (\cdot) \rrbracket_{\mathbf{op}} : \llbracket \mathsf{carrier} \rrbracket^2 \to \llbracket \mathsf{carrier} \rrbracket$ called multiplication; and
- an element $A\llbracket \langle\rangle \rrbracket_{\mathbf{op}} \in A\llbracket \mathsf{carrier} \rrbracket_{\mathbf{sort}}$ called the unit.

An **additive**-algebra A consists of the same data. We thus have the **multiplicative**-algebras $\langle \mathbb{N}, (\cdot), 1 \rangle$, $\langle \mathbb{Z}, (\cdot), 1 \rangle$, and so on given by arithmetic multiplication over the naturals, integers, etc., and the corresponding **additive**-algebras over the same carriers given by the arithmetic addition. The same carriers also have other algebraic structures, for example the **additive**-algebra given by $\langle \mathbb{N}, \max, 0 \rangle$.

Example 2.5. Let $\mathbb{M}^{\mathbb{R}}_{m\times n}$ be the set of m rows by n columns matrices with real-number entries. We define a **FinDim**-algebra by matrix multiplication:

$$\begin{aligned}\llbracket \mathbf{Hom}(m,n) \rrbracket &:= \mathbb{M}^{\mathbb{R}}_{m\times n} \\ \llbracket (\circ)_{m,n,k} \rrbracket &:= (\cdot)\end{aligned} \qquad \llbracket \mathrm{Id}_n \rrbracket := I_n := \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

This example makes essential use of multiple sorts.

Example 2.6. The **multiplicative**-algebra $\langle \mathbb{S}, (+\!\!+), \varepsilon \rangle$ given by the set of strings over some alphabet of letters $\mathbb{L}$ together with string concatenation and the empty string extends to an **inv multiplicative**-algebra $\langle \mathbb{S}, (+\!\!+), \varepsilon, (-)^{\mathrm{rev}} \rangle$ by interpreting the additional operation using string reversal:

$$(a_1 \cdots a_n)^{\mathrm{rev}} := a_n \cdots a_1$$

The **FinDim**-algebra of matrix multiplication has a reversing involution, i.e., a $\mathbf{inv}_{\mathbf{rev}}\mathbf{FinDim}$-algebra structure, given by matrix transposition:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}^{\mathrm{t}} := \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}$$

The **FinDim**-algebra of matrices with complex number entries together with multiplication has a direct/non-reversing involution given by mapping the complex conjugation involution $\overline{(a+ib)} := a - ib$ on its entries:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}^{\mathrm{conj}} := \begin{pmatrix} \overline{a}_{11} & \overline{a}_{12} & \cdots & \overline{a}_{1n} \\ \overline{a}_{21} & \overline{a}_{22} & \cdots & \overline{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{a}_{m1} & \overline{a}_{m2} & \cdots & \overline{a}_{mn} \end{pmatrix}$$

The last two examples make essential use of multiple sorts.

2.2 Terms

Given a set **sort**, a **sort**-*indexed family* X is a sequence of sets $\langle X\llbracket s \rrbracket \rangle_{s\in\mathbf{sort}}$ indexed by the elements of **sort**. Given a signature Σ and a $\mathbf{sort}_\Sigma$-indexed family X, we define the **sort**-indexed family $\mathrm{Term}_\Sigma\, X$ of Σ-*terms over* X by induction:

$$\frac{x \in X\llbracket s \rrbracket}{\mathbf{var}\, x \in \mathrm{Term}_\Sigma\, X\llbracket s \rrbracket} \quad s \in \mathbf{sort}$$

$$\frac{t_i \in \mathrm{Term}_\Sigma\, X\llbracket s_i \rrbracket}{\mathrm{op}(t_1,\ldots,t_n) \in \mathrm{Term}_\Sigma\, X\llbracket s \rrbracket} \quad \mathrm{op} : s_1 \times \cdots s_n \to s$$

This family extends to a Σ-algebra by defining:

$$\mathrm{Term}_\Sigma \llbracket \mathrm{op} \rrbracket_{\mathbf{op}} (t_1,\ldots,t_n) := \mathrm{op}(t_1,\ldots,t_n)$$

Example 2.7. The following expressions are **additive**-terms over $X\llbracket \mathsf{carrier} \rrbracket := \{1,2,3\}$:

$$\mathbf{var}\,2 + ((\mathbf{var}\,1 + \mathbf{var}\,3) + \mathbf{var}\,1) \qquad \mathbf{var}\,1 + \mathbf{var}\,1$$

equivalently, by setting $x := \mathbf{var}\,1, y := \mathbf{var}\,2, z := \mathbf{var}\,3$:

$$y + ((x+z)+x) \qquad x + x$$

So $\mathrm{Term}_\Sigma\, X$ is an algebra over abstract syntax trees whose nodes are Σ-operators with variables taken from X. The family of variables X embeds into terms by the **sort**-indexed family of functions $\mathbf{var}_s : X\llbracket s \rrbracket \to \mathrm{Term}_\Sigma\, X\llbracket s \rrbracket$.

2.3 Homomorphisms

A Σ-homomorphism $h : A \to B$ between Σ-algebras is a $\mathbf{sort}_\Sigma$-indexed family of functions $h_s : A\llbracket s \rrbracket \to B\llbracket s \rrbracket$ that moreover satisfy the following equation for every $\mathrm{op} : s_1 \times \cdots \times s_n \to s$ and $(a_1,\ldots,a_n) \in A\llbracket s_1 \rrbracket \times \cdots \times A\llbracket s_n \rrbracket$:

$$h_s(A\llbracket \mathrm{op} \rrbracket (a_1,\ldots,a_n)) = B\llbracket \mathrm{op} \rrbracket (h_{s_1} a_1, \ldots, h_{s_n} a_n)$$

Example 2.8. For every positive real number $r > 0$, we have two **multiplicative**-homomorphisms:

$$\begin{aligned} r^{-} &: \langle \mathbb{R}, (+), 0 \rangle &&\to \langle (0,\infty), (\cdot), 1 \rangle \\ \log_r &: \langle (0,\infty), (\cdot), 1 \rangle &&\to \langle \mathbb{R}, (+), 0 \rangle \end{aligned}$$

since $r^{a+b} = r^a \cdot r^b$, $r^0 = 1$ and $\log_r a \cdot b = \log_r a + \log_r b$, $\log_r 1 = 0$.

