Modular abstract syntax trees (MAST): substitution tensors with second-class sorts[★]

Marcelo P. Fiore[a,1] Ohad Kammar[b,2] Georg Moser[c,3] Sam Staton[d,4]

[a] *Department of Computer Science and Technology*
University of Cambridge
England

[b] *Laboratory for Foundations of Computer Science*
School of Informatics
University of Edinburgh
Scotland

[c] *Theoretical Computer Science*
Department of Computer Science
University of Innsbruck
Austria

[d] *Department of Computer Science*
University of Oxford
England

Abstract

We adapt Fiore, Plotkin, and Turi's treatment of abstract syntax with binding, substitution, and holes to account for languages with second-class sorts. These situations include programming calculi such as the Call-by-Value λ-calculus (CBV) and Levy's Call-by-Push-Value (CBPV). Prohibiting second-class sorts from appearing in variable contexts means the presheaf of variables is no longer a left-unit for Fiore et al's substitution tensor product. We generalise their development to associative and right-unital *skew* monoidal categories. We reuse much of the development through skew bicategorical arguments. We apply the resulting theory by proving substitution lemmata for varieties of CBV modularly.

Keywords: denotational semantics, presheaves, abstract syntax, binding, skew monoidal categories, skew bicategories, substitution lemma, holes, metaprogramming.

[★] This work was supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0038, ERC Grant BLAST, two ARIA SGAI TA1.1 grants, a Royal Society University Research Fellowship, as well as FWF Project AUTOSARD P 36623. For the purpose of Open Access the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. We thank the following people for interesting and useful discussions and suggestions: Bob Atkey; Greg Brown; Yotam Dvir; Nicola Gambino; Sergey Goncharov; M. Codrin Iftode; Meven Lennon-Bertrand; Paul B. Levy; Justus Matthiesen; Conor McBride; Sean K. Moss; Filip Sieczkowski; Stelios Tsampas; Zoe Stafford; Dmitrij Szamozvancev; and Jacob Walters.

[1] Email: marcelo.fiore@cl.cam.ac.uk

[2] Email: ohad.kammar@ed.ac.uk

[3] Email: georg.moser@uibk.ac.at

[4] Email: sam.staton@cs.ox.ac.uk

1 Introduction

Fiore, Plotkin, and Turi's [31] mathematical foundations for abstract syntax with binding possess several unique properties. It is based on Goguen et al.'s [39] initial algebra semantics, and as such provides an abstract interface that supports modularity and extensibility. It also accommodates both syntactic representations and their semantic models in one semantic setting. Its multi-sorted extension supports intrinsically-typed representation: the abstract syntax also encodes simply-typed constraints, so that only well-typed syntax trees can be expressed. It supports terms with context-aware *holes*, called *metavariables*. Despite its mathematical sophistication, it lends itself to formalisation and computational representation [6,7,24,20]. The abstract approach is robust to generalisation, e.g. to polymorphic [30], and dependently-typed [26] settings.

This manuscript concerns another such generalisation: support for second-class sorts [9], as employed by common calculi such as the Call-by-Value λ-calculus and Levy's Call-by-Push-Value [56,55]. Typically we consider separate syntax for values and for computations, but variables in the language only stand for values, leaving the sorts of computations 'second-class'. Supporting the syntactic needs of these calculi is essential for the applicability of this theory to formalisation and computational representation of programming languages. After all, common programming languages and their syntactic representations are overwhelmingly Call-by-Value.

We make these ideas and motivation precise through a comprehensive case-study of a Call-by-Value λ-calculus (CBV) with records/products, variants/coproducts, and a simple inductive type of natural numbers, with various flavours of recursion: structural, unbounded, and general recursion. Fig 1 presents the raw terms of this calculus. Each type A has *two* sorts of terms associated with it: values V and unrestricted terms M. We can embed values into unrestricted terms through the term constructor val, and use them wherever we may use a term. Values enjoy a first-class status w.r.t. binding: we only have value variables and may only substitute values for variables. This distinction partitions the abstract syntax into first-class sorts for values and second-class sorts for computations. This partition unfortunately invalidates the classical theory, which we will review in greater detail in the next section (Sec. 2). Our contribution is to modify this theory to allow the distinction between these two classes of sorts.

As a concrete yard-stick for this new theory, consider fragments of the calculus. For example, a fragment involving only functions and records, but not natural numbers. Each such fragment makes a different set of semantic demands on its class of models. For example, functions require certain exponentials, natural numbers require a parameterised natural number object, recursion requires parameterised fixed-points, etc. So long as we consider a fragment in isolation, we can simply aggregate all the required structure into one definition, define the semantic interpretation function, and establish its basic properties directly. However, once we consider multiple fragments simultaneously, we quickly need a modular representation of the syntax, its semantics, and their properties. A concrete example of such a property is the semantic substitution lemma, typically phrased as $[\![M[\theta]]\!] = [\![M]\!] \circ [\![\theta]\!]$. Without a modular theory encompassing both syntax and semantics, proving this lemma requires a separate tedious inductive argument for each fragment. The theory we develop here will allow us to prove it modularly for each fragment of the calculus, and combine these results together to $2^7 = 128$ different substitution lemmata for 128 different languages.

2 Overview

We recount the classical theory; explain why it needs extension; and summarise our contribution.

2.1 The classical theory

To make the discussion precise, consider the pure simply-typed λ-calculus (STLC) with holes. Given a set of base types $\beta \in \mathsf{Base}$, the *simple types* are inductively given by either base or function types: $\mathsf{SimpleType} \ni A, B, \ldots ::= \beta \mid A \to B$. Whatever formal theory for binding we choose, we want it to account for the following concepts. A *typing context* $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ associates binding occurrences of variables, x_i, to types A_i. A *hole*, a.k.a. a *metavariable* is an object-level representation of an unspecified program fragment, and so we specify holes $?m : A\,[\Gamma]$ by designating their type and which variables they are aware of. Given some holes $\mathbf{H}$, the λ-calculus's *well-typed terms with holes* from $\mathbf{H}$ are inductively

$$
\begin{array}{llll}
A, B, C ::= & & \text{type} & \\
 & \beta & & \text{base} \\
 | & A \to B & & \text{function} \\
 | & ⦇C_i : A_i \,|\, i \in I⦈ & & \text{record } (I \text{ finite}) \\
 | & \{\!|C_i : A_i \,|\, i \in I|\!\} & & \text{variant } (I \text{ finite}) \\
 | & \mathbb{N} & & \text{natural number}
\end{array}
\qquad
\begin{array}{llll}
V, W ::= & & \text{value} & \\
 & x & & \text{variable} \\
 | & \lambda x : A.M & & \text{function abstraction} \\
 | & (C_i : V_i \,|\, i \in I) & & \text{record constructor} \\
 | & A.C_i\ V & & \text{variant constructor} \\
 | & n & & \text{number literal}
\end{array}
$$

$$
\begin{array}{llll}
M, N, K, L ::= & & \text{term} & \\
 & \mathsf{val}\, V & & \text{value} \\
 | & \mathbf{let}\ x_1 = M_1; \ldots; x_n = M_n\ \mathbf{in}\ N & & \text{sequencing} \\
 | & M @ N & & \text{function application} \\
 | & (C_1 : M_1, \ldots, C_n : M_n) & & \text{record constructor} \\
 | & \mathbf{case}\ M\ \mathbf{of}\ (C_1 x_1, \ldots, C_n x_n) \Rightarrow N & & \text{record pattern match} \\
 | & A.C_i M & & \text{variant constructor} \\
 | & \mathbf{case}\ M\ \mathbf{of}\ \{C_i x_i \Rightarrow M_i \,|\, i \in I\}\ N & & \text{variant pattern match} \\
 | & \mathbf{unroll} M \mid \mathbf{roll} M & & \text{number (de)constructor} \\
 | & \mathbf{fold}\ M\ \mathbf{by}\ \{x \Rightarrow N\} & & \text{bounded iteration} \\
 | & \mathbf{for}\ i = M\ \mathbf{do}\ N & & \text{unbounded iteration} \\
 | & \mathbf{let\ rec}\ f_1 \Gamma_1 : A_1 = M_1; \ldots; f_n \Gamma_n : A_n = M_n\ \mathbf{in}\ N & & \text{recursion}
\end{array}
$$

Fig. 1. Syntax of CBV

generated by the rules for application, abstraction, variables, and holes:

$$
\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M @ N : B}\ \text{app}
\qquad
\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \to B}\ \text{abs}
\qquad
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ \text{var}
\qquad
\frac{(?m : A[\Delta]) \in \mathbf{H} \qquad \text{for all } (y : B) \in \Delta\text{: } \Gamma \vdash M_y : B}{\Gamma \vdash ?m\left[M_y\right]_{y \in \Delta} : A}\ \text{hole}
$$

A foundation for binding must express that the left-most occurence of x in $\lambda x : A.M$ is *binding* and its scope is M, and that none of the remaining syntactic constructs modify the scope. Moreover, variables and metavariables are common to all such languages, and only the language-specific constructs—abstraction and application in our setting—need to be specified. These concepts are natural and standard, though holes are less familiar. They surface naturally during programming language implementation; during formal metatheory development, e.g. when defining contextual equivalence for a calculus; or during metaprogramming when splicing object representations of code.

The classical theory observes that the representation of syntax-with-binding and its models are objects in a category of presheaves that we will later call STLC-*structures*. These STLC-structures consist of sets $P_A\Gamma$ indexed by the types $A \in \mathsf{SimpleType}$ and contexts $\Gamma \in \text{STLC}_{\vdash}$ and equipped with a suitable functorial *variable renaming* action. Objects in and constructions over this category include structure relevant for specifying abstract syntax with binding. For example, the presheaf of *variables* is given by the sets $\mathbb{V}_A\Gamma \coloneqq \{x | (x : A) \in \Gamma\}$. It provides semantic constants/literals representing free and bound variables.

We can form STLC-structures for both the abstract syntax $\mathbb{\Lambda}$ and its denotational semantics $\mathbf{M}$ in a locally-small Cartesian-closed category $\mathcal{C}$ with chosen base-type interpretations, products and exponentials:

$$
\mathbb{\Lambda}_A\Gamma \coloneqq \{M | \Gamma \vdash M : A\} \qquad \mathbf{M}_A\Gamma \coloneqq \mathcal{C}(\llbracket\Gamma\rrbracket, \llbracket A\rrbracket) \quad \text{where: } \llbracket A \to B\rrbracket \coloneqq \llbracket B\rrbracket^{\llbracket A\rrbracket} \quad \llbracket\Gamma\rrbracket \coloneqq \prod\nolimits_{(x:A)\in\Gamma} \llbracket A\rrbracket
$$

The syntactic and semantic interpretations of the two language-specific constructs—abstraction and application—equip them with an algebra structure for the following *signature* functor:

$$\mathbf{O} : \text{STLC-}\mathbf{Struct} \to \text{STLC-}\mathbf{Struct} \quad (\mathbf{O}X)_A\Gamma \coloneqq \left(\coprod_{B\in\mathsf{SimpleType}} X_{B\to A} \times X_B\right) \amalg \left(\coprod_{A=B\to C} X_C(\Gamma, x : B)\right)$$

Indeed, for the syntax presheaf $\mathbb{\Lambda}$, the $\mathbf{O}$-algebra $\mathbb{\Lambda}\,[\![-]\!] : \mathbf{O}\mathbb{\Lambda} \to \mathbb{\Lambda}$ is given by the syntactic constructs:

$$\mathbb{\Lambda}\,\left[\!\left[(M, N) \in (\mathbb{\Lambda}_{B\to A}\Gamma) \times (\mathbb{\Lambda}_B\Gamma)\right]\!\right]_{A;\Gamma} \coloneqq M\,@\,N \qquad \mathbb{\Lambda}\,\left[\!\left[M \in \mathbb{\Lambda}_C(\Gamma, x : B)\right]\!\right]_{A;\Gamma} \coloneqq \lambda x : B.M$$

For the semantics presheaf, this structure is given by their interpretation through evaluation and currying:

$$\mathbf{M}\left[\!\!\left[\left([\![\Gamma]\!] \xrightarrow{f} [\![A]\!]^{[\![B]\!]}, [\![\Gamma]\!] \xrightarrow{a} [\![B]\!]\right) \in \mathbf{M}_{B\to A}\Gamma \times \mathbf{M}_B\Gamma\right]\!\!\right]_{A;\Gamma} \coloneqq [\![\Gamma]\!] \xrightarrow{(f,a)} [\![A]\!]^{[\![B]\!]} \times [\![B]\!] \xrightarrow{\mathsf{eval}} [\![A]\!]$$

$$\mathbf{M}\left[\!\!\left[[\![\Gamma, x : B]\!] \cong [\![\Gamma]\!] \times [\![B]\!] \xrightarrow{M} [\![C]\!] \in \mathbf{M}_C(\Gamma, x : B)\right]\!\!\right]_{A;\Gamma} \coloneqq [\![\Gamma]\!] \xrightarrow{\mathsf{curry}\,M} [\![C]\!]^B$$

Whatever the signature, we equip the syntax and model presheaves with an interpretation for variables:

$$\mathsf{var} : \mathbb{V} \to \mathbb{\Lambda} \qquad \mathsf{var}_{A;\Gamma}\, x \coloneqq x \qquad \mathsf{env} : \mathbb{V} \to \mathbf{M} \qquad \mathsf{env}_{A;\Gamma}\, x \coloneqq [\![\Gamma]\!] = \prod_{(y:B)\in\Gamma} [\![B]\!] \xrightarrow{\pi_x} [\![A]\!]$$

These observations characterise the presheaf of abstract syntax up-to α-equivalence involving application, abstraction, and variables as the initial algebra $\mathbb{\Lambda} \coloneqq \mu Z.(\mathbf{O}Z) \amalg \mathbb{V}$. The classical theory goes beyond this observation and also accounts for substitution. To axiomatise substitution, it defines the *substitution tensor product* $(\circledcirc) : (\text{STLC-}\mathbf{Struct})^2 \to \text{STLC-}\mathbf{Struct}$:

$$(P \circledcirc Q)_A\Gamma \coloneqq \int^{\Delta\in\text{STLC-}\mathbf{Struct}} P_A\Delta \times \prod_{(x:B)\in\Delta} Q_B\Gamma \coloneqq \left(\coprod_{\Delta\in\text{STLC-}\mathbf{Struct}} P_A\Delta \times \prod_{(x:B)\in\Delta} Q_B\Gamma\right) / \sim$$

Each element in the quotient $[t, e]_\Delta$ represents a value from P_A containing variables from Δ, together with an environment of Q-values in context Γ. Capture avoiding substitution becomes an operation on the syntactic presheaf:

$$-[-] : \mathbb{\Lambda} \circledcirc \mathbb{\Lambda} \to \mathbb{\Lambda} \qquad [t, \theta]_\Gamma \mapsto t\,[\theta]$$

The coend's quotient $(\sim)$ amounts to stability under weakening and substituting equals-for-equals:

$$M\,[x] = M \quad (x \text{ does not appear in } M) \qquad (M\,[x \mapsto z, y \mapsto z])\,[z \mapsto N] = M\,[x \mapsto N, y \mapsto N]$$

Importantly, the semantics presheaf possesses the same type of structure, given by composition:

$$-[-] : \mathbf{M} \circledcirc \mathbf{M} \to \mathbf{M} \qquad \left([\![\Delta]\!] \xrightarrow{f} [\![A]\!]\right) \left[\prod_{(y:B)\in\Delta} \left([\![\Gamma]\!] \xrightarrow{e_y} [\![B]\!]\right)\right] \coloneqq [\![\Gamma]\!] \xrightarrow{(e_y)_y} \prod_{y:A} [\![A]\!] = [\![\Delta]\!] \xrightarrow{f} [\![A]\!]$$

The presheaf of variables $\mathbb{V}$ and the tensor $(\circledcirc)$ form a monoidal category, and both syntax and semantics presheaves become monoids through the interpretation of variables and substitution, suitably compatible with the signature functor $\mathbf{O}$ in a structure dubbed an $\mathbf{O}$-*monoid.*