Example 2.9. Conjugation is a **multiplicative** and **additive**-homomorphism between the corresponding algebras over the complex numbers, since: $\overline{z_1 + z_2} = \overline{z}_1 + \overline{z}_2$, $\overline{0} = 0$ and $\overline{z_1 \cdot z_2} = \overline{z}_1 \cdot \overline{z}_2$, $\overline{1} = 1$.

Similarly, component-wise conjugation is a **FinDim**-homomorphism $(-)^{\mathrm{conj}} : \mathbb{M}^{\mathbb{C}} \to \mathbb{M}^{\mathbb{C}}$, and this example makes essential use of multiple sorts. However, transposition is not a homomorphism $(-)^{\mathrm{t}} : \mathbb{M} \to \mathbb{M}$ since it is not a **sort**-indexed family of functions: the domain and codomain of its components have different sorts.

2.4 Equations and axioms

A Σ*-equation in context* $X \vdash l \approx r : s$ consists of:

- a $\mathbf{sort}_\Sigma$-family X;
- a sort $s \in \mathbf{sort}_\Sigma$; and
- two Σ-terms over X: $l, r \in \mathrm{Term}_\Sigma\, X\, [\![s]\!]$ called the *left-hand-side* (LHS) and the *right-hand-side* (RHS).

Example 2.10. The **multiplicative** *associativity* equation:

$$x, y, z : \mathsf{carrier} \vdash \mathbf{var}\, x \cdot (\mathbf{var}\, y \cdot \mathbf{var}\, z) \approx (\mathbf{var}\, x \cdot \mathbf{var}\, y) \cdot \mathbf{var}\, z : \mathsf{carrier}$$

For brevity, we'll omit the injection **var** and the sort. For example, the *neutrality* equations are:

$$x : \mathsf{carrier} \vdash x \cdot 1 \approx x \qquad x : \mathsf{carrier} \vdash 1 \cdot x \approx x$$

A *presentation* $\mathcal{T} = \langle \Sigma_\mathcal{T}, \mathrm{Ax}_\mathcal{T} \rangle$ consists of:

- a signature $\Sigma_\mathcal{T}$; and
- a set $\mathrm{Ax}_\mathcal{T}$ of Σ-equations called *axioms*.

Example 2.11. The **Monoid** presentation axioms are associativity and neutrality over the **multiplicative** signature.

The **CommutativeMonoid** presentation has the **additive** signature, and as axioms the analogous associativity and neutrality equations, with the following commutativity axiom:

$$x, y : \mathsf{carrier} \vdash x + y \approx y + x$$

Example 2.12. The presentation **FinTrans** of finite dimensional transformations over the signature **FinDim** has multi-sorted analogues for associativity and neutrality:

$$f : \mathbf{Hom}(k, \ell), g : \mathbf{Hom}(n, k), h : \mathbf{Hom}(m, n) \vdash f \circ (g \circ h) \approx (f \circ g) \circ h : \mathbf{Hom}(m, \ell)$$

$$f : \mathbf{Hom}(m, n) \vdash f \circ \mathrm{Id}_m \approx f : \mathbf{Hom}(m, n)$$
$$f : \mathbf{Hom}(m, n) \vdash \mathrm{Id}_n \circ f \approx f : \mathbf{Hom}(m, n)$$

This example makes essential use of multiple sorts.

Example 2.13. The presentation $\mathbf{inv_{rev}Monoid}$ of *reversing involutive monoids* over the **multiplicative** signature adds the three involutive axioms:

$$x : \mathsf{carrier} \vdash \overline{\overline{x}} \approx x \quad x, y : \mathsf{carrier} \vdash \overline{x \cdot y} \approx \overline{y} \cdot \overline{x} \quad \vdash \overline{1} \approx 1$$

The presentation $\mathbf{inv_{Id}Monoid}$ of *non-reversing involutive monoids* changes the middle axiom to:

$$x, y : \mathsf{carrier} \vdash \overline{x \cdot y} \approx \overline{x} \cdot \overline{y}$$

By a similar process, define the presentations $\mathbf{inv_{rev}FinTrans}$, and $\mathbf{inv_{Id}FinTrans}$, with the following axioms:

$$f : \mathbf{Hom}(n, k), g : \mathbf{Hom}(m, n) \vdash$$
$$\overline{f \circ g} \approx \overline{g} \circ \overline{f} : \mathbf{Hom}(k, m) \qquad \text{(reversing)}$$
$$\overline{f \circ g} \approx \overline{f} \circ \overline{g} : \mathbf{Hom}(m, k) \qquad \text{(direct)}$$

The last two examples make essential use of multiple sorts.

2.5 Validity

Let A be a Σ-algebra, and X a **sort**-indexed family. An X*-environment in* A e is a **sort**-indexed function $e : X \to A\,[\![-]\!]_{\mathbf{sort}}$ assigning to each variable in $X\,[\![s]\!]$ an A-value of the same sort. Every X-environment e in A induces an *homormorphic interpretation* Σ-homomorphism $A\,[\![-]\!]\,e : \mathrm{Term}_\Sigma\, X \to A$ given inductively by:

$$A\,[\![\mathbf{var}\, x : s]\!]\, e \coloneqq e_s x$$
$$A\,[\![\mathrm{op}(t_1, \ldots, t_n)]\!]\, e \coloneqq A\,[\![\mathrm{op}]\!]\,(A\,[\![t_1]\!]\, e, \ldots, A\,[\![t_n]\!])$$

A Σ algebra A *validates* a Σ-equation $X \vdash l \approx r : s$ when, for every X-environment e in A we have $A\,[\![l]\!]\, e = A\,[\![r]\!]\, e$. Given a presentation $\mathcal{T}$, a $\mathcal{T}$-algebra is then a $\Sigma_\mathcal{T}$-algebra that validates all the axioms in $\mathcal{T}$.

By this point we hope it's clear that a monoid is a **Monoid**-algebra, a commutative monoid is a **CommutativeMonoid**-algebra, and so on.

Example 2.14. The algebra of *finite-dimensional linear transformations* is the **FinDim**-algebra $\mathbb{M}$, which validates the axioms of **FinTrans**. Its extension with transposition validates the axioms of $\mathbf{inv_{rev}FinTrans}$. The analogous algebra of complex-valued matrices with component-wise conjugation validates the axioms of $\mathbf{inv_{id}FinTrans}$. All of these examples make essential use of multiple sorts.