The capstone of this theory is the following representation theorem. It characterises $\mathbb{\Lambda}$ as the initial $\mathbf{O}$-monoid, with the denotational semantics being the unique $\mathbf{O}$-monoid homomorphism $[\![-]\!] : \mathbb{\Lambda} \to \mathbf{M}$. An immediate consequence of this representation theorem is the standard substitution lemma for denotational semantics. While substitution lemmata are not hard to prove, they are tedious to establish formally. The classical theory's universal characterisation justifies omitting them from most technical developments:

Lemma 2.1 (substitution) *For every term $\Delta \vdash M : A$ and substitution $\left(\Gamma \vdash \theta_y : B\right)_{(y:B)\in\Delta}$, we have:*

$$[\![M\,[\theta]]\!] = [\![M]\!] \circ \left([\![\theta_y]\!]\right)_y$$

Proof. By the homomorphism property of the denotational semantics:

$$[\![M\,[\theta]]\!] = [\![(-[-])[M,\theta]_\Delta]\!] = (-[-])\left[[\![M]\!]\,,\left([\![\theta]\!]_y\right)_{y\in\Delta}\right]_\Delta = [\![M]\!]\circ\left([\![\theta_y]\!]\right)_y \qquad \square$$

Incorporating holes into this development is straightforward. Let $\mathbf{H}$ be an STLC-structure whose elements $?m \in \mathbf{H}_A\Gamma$ represent holes ($?m : A\,[\Gamma]$). Then $\mathbb{\Lambda}\mathbf{H} \coloneqq \mu Z.(\mathbf{O}Z) \amalg \mathbb{V} \amalg (\mathbf{H} \circledcirc Z)$ represents the syntax with holes up-to α-equivalence and the structural identifications of the tensor product. The classical theory characterises $\mathbb{\Lambda}\mathbf{H}$ as the *free* $\mathbf{O}$-monoid $\mathbb{\Lambda}$ *over* $\mathbf{H}$, i.e., equipped with a map $\mathsf{meta} : \mathbf{H} \to \mathbb{\Lambda}\mathbf{H}$. This characterisation similarly implies a corresponding substitution lemma, and moreover equips the syntax with a monad structure that coincides with metavariable subsitution/program splicing.

2.2 Abstract syntax for Call-by-Value

The Call-by-Value λ-calculus (CBV) isolates a syntactic fragment of terms—the *values*. In this calculus, its logic, and its semantics, we only need to substitute values for variables [56]. Therefore, formal treatment of CBV should distinguish between two sorts: $s \Coloneqq A \mid \mathsf{comp}\,A$. Values of type A are first-class sorts that can appear in contexts, and computations of type A are second-class sorts that cannot.

Values: $\dfrac{(x:A)\in\Gamma}{\Gamma\vdash x:A}\,\text{var}$ $\qquad\dfrac{\Gamma,x:A\vdash M:\mathsf{comp}\,B}{\Gamma\vdash \lambda x:A.M:A\to B}\,\text{abs}$ $\qquad\dfrac{(?m:s[\Delta])\in\mathbf{H}\qquad\text{for all }(x:A)\in\Delta\text{: }\Gamma\vdash M_x:s}{\Gamma\vdash ?m[M_x]_{x\in\Delta}:s}\,\text{hole}$

Computations: $\dfrac{\Gamma\vdash V:A}{\Gamma\vdash \mathsf{val}\,V:\mathsf{comp}\,A}\,\text{val}$ $\qquad\dfrac{\Gamma\vdash M:\mathsf{comp}(A\to B)\qquad\Gamma\vdash N:\mathsf{comp}\,A}{\Gamma\vdash M\,@\,N:\mathsf{comp}\,B}\,\text{app}$

Values include variables and functions, and embed as computations through an explicit coercion construct val. This construct is typically implicit in syntactic treatments of CBV, and instead one isolates the values as a fragment of the computations. Function values contain computations as subterms, and so the two syntactic classes are mutually inductive. One can include holes of either sort. A similar distinction between first-class and second-class sorts of terms appears in Levy's Call-by-Push-Value.

Defining concrete analogues to the instances of the classical theory for such calculi is straightforward [41, e.g.]. The CBV-structures are presheaves over both sorts (A and $\mathsf{comp}\,A$) and contexts of types Γ. We call these *heterogeneous* structures. Variables still form a presheaf $\mathbb{I}$, albeit a heterogeneous one, given for value sorts by $\mathbb{I}_A\Gamma = \{x|(x : A) \in \Gamma\}$ and empty for computation sorts $\mathbb{I}_{\mathsf{comp}\,A}\Gamma = \emptyset$. This irregularity propagates throughout the technical development. It is straightforward to define an adapted substitution tensor ($\otimes$), but its abstract properties change—it is no longer monoidal, since typically $\mathbb{I} \otimes P \ncong P$. The reason is technical. In STLC sorting, the inverse to the monoidal unitor $\ell^{-1} : P \to \mathbb{V} \circledcirc P$ sends $p \in P_A[]$ to the equivalence class represented by $(x, x \mapsto p)$ at $[x : A]$. In CBV sorting, when $p \in P_{\mathsf{comp}\,A}[]$ is a computation, we can never have a context containing a variable $x : \mathsf{comp}\,A$. The inverse does not exist. We must formulate $\mathbf{O}$-monoids differently in order to capture the universal property of abstract syntax.

2.3 Contribution

Adapting the theory is straightforward by noticing that, while ($\otimes$) is not monoidal, it is *right-skew* monoidal [72]. Moreover, it is *right-unital*: $P \otimes \mathbb{I} \cong P$; and *associative*: $(P \otimes Q) \otimes L \cong P \otimes (Q \otimes L)$. We realise this observation, generalise the classical theory to account for calculi with second-class sorts, and dub the resulting theory *Modular Abstract Syntax Trees* (MAST). This simple change propagates throughout the theory and requires reformulating it from start to finish. Fortunately, two relatively recent developments make our work straightforward.

Skew bicategorical foundations. Fiore, Gambino, Hyland and Winskel [28] characterised the substitution tensor product ($\circledcirc$) as 1-cell composition in a Kleisli bicategory over *profunctors*. While technical, this bicategory captures the structure of ($\circledcirc$) as it operates on heterogeneously sorted structures: one set of sorts for syntactic classes, and one set of sorts for the context of bound variables. We use it to define a skew bicategorical account for second-class sorts, avoiding the many calculations direct treatments incur.

Skew monoidal theory of abstract syntax. Fiore and Szamozvancev [24,25] recently developed an abstract syntax based on a skew monoidal structure. Their account reformulates the classical theory in terms of homogeneous *families*, i.e., sort-and-context indexed sets without a given functorial action. These families have an associated tensor without a quotient $(P \oslash Q)_s\Gamma \coloneqq \coprod_\Delta P_s\Delta \times \prod_{(x:r)\in\Delta} Q_r\Gamma$. The theory compensates for the missing quotient by demanding additional axioms for substitution. Their tensor product $(\oslash)$, like our tensor product $(\otimes)$, is skew, and it is neither unital nor associative. Although skewness is inherent to their development, its source is different to ours. Nonetheless, we reuse some of their results verbatim thanks to our shared categorical abstractions.

Evaluation. To evaluate MAST, we study fragments of the CBV calculus in Fig 1. MAST's modularity allows us to formulate succinctly their syntax and semantics, deriving 128 substitution lemmata.

Alternatives. Instead of re-developing the theory, in practice one can take a concrete semantic structure $\mathbf{M}$ with second-class sorts, and complete it to obtain a model $\overline{\mathbf{M}}$ of the classical theory, in which the second-order sorts can appear in contexts. We are not interested in this alternative. First, it leads to unnecessarily complicated models (i.e., the hypothetical $\overline{\mathbf{M}}$) which contain formal/syntactic extensions needed to account for variables of second-class sorts that programs will never exhibit. Second, we would expect proponents of this alternative to prove this kind of completion is always possible. Doing so would require to either empirically complete all known models, or, more satisfyingly, define a completion construction $\mathbf{M} \mapsto \overline{\mathbf{M}}$. In this work, we define models for second-class sorts, i.e., the structure for $\mathbf{M}$, and moreover show they have the same abstract theory as those with first-class sorts. This development lays the foundation for such wholesale completion construction $\mathbf{M} \mapsto \overline{\mathbf{M}}$, which we leave to further work.

Paper structure. While the theory is technical, applying it concretely involves few self-contained ingredients. Sections 3–7 cover these ingredients in a tutorial style. We accompany each concept with motivating explanations and concrete examples (more than 25 examples overall). This development culminates in the Special Representation Thm 7.4 which characterises the presheaf of abstract syntax with holes in terms of $\mathbf{O}$-monoids in our more general skew setting. This theorem enables us to prove dozens of substitution lemmata using the same reasoning as Lemma 2.1. Sec. 8 reports on a case study: extensions to CBV. For space constraints, we relegate the details to Appendix A, as they follow the same structure as the examples in the tutorial. Sec. 9 surveys closely related work. Sec. 10 concludes.

Appendix B provides a detailed technical outline of the main ingredients in the bicategorical development and presents the General Representation Thm B.2 for the skew structure. Thm B.2 implies the Special Representation Thm 7.4 directly, but abstracts away from the concrete syntactic details and technicalities of presheaves using skew monoidal products and actions. Appendices C–E provide full details.

3 Heterogeneous sorting systems and structures

The first component in the theory specifies the available sort of syntactic classes one is interested in, designating some sorts as *first-class*: they can appear in context and binding positions. A *heterogeneous sorting system* $\mathbf{R} = \left(\mathbf{R}\text{-}\mathsf{sort}, (\vDash_{\mathbf{R}} \mathsf{bnd})\right)$ consists of:

- a set $\mathbf{R}\text{-}\mathsf{sort}$ whose elements are the *sorts*;
- a predicate $(\vDash_{\mathbf{R}} \mathsf{bnd})$ over $\mathbf{R}$ sorts singling out the *bindable* sorts $s \vDash_{\mathbf{R}} \mathsf{bnd}$.

Only the bindable sorts can appear in binding occurrences and contexts, letting $\mathbf{R}\text{-}\mathsf{Bind} \coloneqq \left\{s \middle| s \vDash_{\mathbf{R}} \mathsf{bnd}\right\}$. We omit the sorting system $\mathbf{R}$ and write sort, $s \vDash \mathsf{bnd}$, and Bind when $\mathbf{R}$ is unambiguous. A *homogeneous* sorting system is a sorting system in which all sorts are bindable. Thus our account generalises the classical theory by moving from homogeneous sorting systems to heterogeneous sorting systems.

Example 3.1 The sorting system CBN for the *call-by-name* λ-calculus has the simple types as bindable sorts, i.e., $\text{CBN-}\mathsf{sort} = \mathsf{SimpleType}$, and for all $A \in \mathsf{SimpleType}$ we assign $A \vDash_{\text{CBN}} \mathsf{bnd}$. The sorting system CBV for the *call-by-value* λ-calculus has two sorts for each simple type $A \in \mathsf{SimpleType}$:

- a bindable value sort $A \vDash_{\text{CBV}} \mathsf{bnd}$; and
- a non-bindable computation sort $\mathsf{comp}\,A \nvDash_{\text{CBV}} \mathsf{bnd}$.

These sorting systems codify that in CBV only values may be substituted for variables, and CBN does not make this distinction and all expressions can be substituted for variables.

Example 3.2 Each sorting system $\mathbf{R}$ restricts to a homogeneous system $\mathbf{R}|_{\mathsf{bnd}} \coloneqq \left(\mathbf{R}\text{-}\mathsf{Bind}, (\vDash_{\mathbf{R}} \mathsf{bnd})\right)$.

Every set of bindable sorts determines a category of contexts and renamings in the following way. Given a set S, we write $S_{\vdash} \coloneqq \mathsf{List}\,S$ for the set of finite sequences $\Gamma \in S_{\vdash}$ which we call the *S-sorted contexts*. We will need to refer to positions in a given context. For example, the third position in the context $\Gamma \coloneqq [A, A \to B, B]$ has sort B. To simplify such references, we will label these positions with meta-level variable names. For example, we will write $\Gamma \coloneqq x : A, f : A \to B, y : B$, and refer to the third position by y. This presentation makes it seem as if we are using a nominal representation of binding, whereas in fact we are using a nameless (i.e., de Bruijn [21]) representation, indexing positions in the context by ordinals. We will therefore refer to the set $\mathbb{V}\Gamma$ of positions in a given context Γ as the set of its *variables*, and will use the meta-level labels to refer to its elements. We write $(x : a) \in \Gamma$ when the element a appears in position $x \in \mathbb{V}\Gamma$.

A *renaming* $\rho : \Gamma \to \Delta$ is a function $-[\rho] : \mathbb{V}\Gamma \leftarrow \mathbb{V}\Delta$ satisfying $(y[\rho] : s) \in \Gamma \impliedby (y : s) \in \Delta$. The choice of direction ($\underline{\text{from}}$ Γ $\underline{\text{to}}$ Δ) is a matter of taste. One mnemonic for our choice is the fictional typing judgement $\Gamma \vdash \rho : \Delta$. Renamings compose as functions in opposite order with the identity function acting as the identity renaming, and collect into a category $S_{\vdash}$ of *contexts* and *renamings* between them.

Example 3.3 The following is a renaming from the the context to itself:

$$\rho : \Gamma \coloneqq [x : \beta_1, f : \beta_1 \to \beta_2, g : \beta_1 \to \beta_2, y : \beta_1] \to \Gamma \qquad x[\rho] \coloneqq x \quad f[\rho] \coloneqq g \quad g[\rho] \coloneqq f \quad y[\rho] \coloneqq x$$

Thus a renaming may permute the order, e.g., permuting the variables in positions f and g, identify some variables, e.g., x and y, or weaken the context, e.g., y is not in the image.

Example 3.4 Each category $S_{\vdash}$ has finite products. The terminal object is the empty context $\mathbb{1} \coloneqq []$ with the unique renaming $(\_) : \Gamma \to \mathbb{1}$ given vacuously by the empty function $\mathbb{V}\Gamma \leftarrow \mathbb{V}[]$. The product of two contexts is their concatenation, with the left/right projection sending the i^{th} position from the left/right to the i^{th} position to the left/right. Letting $\Gamma \coloneqq [x_1 : s_1, \ldots, x_n : s_n]$ and $\Delta \coloneqq [x_{n+1} : s_{n+1}, \ldots, x_{n+m} : s_{n+m}]$:

$$\Gamma +\!\!\!+ \Delta \coloneqq [x_1 : s_1, \ldots, x_{n+m} : s_{n+m}] \qquad \Gamma \xleftarrow{x_i[\boldsymbol{\pi}_1] :=x_i} \Gamma +\!\!\!+ \Delta \xrightarrow{x_{n+i} =: x_i[\boldsymbol{\pi}_2]} \Delta$$

Let $\mathbf{R}$ be a sorting system. Consider the set $\mathbf{R}\text{-}\mathsf{sort}$ as a discrete category. An *$\mathbf{R}$-structure* P is a presheaf $P \in \mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{sort} \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right)$, i.e., a functor $P : \mathbf{R}\text{-}\mathsf{sort} \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$ indexed by sorts and contexts and contravariant in renamings. If I is a set, an *I-family* F is a presheaf over the discrete category induced by I, and an $\mathbf{R}$-family is a family over $\mathbf{R}\text{-}\mathsf{sort} \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}$. Thus an $\mathbf{R}$-family F assigns to each sort $s \in \mathbf{R}\text{-}\mathsf{sort}$ and context $\Gamma \in \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}$ a set $F_s\Gamma$. An $\mathbf{R}$-structure P is an $\mathbf{R}$-family equipped with a functorial action:

$$-[\rho]_P : P_s\Gamma \leftarrow P_s\Delta \qquad p[\mathsf{id}]_P = p \qquad (q[\sigma])[\rho] = q[\sigma \circ \rho] \qquad (s \in \mathsf{sort}, p \in P_s\Gamma, q \in P_s\Xi, \text{ and } \Gamma \xrightarrow{\rho} \Delta \xrightarrow{\sigma} \Xi)$$

Let $\mathbf{R}\text{-}\mathbf{Struct}$ be the category of $\mathbf{R}$-structures and natural transformations $\alpha : P \to Q$ between them. This category is our central semantic domain. Through it we will define constructions and universal properties for most other concepts.