2.6 Free extensions

We now have the vocabulary to talk generically about algebraic expressions over a concrete domain of discourse, extended with a collection of free variables modulo some behavioural equivalences and evaluation. We represent the semantic equivalences by considering a presentation $\mathcal{T}$, e.g., **CommutativeMonoid**. The domain of discourse is a $\mathcal{T}$-algebra A, e.g., $(\mathbb{Z}, (\cdot), 1)$. The collection of free variables we wish to adjoin is given by a $\mathbf{sort}_{\Sigma_\mathcal{T}}$-indexed family X, e.g., $\{x_1, \ldots, x_k : \mathsf{carrier}\}$. Since our extended domain should support the operations in the signature $\Sigma_\mathcal{T}$ and satisfy the behavioural equivalences of interest, we are looking for another $\mathcal{T}$-algebra, $A[X]$. For, e.g., **CommutativeMonoid** it means $A[X]$ supports a commutative associative binary operation and a neutral element. Since variables (= placeholders for dynamic values) ought to embed into the extended algebra, we want a **sort**-indexed function $\mathrm{dyn} : X \to A[X]\,[\![-]\!]_{\mathbf{sort}}$. It allows us to express fully dynamic expressions such as $\mathrm{dyn}\, x \cdot \mathrm{dyn}\, y$. Since we want the algebra A (of static values) to embed into the extended algebra and preserve its behaviour, we want a Σ-homomorphism $\mathrm{sta} : A \to A[X]$. It allows us to express partially-static expressiong such as $\mathrm{sta}\, 2 \cdot \mathrm{dyn}\, x \cdot \mathrm{sta}(-3)$ which, together with the axioms in the theory and the homomorphism condition must be equivalent to $\mathrm{sta}(-6) \cdot \mathrm{dyn}\, x$. Summarising these properties, we want to single out a triple $\langle A[X], \mathrm{sta}, \mathrm{dyn} \rangle$ out of the collection of triples $\langle B, h, e \rangle$ consisting of:

- a $\mathcal{T}$-algebra B;

$$\frac{\mathrm{ax} \in \mathrm{Ax}_{\mathcal{T}}}{\mathrm{ax}} \ \text{(axiom)} \qquad X \vdash t \approx t : s \ \text{(reflexivity)}$$

$$\frac{X \vdash l \approx r : s}{X \vdash r \approx l : s} \ \text{(symmetry)}$$

$$\frac{X \vdash l \approx m : s \quad X \vdash m \approx r : s}{X \vdash r \approx l : s} \ \text{(transitivity)}$$

$$\frac{X \vdash l \approx r : s \quad \theta : X \to \mathrm{Term}_{\Sigma_{\mathcal{T}}} Y}{Y \vdash l[\theta] \approx r[\theta] : s} \ \text{(substitution)}$$

$$\frac{\text{for } i = 1, \ldots, n\text{: } X \vdash l_i \approx r_i : s_i}{X \vdash \mathrm{op}(l_1, \ldots, l_n) \approx \mathrm{op}(r_1, \ldots, r_n) : s} \ \text{(congruence)}$$

Figure 1. Provability

- a Σ-homomorphism $h : A \to B$; and
- a **sort**-indexed function $e : X \to B[\![-]\!]_{\mathbf{sort}}$.

Let's call such a triple an *extension of A by X*. One canonical way to single out such a triple is to define structure-preserving maps between extensions, and use a universal property to single out freely. Formally an *extension morphism* $\varphi : \langle B_1, h_1, e_1 \rangle \to \langle B_2, h_2, e_2 \rangle$ consists of a Σ-homomorphism $\varphi : B_1 \to B_2$ such that:

$$\varphi_s(h_{1,s}a) = h_{2,s}a \qquad (a \in A[\![s]\!])$$
$$\varphi_s(e_{1,s}x) = e_{2,s}x \qquad (x \in X[\![s]\!])$$

An extension $\langle A[X], \mathrm{sta}, \mathrm{dyn} \rangle$ is then *free* when, for every extension $\langle A[X], \mathrm{sta}, \mathrm{dyn} \rangle \to \langle B, h, e \rangle$, there is a unique extension morphism $[B; h; e] : \langle A[X], \mathrm{sta}, \mathrm{dyn} \rangle \to \langle B, h, e \rangle$.

To justify why free extensions are what we look for, here is a more concrete characterisation of the free extension as a quotient. Every presentation $\mathcal{T}$ induces an equivalence relation between terms using the deduction rules of multi-sorted equational logic. Explicitly, we define the *provability* relation $l \vdash_{\mathcal{T}} r \approx s$: inductively as in Fig 1. It is the smallest relation containing the axioms that is closed under the rules for an equivalence (reflexivity, symmetry, and transitivity), substituting the same terms into equivalent terms (substitution), and substitution by equals into the same term (congruence).

Now given any presentation $\mathcal{T}$, let $\mathcal{T}_A$ be the presentation:

- whose signature consists of:
  - the same set of sorts; and
  - the operations in $\Sigma_{\mathcal{T}}$ together with newly adjoined constants $\underline{a}_s : s$ for every $a \in A[\![s]\!]$; and
- whose axioms include the axioms in $\mathcal{T}$, and in addition, for every $\mathrm{op} : s_1 \times \cdots \times s_n \to s$ and $(a_1, \cdots a_n) \in A[\![s_1]\!] \times \cdots \times A[\![s_n]\!]$, the axiom:

$$\vdash \mathrm{op}(\underline{a}_1, \ldots, \underline{a}_n) \approx \underline{A[\![\mathrm{op}]\!](a_1, \ldots, a_n)} : s \qquad \text{(evaluation)}$$

Then for every frex $\langle A[X], \mathrm{sta}, \mathrm{dyn} \rangle$ there is an isomorphism, i.e. a bijective homomorphism whose inverse is a homomorphism, $A[X] \cong (\mathrm{Term}_{\Sigma_{\mathcal{T}_A}} X)/(X \vdash_{\mathcal{T}_A})$. I.e., the carrier sets are equivalence classes of terms modulo the evaluation axioms and $\mathcal{T}$'s axioms. This quotient Q is an extension via:

- the homomorphism $\mathrm{sta}' : A \to Q$ by $\mathrm{sta}' a := [\underline{a}]$; and
- the functions $\mathrm{dyn}' : X \to Q[\![-]\!]$ by $\mathrm{dyn}'_s x := [\mathbf{var}\, x]$

mapping A-elements and variables to their equivalence class. The isomorphism $A[X] \cong Q$ is an isomorphism of extensions $\langle A[X], \mathrm{sta}, \mathrm{dyn} \rangle \cong \langle Q, \mathrm{sta}', \mathrm{dyn}' \rangle$. In this sense the free extension $A[X]$ represents terms-modulo-semantics.