Example 3.5 Let $\mathbf{R}$ be a sorting system. The presheaf of *variables* $\mathbb{I} \in \mathbf{R}\text{-}\mathbf{Struct}$ is given by renaming:

$$\mathbb{I}_s\Gamma \coloneqq \{x | (x : s) \in \Gamma\} \qquad x[\rho]_{\mathbb{I}} \coloneqq x[\rho] \qquad \text{Note: } s \nvDash \mathsf{bnd} \implies \mathbb{I}_s\Gamma = \emptyset$$

When $\mathbf{R}$ happens to be homogeneous, we use the different notation $\mathbb{V} \coloneqq \mathbb{I}$ to signifty so.

Example 3.6 Let $\mathbb{\Lambda}^{\text{CBV}}$ be the CBV-structure comprising the values and terms of the CBV λ-calculus:

$$\mathbb{\Lambda}^{\text{CBV}}_{A}\Gamma \coloneqq \{V | \Gamma \vdash V : A\} \qquad \mathbb{\Lambda}^{\text{CBV}}_{\mathsf{comp}\,A}\Gamma \coloneqq \{M | \Gamma \vdash M : \mathsf{comp}\,A\} \qquad X[\rho]_{\mathbb{\Lambda}^{\text{CBV}}} \coloneqq X[\rho]$$

Its functorial action is given by the usual, syntactic, definition of renaming.

Example 3.7 Let P be an $\mathbf{R}$-structure. Define the $\mathbf{R}|_{\mathsf{bnd}}$-structure by restriction, so that $\mathbb{I}|_{\mathsf{bnd}} = \mathbb{V}$:

$$P|_{\mathsf{bnd}} \in \mathbf{PSh}\left(\mathsf{Bind} \times \mathsf{Bind}_{\vdash}\right) \qquad (P|_{\mathsf{bnd}})_s\Gamma \coloneqq P_s\Gamma$$

Restriction plays a central role, allowing us to relate MAST to its classical counterpart.

Let C be a category with chosen finite products. Every functor $F : S \to C$ extends to a product-preserving functor $F^{\mathrm{Env}} : S_{\vdash} \to C$ via:

$$F^{\mathrm{Env}}_{\Gamma} := \prod_{(x:s)\in\Gamma} F_s \qquad F^{\mathrm{Env}}_{\rho} := \left(F^{\mathrm{Env}}_{\Gamma} \xrightarrow{\boldsymbol{\pi}_{x[\rho]}} F_s\right)_{(x:s)\in\Delta} : F^{\mathrm{Env}}_{\Gamma} \to F^{\mathrm{Env}}_{\Delta} \qquad (\rho : \Gamma \to \Delta)$$

The product-preservation condition provides the *concatenation* operation $(+\!\!\!+) : P^{\mathrm{Env}}_{\Gamma} \times P^{\mathrm{Env}}_{\Delta} \to P^{\mathrm{Env}}_{\Gamma +\!\!\!+ \Delta}$. The typical case is $C := \mathbf{PSh}\,(\mathbf{R}\text{-}\mathsf{Bind})$. By considering each $P \in \mathbf{R\text{-}Struct}$ as a functor $\mathbf{R}\text{-}\mathsf{sort} \to \mathbf{PSh}\,(\mathbf{R}\text{-}\mathsf{Bind})$, we form the presheaf $P^{\mathrm{Env}} \in \mathbf{PSh}\,(\mathbf{R}\text{-}\mathsf{Bind})$. We call the elements $e \in P^{\mathrm{Env}}_{\Gamma}\Delta$, the *$P$-valued Γ-environments* in context Δ. They are are Γ-indexed tuples of $P\Delta$ elements of the appropriate sorts, and can represent both semantic and syntactic substitutions.

Example 3.8 The environment $(x, x, \lambda z : \beta.f\,@(f\,@\,z))$ is in $(\Lambda^{\mathrm{CBV}})^{\mathrm{Env}}_{[a:\beta,b:\beta,c:\beta\to\beta]}[x : \beta, f : \beta \to \beta]$.

Remark 3.9 We chose a nameless representation for contexts since it simplifies some concrete aspects in the development. As in the classical theory, this choice is not essential. For example, we can represent contexts nominally as a list pairing variable names and sorts. To define $\mathbb{V}$ and $\mathbb{I}$, we must disambiguate variables with the same concrete name. These different choices give equivalent small categories of contexts and therefore equivalent categories of presheaves. All of our concepts are defined by universal properties, and so the same development can be carried out in any of those representations.

4 Signature combinators

One defining feature of the classical theory is the way in which it breaks signatures with binding [3,63] into smaller components. Formally, and following categorical logic and Goguen's initial algebra approach to semantics, we use endofunctors $\mathbf{O} : \mathbf{R\text{-}Struct} \to \mathbf{R\text{-}Struct}$ to represent signatures. Each element $\mathrm{op} \in (\mathbf{O}X)_s\Gamma$ represents one of the language constructs of sort s that we can make with a collection of sub-terms labelled by the presheaf X in context Γ. The resulting representation enables some degree of modularity. For example, the coproduct of signature functors $\mathbf{O}_1 \amalg \mathbf{O}_2$ gives terms in which we can take operators from either $\mathbf{O}_1$ or $\mathbf{O}_2$. This decomposition allows us to study each operator on its own and combine their properties modularly.

Application, restriction, and extension

We often want to project out one or more subterms, or only define a language construct in a specific collection of sorts. For example, in the simplest case we project out or define in a single sort:

$$(@s_0) : \mathbf{R\text{-}Struct} \to \mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right) \qquad X\ @\ s_0 := X_{s_0} \qquad (s_0 \in \mathbf{R}\text{-}\mathsf{sort})$$

$$\underset{s_0}{\looparrowright} : \mathbf{R\text{-}Struct} \leftarrow \mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right) \qquad (\underset{s_0}{\looparrowright} Y)_s := \begin{cases} s = s_0 : & Y \\ \text{otherwise:} & \mathbb{0} \end{cases}$$

Example 4.1 The CBV inclusion of A-values into A-computations is: $\mathsf{ValOp}_A\, X := \looparrowright_{\mathsf{comp}\,A} X_A$.

In a more general form of these combinators we restrict to/extend from a subset of sorts:

$$(|_I) : \mathbf{R\text{-}Struct} \to \mathbf{PSh}\left(I \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right) \qquad (X|_I)_s := X_s \qquad (I \subseteq \mathbf{R}\text{-}\mathsf{sort})$$

$$\underset{I}{\looparrowright} : \mathbf{R\text{-}Struct} \leftarrow \mathbf{PSh}\left(I \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right) \qquad (\underset{I}{\looparrowright} Y)_s := \begin{cases} s \in I : & Y \\ \text{otherwise:} & \mathbb{0} \end{cases}$$

The two combinators are adjoint to each other $\looparrowright_I \dashv (|_I)$.

Example 4.2 Taking $I := \mathbf{R}\text{-}\mathsf{Bind}$ recovers the combinator $(|_{\mathsf{bnd}})$ from Ex. 3.7. We have that $\mathbb{I} = \looparrowright_{\mathsf{Bind}} \mathbb{V}$.

Products and coproducts

As in categorical logic and initial algebra semantics, the bread-and-butter combinators are products and coproducts. Products allow us to express n-ary syntactic constructs. Coproducts allow us to combine signatures into larger signatures. We will use both in many different settings, and so define them in their most generality as $\prod_I, \coprod_I : \mathcal{C}^I \to \mathcal{C}$, where $\mathcal{C}$ has I-indexed products or coproducts, as appropriate. We will define I through the following set-comprehension notation for the limiting/colimiting cones:

$$X_j \xrightarrow{C_j} \{\!\!\{ C_i : X_i \,\big|\, i \in I \}\!\!\} \text{ denotes } X_j \to \coprod_{i\in I} X_i \qquad (\!(C_i : X_i \,\big|\, i \in I)\!) \xrightarrow{(-_{C_j})} X_j \text{ denotes } \prod_{i\in I} X_i \to X_j$$

We will write $\left(C_i : v_i\right)_{i\in I}$ for the tuple containing v_i for the label C_i.

Example 4.3 Combine value inclusions in one functor $\mathsf{ValOp}\,X := \{\!\!\{ \mathsf{val}_A : \mathsf{ValOp}_A\,X \,\big|\, A \in \mathsf{SimpleType} \}\!\!\}$.

Example 4.4 Application in CBV has the signature:

$$\mathsf{AppOp}\,X := \left\{\!\!\left\{ (@) : \underset{\mathsf{comp}\,B}{\looparrowright} ((X \mathbin{@} \mathsf{comp}(A \to B)) \times (X \mathbin{@} \mathsf{comp}\,A)) \,\middle|\, A \in \mathsf{SimpleType} \right\}\!\!\right\}$$

Scope shift

We use the following scope shift operation to express operators that bind variables:

$$\Gamma\rhd : \textbf{R-Struct} \to \textbf{R-Struct} \qquad\qquad (\Gamma \rhd X)_s\Delta := X_s(\Delta + \Gamma) \qquad\qquad (\Gamma \in \mathbf{R}\text{-}\mathsf{Bind}_{\vdash})$$

Aside. In the classical single-sorted and homogeneous theory, scope shift by one variable is presheaf exponentiation by the presheaf of variables. In our setting, $(\Gamma\rhd) = (\mathbf{y}_\Gamma \multimap)$, where $\mathbf{y} : \mathbf{R}\text{-}\mathsf{Bind}_{\vdash} \to \mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right)$ is the Yoneda embedding and, for every single-sorted presheaf $G \in \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}$, the functor $(G \multimap) : \textbf{R-Struct} \to \textbf{R-Struct}$ is the right adjoint to $(G\odot) : \textbf{R-Struct} \to \textbf{R-Struct}$ given by: $(G \odot P)_s\Gamma := G\Gamma \times P_s\Gamma$.

Example 4.5 For abstraction: $\mathsf{LamOp} := \left\{\!\!\left\{ (\lambda x : A.) : \underset{A\to B}{\looparrowright} [x : A] \rhd X \mathbin{@} \mathsf{comp}\,B \,\middle|\, A, B \in \mathsf{SimpleType} \right\}\!\!\right\}$.

Combining these together, we have the full CBV signature: $\mathsf{CbvOps} := \mathsf{LamOp} \amalg \mathsf{ValOp} \amalg \mathsf{AppOp}$. The presheaf of syntax is then the initial algebra $\mathbb{\Lambda}^{\mathrm{CBV}} = \boldsymbol{\mu} X.\left((\mathsf{CbvOps}\,X) \amalg \mathbb{I}\right)$. Using the same methodology, one can mechanically translate, e.g., Aczel's [3,63] *binding signatures* which express a wide class of syntax.

5 Substitution tensors

The substitution tensor imposes semantic invariants on the input for syntactic or semantic substitution operations $-[-] : P \otimes Q \to L$ where P, Q, and L are $\mathbf{R}$-structures. Given an $\mathbf{R}$-structure P, consider the functor $P^{\mathrm{Env}} := \left(P|_{\mathsf{bnd}}\right)^{\mathrm{Env}} : \mathbf{R}\text{-}\mathsf{Bind}_{\vdash} \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$, i.e., its component sets $P^{\mathrm{Env}}_{\Delta}\Gamma := \prod_{(x:s)\in\Delta} P_s\Gamma$ are P-environments. We define the substitution tensor by:

$$(\otimes) : (\textbf{R-Struct})^2 \to \textbf{R-Struct} \qquad (P \otimes Q)_s\Gamma := \int^{\Delta\in\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}} P_s\Delta \times Q^{\mathrm{Env}}_{\Delta}\Gamma := \left(\coprod_{\Delta\in\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}} P_s\Delta \times Q^{\mathrm{Env}}_{\Delta}\Gamma\right)/(\sim)$$

The coend's quotienting relation $(\sim)$ is the least equivalence relation generated by relating the triples:

$$\left(\Delta_1, t[\rho]_P, e\right) \sim \left(\Delta_2, t, e_{-[\rho]}\right) \qquad\qquad (\rho : \Delta_1 \to \Delta_2,\ t \in P_s\Delta_2,\ e \in Q^{\mathrm{Env}}_{\Delta_1}\Gamma)$$

As we explained in the introduction, these identification represent invariants for substitution:

Example 5.1 Consider the syntax presheaf $\mathbb{\Lambda}^{\mathrm{CBV}}$. For brevity, we use the vernacular $(k\,y)$, rather then elaborate $((\mathsf{val}\,k)@(\mathsf{val}\,y))$, syntax. Consider the following equivalences in $\mathbb{\Lambda}^{\mathrm{CBV}} \otimes \mathbb{\Lambda}^{\mathrm{CBV}}$:

- Assigning the same value to different variables $(f, g$ below) relates to renaming the two variables to one $(f, g \mapsto h)$. Formally, taking $f[\rho] \coloneqq h, g[\rho] \coloneqq h$, $t \coloneqq \lambda x.f(g\,x)$, and $e \coloneqq (f : k\,y, g : k\,y)$ witnesses:

$$[\lambda x.f(g\,x), (f : k\,y, g : k\,y)]_{[f,g:\beta\to\beta]} = [\lambda x.h(h\,x), (h : k\,y)]_{[h:\beta\to\beta]} \in (\mathbb{\Lambda}^{\mathrm{CBV}} \otimes \mathbb{\Lambda}^{\mathrm{CBV}})_{\beta\to\beta}[k : \beta \to \beta, y : \beta]$$

- Weakening the context by unused variables relates to projecting only the used variables. Formally, taking $z[\rho] \coloneqq z$, $t \coloneqq \lambda x.z$, and $e \coloneqq (f : \lambda x.x, z : y)$ witnesses:

$$[\lambda x.z, (f : \lambda x.x, z : y)]_{[f:\beta\to\beta,z:\beta]} = [\lambda x.z, (z : y)]_{[z:\beta]} \qquad \in (\mathbb{\Lambda}^{\mathrm{CBV}} \otimes \mathbb{\Lambda}^{\mathrm{CBV}})_{\beta\to\beta}[y : \beta]$$

- Permuting the environment relates to permuting the variables. Formally, taking $f[\rho] \coloneqq g, g[\rho] \coloneqq f$, $t \coloneqq \lambda x.g(f\,x)$, and $e \coloneqq (f : \lambda x.x, g : k)$ witnesses:

$$[\lambda x.f(g\,x), (f : \lambda x.x, g : k)]_{[f,g:\beta\to\beta]} = [\lambda x.g(f\,x), (f : k, g : \lambda x.x)]_{[f,g:\beta\to\beta]} \in (\mathbb{\Lambda}^{\mathrm{CBV}} \otimes \mathbb{\Lambda}^{\mathrm{CBV}})_{\beta\to\beta}[k : \beta \to \beta]$$

These examples are representative in the following sense. We can represent every renaming $\rho : \Delta_0 \to \Delta_2$ as the composition: $\rho : \Delta_0 \overset{i}{\rightarrowtail} \Delta_1 \xrightarrow{\tau} \Delta_2$, where: $i : \Delta_0 \rightarrowtail \Delta_1$ is a renaming with a surjective action on variables, and τ is a *thinning*, a renaming with an injective and relative-order-preserving action on variables. Permuting the variables and then repeatedly identifying some, but not necessarily all, adjacent variables of the same sort generates all renamings with surjective actions. Repeatedly thinning out a variable generates all thinnings. Therefore $(\sim)$ is the smallest equivalence relation that contains the following three identifications (cf. Ex. 5.1):

- Identifying two variables vs. environments containing the same value in their positions.
- Weakening by a thinning vs. projecting according to a thinning.
- Permuting two variables in the term vs. permuting the values in their positions.