Example 2.15. Let A be a commutative monoid. Its frex by a finite set of variables can be represented by:

$$A[x_1, \ldots, x_n] := A[\![\mathsf{carrier}]\!] \times \mathbb{N}^n$$

this frex represents the term quotient via:

$$\langle a, \langle k_1, \ldots, k_n \rangle \rangle \leftrightarrow [\underline{a} + k_1 \cdot x_1 + \cdots + k_n \cdot x_n]$$

When X is an infinite set, use finite bags of variables:

$$A[X] := A[\![\mathsf{carrier}]\!] \times \mathrm{Bag}(X[\![\mathsf{carrier}]\!])$$
$$\langle a, \wr x_1 : k_1, \ldots, x_n : k_n \wr \rangle \leftrightarrow [\underline{a} + k_1 \cdot x_1 + \cdots + k_n \cdot x_n]$$

The remaining structure is given by:

$$\langle a, U \rangle + \langle b, V \rangle := \langle a + b, U \cup V \rangle \qquad 0 := \langle 0, \emptyset \rangle$$
$$\mathrm{sta}\, a := \langle a, \emptyset \rangle \qquad \mathrm{dyn}\, x := \langle 0, \wr x \wr \rangle$$
$$[B; h; e]\, \langle a, \wr x_1 : k_1, \ldots, x_n : k_n \wr \rangle := h\, a + k_1 \cdot e\, x_1 + \cdots + k_n \cdot e\, x_n$$

Example 2.16. Let A be a monoid. We define one possible frex $A[X]$ as follows. Given a set U and a relation $(\sim)$ over U, we define the set $\mathrm{List}_{\nsim} U$ of *$\sim$-alternating lists over U* by:

$$\mathrm{List}_{\nsim} U := \left\{ \langle u_1, \ldots, u_n \rangle \in \mathrm{List}\, U \middle| \forall i = 1, \ldots, n. u_i \nsim u_{i+1} \right\}$$

Let $A_{\neq 1} := \{a \in A[\![\mathsf{carrier}]\!] | a \neq 1\}$ denote the non-neutral elements. Take $A[X]$ to be the $\sim$-alternating lists over the disjoint union $A_{\neq 1} + (\mathbb{N}_+ \times X)$ of the non-neutral elements in the carrier A and variables tagged with a positive natural. We define the alternation relation by $a \sim b$ for all $a, b \in A$ and $(n, x) \sim (m, x)$ for all $x \in X$. The alternation condition means that consecutive variables are different, and every two non-neutral A-elements are separated by at some variables. The positive natural tagging each variable represents the multiplicity in which it occurs.

To multiply frex elements, we concatenate them while multiplying adjacent concrete elements, taking care to remove them if they cancel to 1. In that case, we also check whether two adjacent variables are the same, and if so merge them while adding their multiplicity.

Example 2.17. Let A be any **FinTrans**-algebra, such as $\mathbb{M}$, and X a set of variables. Take $A[X][\![\mathbf{Hom}(s,t)]\!]$ to be the $\sim$-alternating lists over non-neutral elements of the dependent sum $\sum_{m,n}(A[\![\mathbf{Hom}(m,n)]\!] + X[\![\mathbf{Hom}(m,n)]\!])$ where we allow the alternations $\langle m, n; u \rangle \nsim \langle n, k; v \rangle$ of composable elements and prohibit alternations of A-elements, and moreover only include the empty list when $s = t$, and require that the source of the last-element and the target of the first element are s and t respectively.

Like the monoid frex, we compose composable elements by concatenation, taking care to compose adjacent A-elements, and deleting any resulting identity $A[\![\mathrm{Id}_n]\!]$. This example makes essential use of multiple sorts.

Compact representations of the frex can be nuanced, or maintain subtle semantic invariants. In the remainder of this manuscript we explore ways in which we can modularly combine these representations without compromising the nuanced invariants, by maintaining the frex interface.

3 Involutive algebras

Consider a reversing involutive monoid A (Ex. 2.9), and let A' be the underlying monoid (i.e. we forget the involutive operation). The basic observation in our first construction is that we can represent $A[X]$ using $A'[\mathrm{Bool}\times X]$. The boolean tag marks whether a variable appears as involved. To involve an element in this frex, which we represented as a list in Ex. 2.16, we reverse the list, involve the concrete elements (which maintains their non-neutrality thanks to the axiom $\overline{1} = 1$), and negate the boolean tags on each variable while maintaining its multiplicity. This operation may seem involved at first sight, but we can access it generically — for any frex of any presentation through the frex interface.

To do so, we use Jacobs's [2021] axiomatisation of involution, specialised to our concrete setting. An *involutive structure* $\overline{(-)}$ for a presentation $\mathcal{T}$ consists of:

- an *involutive permutation* $\overline{(-)}_{\mathbf{sort}} : \mathbf{sort} \cong \mathbf{sort}$ on the set of sorts, i.e. $\overline{\overline{s}} = s$; and
- for each $\mathcal{T}$-algebra A, a $\mathcal{T}$-algebra $\overline{A}$ such that:
  - $\overline{A}[\![s]\!] = A[\![\overline{s}]\!]$ for every $s \in \mathbf{sort}$; and
  - $\overline{\overline{A}} = A$.

Example 3.1. Monoids have the involutive structure $(-)^{\mathrm{rev}}$:

- $\mathrm{carrier}^{\mathrm{rev}} := \mathrm{carrier}$;
- $A^{\mathrm{rev}}[\![\mathrm{carrier}]\!] := A[\![\mathrm{carrier}]\!]$ and $A^{\mathrm{rev}}[\![1]\!] := A[\![1]\!]$; and
- $A^{\mathrm{rev}}[\![(\cdot)]\!]\langle x, y\rangle := y \cdot x$

The resulting **multiplicative**-algebra is a monoid since the monoid axioms are symmetric under reversal, and it's straightforward to check that $(A^{\mathrm{rev}})^{\mathrm{rev}} = A$.

Example 3.2. The presentation **FinTrans** has the following involutive structure $(-)^{\mathrm{rev}}$:

- $\overline{A}[\![\mathbf{Hom}(m,n)]\!] := \mathbf{Hom}(n,m)$;
- $\overline{A}[\![\mathrm{Id}_n]\!] := A[\![\mathrm{Id}_n]\!]$, and $\overline{A}[\![(\circ)]\!]\langle f, g\rangle := g \circ f$.

This example makes essential use of multiple sorts.