The substitution tensor product has the following *left/right unitors* and *associator* maps:

$$\begin{aligned}
&\boldsymbol{\ell} : \mathbb{I} \otimes P \to P && \mathbf{r} : P \otimes \mathbb{I} \xrightarrow{\cong} P && \mathbf{a} : (P \otimes Q) \otimes L \xrightarrow{\cong} P \otimes (Q \otimes L) \\
&\boldsymbol{\ell}[x, e]_\Delta \coloneqq e_x && \mathbf{r}[p, \rho]_\Delta \coloneqq p[\tilde{\rho}] && \mathbf{a}\left[[p, q]_{\Delta_1}, e\right]_{\Delta_2} \coloneqq \left[p, \left(\left[q_x, e\right]_{\Delta_2}\right)_{(x:s)\in\Delta_1}\right]_{\Delta_1}
\end{aligned}$$

The right unitor uses the natural isomorphism $\tilde{-} : \mathbb{I}^{\mathrm{Env}}_{\Delta}\Gamma \xrightarrow{\cong} \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}(\Gamma, \Delta)$. Only the right unitor $\mathbf{r}' \coloneqq \mathbf{r}^{-1}$ and associator $\mathbf{a}$ are nautral isomorphisms, while the left unitor $\boldsymbol{\ell}$ is not:

Example 5.2 Since $\mathbb{I}_{\mathsf{comp}\,\beta\to\beta}[] = \emptyset$, the following calculation implies $\boldsymbol{\ell} : \mathbb{I} \otimes \mathbb{\Lambda}^{\mathrm{CBV}} \to \mathbb{\Lambda}^{\mathrm{CBV}}$ isn't invertible:

$$(\mathbb{I} \otimes \mathbb{\Lambda}^{\mathrm{CBV}})_{\mathsf{comp}\,\beta\to\beta}[] = \emptyset \neq \mathbb{\Lambda}^{\mathrm{CBV}}_{\mathsf{comp}\,\beta\to\beta}[] \ni \mathsf{val}(\lambda x.x)$$

The following standard definition formulates the appropriate collection of properties the substitution tensor possesses. A *right-skew monoidal category* [72] $\big(\mathcal{C}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\big)$ consists of a category $\mathcal{C}$ equipped with a functor $(\otimes) : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, an object $\mathbb{I} \in \mathcal{C}$, and three transformations called the *associator*, natural in $a, b, c \in \mathcal{C}$ and the *left/right unitors*, natural in $a \in \mathcal{B}$:

$$\mathbf{a} : (a \otimes b) \otimes c \to a \otimes (b \otimes c) \qquad \boldsymbol{\ell} : \mathbb{I} \otimes a \to a \qquad \mathbf{r}' : a \to a \otimes \mathbb{I}$$

satisfying standard commutative equations, see Appendix C.1 for the full details. A skew category is *associative* when its associator is invertible, and *left/right-unital* when its corresponding unitor is invertible.[5]

Theorem 5.3 *Let* **R** *be a heterogeneous sorting system. The category of* **R***-structures equipped with the substitution tensor product, the presheaf of variables, and their associator and unitors is associative right-unital right-skew monoidal:*

$$\mathbf{R}\text{-}\mathbf{Struct} = \left(\mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{sort} \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}^{\mathsf{op}}\right), (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}^{-1}\right)$$

This tensor relates to its classical counterpart through restriction $(|_{\mathsf{bnd}}) : \mathbf{R}\text{-}\mathbf{Struct} \to \mathbf{R}\text{-}\mathsf{Bind}\text{-}\mathbf{Struct}$:

Lemma 5.4 *We have a coend-preserving isomorphism, natural in the* **R***-structures* $P, Q \in \mathbf{R}\text{-}\mathbf{Struct}$*:*

$$(P \otimes Q)|_{\mathsf{bnd}} \xrightarrow{\cong} (P|_{\mathsf{bnd}}) \olessthan (Q|_{\mathsf{bnd}})$$

This lemma and Ex. 3.7 mean restriction lifts the monoidal structure from homogeneous structures to heterogeneous structures. We can generally lift monoidal structure along *faithful* functors, and we will do so in §6. Unfortunately, $(|_{\mathsf{bnd}})$ is not faithful: it forgets how morphisms act on unbindable sorts. We thus need another proof for Thm 5.3, and the bicategorical development outlined in §B.1 delivers it smoothly.

We use the skew structure in the following definition. Let $\mathcal{C} = \left(\underline{\mathcal{C}}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$ be a right-skew monoidal category. A $\mathcal{C}$*-monoid* $\mathbf{M}$ is a triple $\left(\underline{\mathbf{M}}, \bullet, \mathsf{e}\right)$ consisting of an object $\underline{\mathbf{M}} \in \underline{\mathcal{C}}$ and two morphisms $\mathbb{I} \xrightarrow{\mathsf{e}} \underline{\mathbf{M}} \xleftarrow{(\bullet)} \underline{\mathbf{M}} \otimes \underline{\mathbf{M}}$ satisfying the following three equations:

$$\begin{tikzcd}
\mathbb{I} \otimes \underline{\mathbf{M}} \arrow[rr, "\mathsf{e} \otimes \mathsf{id}"] \arrow[dr, "\boldsymbol{\ell}"'] & \substack{\text{left}\\ \text{unit}\\ =} & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[dl, "(\bullet)"] \\
& \underline{\mathbf{M}} &
\end{tikzcd}$$

$$\begin{tikzcd}
& \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[dr, bend left, "(\bullet)"] & \\
(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \otimes \underline{\mathbf{M}} \arrow[ur, bend left, "(\bullet) \otimes \mathsf{id}"] \arrow[d, "\mathbf{a}"'] & \substack{\text{associativity}\\ =} & \underline{\mathbf{M}} \\
\underline{\mathbf{M}} \otimes (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[rr, bend right, "\mathsf{id} \otimes (\bullet)"'] & & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[u, "(\bullet)"']
\end{tikzcd}$$

$$\begin{tikzcd}
\underline{\mathbf{M}} \otimes \mathbb{I} \arrow[r, "\mathsf{id} \otimes \mathsf{e}"] & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[d, "(\bullet)"] \\
\underline{\mathbf{M}} \arrow[u, "\mathbf{r}'"] \arrow[r, equal] \arrow[ur, phantom, "\substack{\text{right}\\ \text{unit}\\ =}"] & \underline{\mathbf{M}}
\end{tikzcd}$$

We use the more suggestive notation $\mathbf{M} = \left(\underline{\mathbf{M}}, -[-], \mathsf{env}\right)$ for monoids in **R-Struct**, which we call *subsitution monoids*. For such monoids we only need to define the variable map on bindable sorts, and validate the left-unit axiom for bindable sorts. Indeed, for a non-bindable sort $s \nvDash \mathsf{bnd}$, we have that $\mathbb{I}_s = \mathbb{0}$ and $(\mathbb{I} \otimes \underline{\mathbf{M}})_s = \mathbb{0}$ and so there is a unique choice for var_s and the left-unit diagram commutes by initiality.

Example 5.5 The syntax presheaf Λ^{CBV} has a CBV-monoid structure given by capture-avoiding simultaneous substitution $[t, e]_\Gamma \mapsto t[e]$ and the inclusion of variables $\mathsf{var} : \mathbb{I} \xrightarrow{x \mapsto x} \Lambda^{\text{CBV}}$. This definition can be derived from the construction in the representation theorem (Thm 7.4).

Example 5.6 Let $\mathcal{C}$ be a Cartesian closed category with chosen finite products $\prod$ and exponentials b^a. Every choice of interpretation $[\![\beta]\!]$ for the base type equips $\mathcal{C}$ with a CBV-monoid structure by first extending the interpretation to types, and then defining the carrier presheaf by:

$$[\![A \to B]\!] \coloneqq [\![B]\!]^A \qquad [\![\Gamma]\!] \coloneqq \prod_{(x:A)\in\Gamma} [\![A]\!] \qquad \underline{\mathbf{M}}_A\Gamma \coloneqq \mathcal{C}([\![\Gamma]\!], [\![A]\!]) \qquad \underline{\mathbf{M}}_{\mathsf{comp}\,A}\Gamma \coloneqq \mathcal{C}([\![\Gamma]\!], [\![A]\!])$$

defining the unit by $\left(\mathsf{var}_{B;\Gamma}\, y\right) : [\![\Gamma]\!] = \prod_{(x:A)\in\Gamma} [\![A]\!] \xrightarrow{\boldsymbol{\pi}_y} [\![B]\!]$ and multiplication/substitution by:

$$([\![\Delta]\!] \xrightarrow{f} [\![A]\!])\left[([\![\Gamma]\!] \xrightarrow{\theta_y} [\![B]\!])_{(y:B)\in\Delta}\right] \coloneqq \left([\![\Gamma]\!] \xrightarrow{(\theta_y)_{(y:B)\in\Delta}} \prod\nolimits_{(y:B)\in\Delta} [\![B]\!] = [\![\Delta]\!] \xrightarrow{f} [\![A]\!]\right)$$

We explore richer models for CBV in §A.1.