Example 3.3. Given an "empty" sorted-signature, i.e. a set **sort** of sorts, each involutive permutation $\overline{(-)} : \mathbf{sort} \cong \mathbf{sort}$ induces an involutive structure on the **sort**-indexed sets.

Every presentation $\mathcal{T}$ has an identity involutive structure that keeps the sorts and the algebras the same.

The 'hidden' structure behind the notion of an involutive structure is that it extends to a functor on the $\mathcal{T}$-algebras that preserves an underlying involutive functor on the **sort**-indexed families. While we do not make this structure more explicit here, it powers our proofs.

Lemma 3.4. *Let $\overline{(-)}$ be an involutive structure for $\mathcal{T}$. Then the components of every $\mathcal{T}$-homomorphism $h : A \to B$ form a $\mathcal{T}$-homomorphism $\overline{h} : \overline{A} \to \overline{B}$.*

The purpose of involutive structures is to allow us to define involutions in general. Let $\overline{(-)}$ be an involutive structure on $\mathcal{T}$. An $\overline{(-)}$*-involutive $\mathcal{T}$-algebra* $\left\langle A, \overline{(-)}\right\rangle$ consists of:

- a $\mathcal{T}$-algebra A; and
- a $\mathcal{T}$-homomorphism $\overline{(-)} : \overline{A} \to A$;

such that $\overline{\overline{a}} = a$ for every $a \in A[\![s]\!]$.

Example 3.5. In Ex. 2.9, we saw that a reversing involution $\overline{(-)}$ over a monoid A is *not* a **multiplicative**-homomorphism. The reversing involutive structure on monoids allows us to see it as a homomorphism $\overline{(-)} : \overline{A} \to A$. We then have that involutive monoids amount to involutive **Monoid**-algebras.

Example 3.6. Matrix transposition is a reversing involution over the algebra of finite dimensional linear transformations. Component-wise conjugation is a non-reversing involution over the finite dimensional complex linear transformations. These two examples make essential use of multiple sorts.

An *involutive algebra homomorphism* between two given involutive algebras $h : \langle A, \overline{(-)}\rangle \to \langle B, \overline{(-)}\rangle$ is a homomorphism $h : A \to B$ that preserves the involution: $\overline{h\,a} = h\,\overline{a}$. We will now show that if the algebras for a presentation $\overline{\mathcal{T}}$ are involutive algebras for some presentation $\mathcal{T}$ then we can construct frexes for $\overline{\mathcal{T}}$ from frexes for $\mathcal{T}$. More precisely: A *presentation of $\overline{(-)}$-involutive $\mathcal{T}$-algebras* consists of:

- A presentation $\overline{\mathcal{T}}$ with the same sorts: $\mathbf{sort}_{\overline{\mathcal{T}}} = \mathbf{sort}_{\mathcal{T}}$.
- An equivalence $E : \mathrm{Alg}\overline{\mathcal{T}} \simeq \mathrm{InvAlg}(\mathcal{T}, \overline{(-)})$ that preserves the underlying indexed sets.

All of our examples for involutive algebras have associated presentations in this sense.

Theorem 3.7 (modular frex construction for involutive algebras)**.** *Let $\langle\overline{\mathcal{T}}, E\rangle$ be a presentation of $\overline{(-)}$-involutive algebras, and A a $\overline{\mathcal{T}}$-algebra. Then:*

$$A[X] := E^{-1}\left\langle EA[\mathrm{Bool}\times X], \overline{(-)}\right\rangle$$

sta *is inherited as is,* $\mathrm{dyn}\,x$ *is given by* $\mathrm{dyn}(\mathbf{false}, x)$*, and the involution is given by* $\overline{h} : \overline{EA[\mathrm{Bool}\times X]} \to EA[\mathrm{Bool}\times X]$*, where h is the unique $\mathcal{T}$-homomorphism satisfying:*

$$h \circ \mathrm{sta} = \overline{\mathrm{sta}} \circ \overline{=}_{EA} \qquad h \circ \mathrm{dyn}_s = \mathrm{dyn}_{\overline{s}} \circ (\neg) \times \overline{(-)}$$

Example 3.8. In Ex. 3.5 we saw that involutive monoids amount to involutive **Monoid**-algebras. Thus, strings with string reversal (Ex. 2.6) are a reversing involutive monoid, and we can apply the theorem to the frex we constructed in Ex. 2.16. For example, we can evaluate the following expression inside the frex with two adjoined variables $X \coloneqq \{x, y\}$:

$$(x \mathbin{+\!\!+} \text{“ olleH”})^{\text{rev}} \mathbin{+\!\!+} \text{“, world”} \mathbin{+\!\!+} y^2$$

It evaluates to this alternating sequence:

$$(\text{“Hello ”}, (1, (\mathbf{true}, x)), \text{“, world”}, (2, (\mathbf{false}, y)))$$

which represents the normalised partially-static expression:

$$\text{“Hello ”} \mathbin{+\!\!+} x^{\text{rev}} \mathbin{+\!\!+} \text{“, world”} \mathbin{+\!\!+} y^2$$

The homomorphism $[\text{sta}; \begin{pmatrix} x \mapsto \text{“ereht”} \\ y \mapsto \text{“!!”} \end{pmatrix}]$ maps the sequence to:

$$\text{“Hello there, world!!!!”}$$

Example 3.9. Similarly, we can use this theorem to partially evaluate finite dimensional matrices with transposition. For example, extending with a single 2-dimensional vector $X \coloneqq \{x : \mathbf{Hom}(2, 1)\}$, we can partially evaluate the expression:

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \left(x^{\text{t}} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\right)^{\text{t}} x^{\text{t}}$$

It evaluates to this alternating sequence:

$$\left(\begin{pmatrix} 0 & 2 \\ -1 & 0 \end{pmatrix}, (1, (\mathbf{false}, x)), (1, (\mathbf{true}, x))\right)$$

which represents the normalised partially-static expression:

$$\begin{pmatrix} 0 & 2 \\ -1 & 0 \end{pmatrix} x x^{\text{t}}$$

Each homomorphism $[\text{sta}; (x \mapsto (a\ b)^{\text{t}})]$ sends it to:

$$\begin{pmatrix} 2ba & b^2 \\ -a^2 & -ab \end{pmatrix}$$

This example makes essential use of multiple sorts.

4 Homomorphism graphs

Consider the additive commutative monoid over the integers $\langle \mathbb{Z}, (+), 0 \rangle$ and the multiplicative commutative monoid over the rationals $\langle \mathbb{Q}, (\cdot), 1 \rangle$. For every integer $a \in \mathbb{Z}$ we have a homomorphism $a \cdot (-) : \langle \mathbb{Z}, (+), 0 \rangle \to \langle \mathbb{Z}, (+), 0 \rangle$, and, for every positive rational $0 < q \in \mathbb{Q}$, we have a homomorphism $q^{-} : \langle \mathbb{Z}, (+), 0 \rangle \to \langle \mathbb{Q}, (\cdot), 1 \rangle$:

$$\begin{tikzcd} \langle \mathbb{Z}, (+), 0 \rangle \arrow[loop left, "a \cdot (-)"] \arrow[r, "q^{-}"] & \langle \mathbb{Q}, (\cdot), 1 \rangle \end{tikzcd}$$

We can write expressions involving exponents of rational numbers, multiplication by integer constants, addition of integer variables and multiplication of rational expressions, for example $x, y : \mathbb{Z}, u : \mathbb{Q} \vdash 5^{2 \cdot (3 \cdot x + y)} \cdot 3^{3 \cdot y + 2 \cdot x} \cdot u^2 \cdot 5^{8 \cdot y}$. We'll show how to modularly construct frexes for such compound terms, involving a graph of algebras of the same theory, connected by homomorphisms. The whole section makes essential use of multiple sorts: even if the starting theory is single-sorted, whenever the ambient graph has more than 1 vertex, the resulting theory will have multiple sorts.

A *quiver* Q is a directed graph that allows loops and multiple edges between the same vertices (multidigraph), i.e., a tuple $\left\langle \mathrm{V}_Q, \mathrm{E}_Q, \mathrm{src}_Q, \mathrm{tgt}_Q \right\rangle$ consisting of:

- sets V_Q, *vertices*, and E_Q, *edges*; and
- functions $\mathrm{src}_Q, \mathrm{tgt}_Q : \mathrm{E}_Q \to \mathrm{V}_Q$ assigning to each edge its source and target vertices.

When $e \in \mathrm{E}_Q$, $\mathrm{src}\, e = u$, $\mathrm{tgt}\, e = v$, we write $e : u \to v$ in Q. Each quiver has an associated category of *paths* $\mathrm{Path}\, Q$: it has the quiver's vertices as objects, and a morphism $p : u \rightsquigarrow v$ is a sequence, potentially empty, $p = \langle e_1, \dots, e_n \rangle$ of *composable* arrows, i.e.:

- $\mathrm{src}\, e_1 = u$, $\mathrm{tgt}\, e_n = v$ (when $n \geq 1$); and
- for all $1 \leq i < n$, we have $\mathrm{tgt}\, e_i = \mathrm{src}\, e_{i+1}$.

Example 4.1. We have a quiver with two vertices, say **integer** and **rational**, and arrows $\mathbf{mult}\, a : \mathbf{integer} \to \mathbf{integer}$ for every integer $a \in \mathbb{Z}$ and $\mathbf{exp}\, q : \mathbf{integer} \to \mathbf{rational}$ for every positive rational $0 < q \in \mathbb{Q}$:

$$\begin{tikzcd} \mathbf{integer} \arrow[loop left, "\mathbf{mult}\, a"] \arrow[r, "\mathbf{exp}\, q"] & \mathbf{rational} \end{tikzcd}$$

It has the paths:

- $\langle \mathbf{mult}\, 3, \mathbf{mult}\, 2 \rangle : \mathbf{integer} \rightsquigarrow \mathbf{integer}$
- $\langle \rangle : \mathbf{rational} \rightsquigarrow \mathbf{rational}$
- $\langle \mathbf{mult}\, 3, \mathbf{mult}\, 2, \mathbf{exp}\, 5 \rangle : \mathbf{integer} \rightsquigarrow \mathbf{rational}$

Let $\mathcal{T}$ be a presentation. A *$\mathcal{T}$-homomorphism graph G* is a quiver $\left\langle \mathrm{V}_G, \mathrm{E}_G, \mathrm{src}_G, \mathrm{tgt}_G \right\rangle$ together with a *$\mathcal{T}$-labelling* $G[\![-]\!]$, consisting of:

- for every vertex $u \in \mathrm{V}_G$, a $\mathcal{T}$-algebra $G[\![u]\!]$; and
- for every edge $e : u \to v \in \mathrm{E}_G$, a $\mathcal{T}$-homomorphism $G[\![e]\!] : G[\![u]\!] \to G[\![v]\!]$.

Such a labelling induces, via composition, the homomorphism $G[\![e_1, \dots, e_n]\!] \coloneqq u \xrightarrow{[\![e_1]\!]} \cdots \xrightarrow{[\![e_n]\!]} v$ for every path $\langle e_1, \dots, e_n \rangle : u \rightsquigarrow v$.

Example 4.2. Our introductory example is a commutative monoid homomorphism graph, labelling the quiver from Ex. 4.1 by:

- $[\![\mathbf{integer}]\!] \coloneqq \langle \mathbf{integer}, (+), 0 \rangle$;
- $[\![\mathbf{rational}]\!] \coloneqq \langle \mathbb{Q}, (\cdot), 1 \rangle$;
- $[\![\mathbf{mult}\, a]\!]\, x \coloneqq a \cdot x$; and $[\![\mathbf{exp}\, q]\!]\, x \coloneqq q^x$

For *open* terms, a $\mathcal{T}$-homomorphism graph G over a quiver Q_G is an algebra for the following presentation $Q^{\mathcal{T}}$:

- sorts are pairs $\langle u, s \rangle \in \mathrm{V}_Q \times \mathbf{sort}_{\mathcal{T}}$, written as u_s;
- for each operation $(f : s_1 \times \cdots \times s_n \to s) \in \mathcal{T}$ and vertex $u \in \mathrm{V}_Q$, an operation $f_u : u_{s_1} \times \cdots \times u_{s_n} \to u_s$;

- for every edge $e : u \to v$ in Q and sort $s \in \mathbf{sort}_{\mathcal{T}}$, an operation $e_s : u_s \to v_s$;
- for every axiom ax $\in \mathcal{T}$ and vertex $u \in \mathrm{V}_Q$, a "copy" of ax in which we label each sort and operation by u; these state that each $\langle u_s \rangle_s$ will form a $\mathcal{T}$-algebra u;
- for every operation $f : s_1 \times \cdots \times s_n \to s$ and edge $e : u \to v$, the axiom:

$$x_1 : u_{s_1}, \ldots, x_n : u_{s_n} \vdash \\ e_s(f_u(x_1, \ldots, x_n)) = f_v(e_{s_1} x_1, \ldots, e_{s_n} x_n)$$

  i.e., each $\langle e_s \rangle_s$ is a $\mathcal{T}$-homomorphism $e : u \to v$.

Thus a $\mathcal{T}$-homomorphism graph over Q is a $Q^{\mathcal{T}}$-algebra, and we may talk about the frex of a $\mathcal{T}$-homomorphism graph.

Example 4.3 (map fusion)**.** We use homomorphism graphs to partially-evaluate map-fusion operations on lists.

Let $\mathcal{I}$ be a collection of sets and $\mathcal{F}$ a collection of functions $f : a \to b$, $f \in \mathcal{F}$ between sets $a, b \in \mathcal{I}$ in $\mathcal{I}$. Consider computations involving $\mathbf{map}\, f$, for any $f \in \mathcal{F}$, and list concatenation $(+\!\!+) :$ List $a \to$ List $a \to$ List a, $a \in \mathcal{I}$ and the empty list $[]$. The associated homomorphism graph $G(\mathcal{I}, \mathcal{F})$ over the theory of monoids has $\mathcal{I}$ as vertices, interpreting each vertex $a \in \mathcal{I}$ as the (free) monoids $\langle \text{List}\, a, (+\!\!+), [] \rangle$. The edges are $\mathcal{F}$ interpreting each edge $(f : a \to b) \in \mathcal{F}$ as the homomorphism $\mathbf{map}\, f : \langle \text{List}\, a, (+\!\!+), [] \rangle \to \langle \text{List}\, b, (+\!\!+), [] \rangle$.

The terms in $Q(\mathcal{I}, \mathcal{F})^{\mathcal{T}}$ express map operations:

$$\mathbf{map}\, f\ (x_1 +\!\!+ \mathbf{map}\, g\, x_2 +\!\!+ \mathbf{map}\, g\, x_3)$$

Example 4.4 (map-reduce)**.** Generalising the previous example, let $\mathcal{J}$ be further a collection of monoids, and $\mathcal{H}$ a collection of homomorphisms between the monoids in List$[\mathcal{I}] \cup \mathcal{J}$. The homomorphism graph $G(\mathcal{I}, \mathcal{J}, \mathcal{F}, \mathcal{H})$ has as vertices the union $\langle \text{List}\, [\mathcal{I}], (+\!\!+), [] \rangle \cup \mathcal{J}$, with each each monoid denoting itself. The edges are the homomorphisms $\mathbf{map}\, [\mathcal{F}] \cup \mathcal{H}$ denoting themselves.

The terms in $G(\mathcal{I}, \mathcal{J}, \mathcal{F}, \mathcal{H})^{\mathcal{T}}$ express homomorphisms between the monoids, including map, and *reduce* operations:

$$\frac{\langle b, (*), e \rangle \in \mathrm{V} \qquad v : a \to b, (\rangle\!\!\models v : \text{List}\, a \to \langle b, (*), e \rangle) \in \mathrm{E}}{\mathbf{reduce}\, (*)\, e\, v : \text{List}\ a \to b}$$

Turning to the frex, given a homomorphism graph, its extending set X is a $\mathrm{V}_G \times \mathbf{sort}$-indexed set X of variables, extending each sort of the algebra at each vertex. Central is this $\mathrm{V}_G \times \mathbf{sort}$-indexed set and the action of paths on it:

$$(\mathbf{P}_G X)_{u,s} \coloneqq \sum_{\substack{v \in \mathrm{V} \\ p \in \mathrm{Path}\,(v,u)}} X\, [\![(v, s)]\!]$$

$$(\odot) : \mathrm{Path}\,(u_1, u_2) \times (\mathbf{P}_G X)_{u_1,s} \to (\mathbf{P}_G X)_{u_2,s}$$

$$\big\langle p, \langle v, q, x \rangle \big\rangle \mapsto \langle v, p \circ q, x \rangle$$

They are crucial because the variables extending sort s of vertex v contribute terms to expressions of the same sort but at vertex u when we promote them along a sequence of homomorphism edges, i.e., a path, $p : v \rightsquigarrow u$. Thus we'll be interpreting the vertex u in the frex $G[X]$ by the frex of its underlying algebra $G\, [\![u]\!]\, [(\mathbf{P}X)_u]$. The action $(\odot)$ lets us equip the frex $G\, [\![u]\!]\, [(\mathbf{P}X)_{u,-}]$ with the structure of an extension of $G\, [\![u']\!]$ by $(\mathbf{P}X)_{u',-}$ for any edge $e : u' \to u$ in G, which we'll use to interpret the homomorphism $G[X]\, [\![e]\!] : G\, [\![u']\!]\, [(\mathbf{P}X)_{u'}] \to G\, [\![u]\!]\, [(\mathbf{P}X)_u]$. These data form a homomorphism graph, and in fact form the frex. Indeed, given any extension $G \xrightarrow{h} B \xleftarrow{e} X$, we thus obtain a family of vertex-wise extensions $G\, [\![u]\!] \xrightarrow{h_u} B\, [\![u]\!] \xleftarrow{e_u} X\, [\![u]\!]$, from which we may construct a vertex-wise family of homomorphisms $G\, [\![u]\!]\, [(\mathbf{P}X)_u] \xrightarrow{\bar{h}_u} B\, [\![u]\!]$. In summary:

Theorem 4.5. *Given a homomorphism graph G and a* $\mathrm{V}_G \times \mathbf{sort}$*-indexed set X of variables, the frex* $G[X]$ *is given by* $G[X]\, [\![u]\!] \coloneqq G\, [\![u]\!]\, [(\mathbf{P}X)_u]$ *on vertices and* $G[X]\, [\![e]\!]$ *on edges. The homomorphism* $\mathrm{sta} : G \to G[X]$ *is given by the homomorphism* $\mathrm{sta}_{u_s} : a \mapsto \mathrm{sta}_s$ *of the vertex-wise frex, and the function* $\mathrm{dyn}_{u_s} : X\, [\![(u, s)]\!] \to G[X]\, [\![u_s]\!]$ *sends each* $x \in X\, [\![s]\!]$ *to* $\mathrm{dyn}_s(u, \mathrm{id}, s, x)$*. We get the unique extension homomorphism* $[h; e] : G[X] \to A$ *vertex-wise.*

Example 4.6. This theorem applies to our first example, namely extending exponentiation with the indexed-set of variables $X \coloneqq \{x, y : \mathbf{integer}, u : \mathbf{rational}\}$ and partially evaluating the partially-static expression of sort **rational**:

$$5^{2\cdot(3\cdot x+y)} \cdot 3^{3\cdot y+2\cdot x} \cdot u^2 \cdot 5^{8\cdot y}$$

It evaluates to the bag:

$$\left\lbag \begin{array}{l} (1, ((3^{-}, (2\cdot)), x))\,, (1, ((5^{-}, (2\cdot), (3\cdot)), x))\,, \\ (1, ((3^{-}, (3\cdot)), y)), (1, ((5^{-}, (2\cdot)), y)), (1, ((5^{-}, (8\cdot)), y)), \\ (2, ((), u)) \end{array} \right\rbag$$

representing the normalised expression:

$$3^{2\cdot x} \cdot 5^{2\cdot(3\cdot x)} \cdot 3^{3\cdot y} \cdot 5^{2\cdot y} \cdot 5^{8\cdot y} \cdot u^2$$

As we can see, the combined frex takes into account the homomorphism equations. But it is also oblivious to the higher-order structure of the homomorphisms, and does *not* simplify the sequences of composed homomorphisms. The map-fusion and map-reduce examples are similarly limited.

5 Conclusion and prospects

We have shown how to reuse free extensions of component theories to construct free extensions of combined theories. In ongoing work, we investigate similarly modular constructions for varieties of semirings, i.e. algebras with multiplication that distributes over addition. In this work-in-progress, we decompose the construction of polynomial semirings into a construction that combines universal constructions for varieties of monoids. It is substantially more technically involved than the constructions we have presented so far, utilising the more abstract notions of operads and multi-categories.

Implicit in this presentation is our use of 2-dimensional category theory, which we have omitted to improve the accessibility of this manuscript. All 3 modular constructions involve certain 2-functors between (strict) 2-categories. In our first construction (S3), we use a 2-functor from Jacobs [2021] that sends the category of algebras together with an involutive structure over it to its category of involutive algebras. In our second construction (S4), we use a 2-functor that sends a category of algebras to its category of homomorphism graphs. For semiring varieties, we use a 2-functor that sends a multi-category to a category of operadic algebras in this multi-category. The 2-dimensional structures provide a high-level interface for transporting uniform universal properties *qua* adjunctions, yielding conceptual proofs.

The development so far is theoretical, establishing the universal properties behind the intended data-structures and normalisation procedures. We will implement these procedures and their applications to partial evaluation [Yallop et al. 2018] and proof synthesis [Allais et al. 2022].

Another modularity axis we are exploring concerns the combination of first-order free-extensions with higher-order functions [Corbyn et al. 2022]. Concretely, we develop a generic normalisation-by-evaluation algorithm for a higher-order language with a base-type obeying equational laws. The algorithm uses only the frex interface, and allows us to leverage frex evaluators for first-order structures to partial evaluators for higher-order structures. We expect such a development to mesh well with the current work, and provide better partial evaluators for inherently higher-order use-cases such as our map-reduce examples.

Acknowledgments

We thank Edwin Brady, Jacques Carette, Marcelo Fiore, James McKinna, and the anonymous reviewers for useful comments and discussions. Supported by the Engineering and Physical Sciences Research Council grant EP/T007265/1 and an Industrial CASE Studentship, a Royal Society University Research Fellowship, a Facebook Research Award, an Alan Turing Institute seed-funding grant, a UKRI Future Leaders Fellowship MR/T043830/1 (EHOP), a travel grant by the Ericsson Research Foundation, and the Swedish Foundation for Strategic Research (SSF) under the project Octopi (Ref. RIT17-0023).

References

Guillaume Allais, Edwin Brady, Nathan Corbyn, Ohad Kammar, and Jeremy Yallop. 2022. Frex: dependently-typed algebraic simplification. (2022).

Jacques Carette and Oleg Kiselyov. 2011. Multi-stage programming with functors and monads: Eliminating abstraction overhead from generic code. *Sci. Comput. Program.* 76, 5 (2011), 349–375. https://doi.org/10.1016/j.scico.2008.09.008

Jacques Carette, Oleg Kiselyov, and Chung-chieh Shan. 2009. Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages. *J. Funct. Program.* 19, 5 (2009), 509–543. https://doi.org/10.1017/S0956796809007205

Nathan Corbyn, Ohad Kammar, Sam Lindley, Nachiappan Valliappan, and Jeremy Yallop. 2022. Normalization by Evaluation with Free Extensions. (2022). The 7th ACM SIGPLAN International Workshop on Type-Driven Development (TyDe'2022).

Martin Hyland and John Power. 2006. Discrete Lawvere theories and computational effects. *Theoretical Computer Science* 366, 1 (2006), 144–162. https://doi.org/10.1016/j.tcs.2006.07.007 Algebra and Coalgebra in Computer Science.

Bart Jacobs. 2021. Involutive Categories and Monoids, with a GNS-Correspondence. *Foundations of Physics* 42 (2021), 874–895. Issue 7. https://doi.org/10.1007/s10701-011-9595-7

Oleg Kiselyov, Kedar N. Swadi, and Walid Taha. 2004. A methodology for generating verified combinatorial circuits. In *EMSOFT 2004, September 27-29, 2004, Pisa, Italy, Fourth ACM International Conference On Embedded Software, Proceedings*, Giorgio C. Buttazzo (Ed.). ACM, 249–258. https://doi.org/10.1145/1017753.1017794

Torben Ægidius Mogensen. 1988. Partially Static Structures in a Self-Applicable Partial Evaluator. In *Partial Evaluation and Mixed Computation*, D. Bjørner, A.P. Ershov, and N.D. Jones (Eds.).

Tiark Rompf, Arvind K. Sujeeth, Nada Amin, Kevin J. Brown, Vojin Jovanovic, HyoukJoong Lee, Manohar Jonnalagedda, Kunle Olukotun, and Martin Odersky. 2013. Optimizing data structures in high-level programs: new directions for extensible compilers based on staging. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, Roberto Giacobazzi and Radhia Cousot (Eds.). ACM, 497–510. https://doi.org/10.1145/2429069.2429128

Jeremy Yallop. 2017. Staged generic programming. *Proc. ACM Program. Lang.* 1, ICFP (2017), 29:1–29:29. https://doi.org/10.1145/3110273

Jeremy Yallop, Tamara von Glehn, and Ohad Kammar. 2018. Partially-Static Data as Free Extension of Algebras. *Proc. ACM Program. Lang.* 2, ICFP, Article 100 (July 2018), 30 pages. https://doi.org/10.1145/3236795