[5] The terminology 'associative' and 'unital' is not standard, the other existing terminology we know of is Lack and Street [52], who call the 'unital' property '*normal*'.

6 Signature functors

The structure we specified so far, namely signature functors, their algebras, and **R-Struct**-monoids suffices to define the abstract syntax and its denotational semantics. However, it does not uniquely specify substitution nor establish a general substitution lemma. The missing ingredient is a structural map that specifies how to avoid unintended capture when moving under each operator in the signature. To do so, the classical theory uses pointed tensors, actions, and strengths, which we adapt to heterogeneous structures in this section.

To describe scope changes, Fiore et al. use a point-free way to reason about how a prehseaf encodes variables $[\![x : s]\!]_P \in P_s[x : s]$. Since $\mathbb{I}_s = \mathbf{y}_{[x:s]} \in \mathbf{PSh}\,(\mathbf{R}\text{-}\mathsf{Bind})$, the Yoneda lemma represents $\left([\![x : s]\!]_P\right)_{s \vDash \mathsf{bnd}}$ via a natural transformation:

$$\mathsf{env} : \mathbb{I} \to P \qquad \mathsf{env}_{s;\Gamma}\, x \coloneqq [\![x : s]\!]_P \left[[x : s] \xleftarrow{\pi_x} \Gamma\right] \in P_s\Gamma$$

Let $\mathcal{C} = \left(\underline{\mathcal{C}}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$ be a skew monoidal category. Recall the *coslice* category $\underline{\mathcal{C}}^{\bullet} \coloneqq \mathbb{I}/\underline{\mathcal{C}}$ given by:

- *pointed objects* A: pairs $\left(\underline{A}, \mathsf{env}^A : \mathbb{I} \to A\right)$ consisting of an object $\underline{A}$ in $\underline{\mathcal{C}}$ and an $\underline{\mathcal{C}}$-arrow $\mathsf{env}^A : \mathbb{I} \to A$ in called the *point*; and
- arrows $f : A \to B$ are point-preserving $\underline{\mathcal{C}}$-morphisms $f : \underline{A} \to \underline{B}$ (cf. diagram on right).

$$\begin{tikzcd} & \underline{A} \arrow[dd, "f"] \\ \mathbb{I} \arrow[ur, "\mathsf{env}^A"] \arrow[dr, "\mathsf{env}^B"'] & = \\ & \underline{B} \end{tikzcd}$$

The forgetful functor $\underline{-} : \underline{\mathcal{C}}^{\bullet} \to \underline{\mathcal{C}}$ sending pointed objects and morphisms to their underlying objects and morphisms is faithful. The tensor product $(\otimes)$ lifts along $\underline{-}$ to the following *pointed* tensor:

$$(\otimes^{\bullet}) : \underline{\mathcal{C}}^{\bullet} \times \underline{\mathcal{C}}^{\bullet} \to \underline{\mathcal{C}}^{\bullet} \quad \underline{A \otimes^{\bullet} B} : \underline{A} \otimes \underline{B} \quad \mathsf{env}^{A \otimes^{\bullet} B} \coloneqq \mathbb{I} \xrightarrow{\mathbf{r}'} \mathbb{I} \otimes \mathbb{I} \xrightarrow{\mathsf{env}^A \otimes \mathsf{env}^B} \underline{A} \otimes \underline{B} \quad (A \xrightarrow{f} A') \otimes^{\bullet} (B \xrightarrow{g} B') \coloneqq f \otimes g$$

This definition for $f \otimes^{\bullet} g$ relies on a simple point-preservation argument (Cf. Appendix C.1).

The initial pointed object is given by $\mathbb{I}$ equipped with its identity: $\mathbb{I}^{\bullet} \coloneqq \left(\mathbb{I}, \mathsf{id}_{\mathbb{I}}\right) \in \underline{\mathcal{C}}^{\bullet}$. The unique point-preserving morphism to any pointed object A is given by the point: $[] \coloneqq \mathsf{env}^A : \mathbb{I}^{\bullet} \to A$.

The pointed objects inherit the skew monoidal structure from $\mathcal{C}$ (see Appendix C.1 for the proof):

Proposition 6.1 (Fiore and Szamozvancev [25]) *Every right-skew monoidal category $\mathcal{C}$ yields a right-skew monoidal category of pointed objects $\mathcal{C}^{\bullet} \coloneqq \left(\underline{\mathcal{C}}^{\bullet}, (\otimes^{\bullet}), \mathbb{I}^{\bullet}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$. I.e., the skew mediators preserve points, hence lift to $\mathcal{C}^{\bullet}$. The category $\mathcal{C}^{\bullet}$ is associative, or left/right unital iff $\mathcal{C}$ is.*

Example 6.2 Let $A \in \mathbf{R}\text{-}\mathbf{Struct}^{\bullet}$ be a pointed structure. Due to Yoneda, its point env^A is uniquely determined by the tuple of variable interpretations $\left([\![x : s]\!]_A \coloneqq \mathsf{env}^A_{s;[x:s]}\, x\right)_{s \in \mathbf{R}\text{-}\mathsf{Bind}}$. Given any other pointed structure B, the variable interpretation for $A \otimes^{\bullet} B$ is $[\![x : s]\!]_{A \otimes^{\bullet} B} = \left[[\![x]\!]_A, \left(x : [\![x]\!]_B\right)\right]_{[x:s]}$.

Let $\mathcal{C} = \left(\mathcal{B}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$ be an associative and right-unital category. An *associative, unital right action* $\mathcal{A} = \left(\mathcal{A}, (\rtimes), \mathbf{a}, \mathbf{r}'\right)$ consists of a category $\mathcal{A}$ equipped with a functor $(\rtimes) : \underline{\mathcal{A}} \times \underline{\mathcal{C}} \to \underline{\mathcal{A}}$, an *associator* isomorphism $\mathbf{a} : (x \rtimes a) \rtimes b \xrightarrow{\cong} x \rtimes (a \otimes b)$ natural in x, a, b and a *right unitor* isomorphism $\mathbf{r}' : x \xrightarrow{\cong} x \rtimes \mathbb{I}$ natural in x, satisfying the conditions[6]:

$$\begin{tikzcd}[column sep=small] & & (x \rtimes a) \rtimes (b \otimes c) \arrow[drr, "\mathbf{a}"] & & \\ ((x \rtimes a) \rtimes b) \rtimes c \arrow[urr, "\mathbf{a}"] \arrow[dr, "\mathbf{a} \otimes \mathsf{id}"'] & & \substack{\text{action} \\ \text{pentagon} \\ =} & & x \rtimes (a \otimes (b \otimes c)) \\ & (x \rtimes (a \otimes b)) \rtimes c \arrow[rr, bend right, "\mathbf{a}"'] & & x \rtimes ((a \otimes b) \otimes c) \arrow[ur, "\mathsf{id} \otimes \mathbf{a}"'] & \end{tikzcd}
\qquad
\begin{tikzcd} (x \rtimes \mathbb{I}) \rtimes a \arrow[r, "\mathbf{a}"] & x \rtimes (\mathbb{I} \otimes a) \arrow[d, "\mathsf{id} \otimes \boldsymbol{\ell}"] \\ x \rtimes a \arrow[u, "\mathbf{r}' \otimes \mathsf{id}"] \arrow[r, equal, "\substack{\text{action} \\ \text{rectangle} \\ =}"] & x \rtimes a \end{tikzcd}$$

This definition uses the associativity and right-unitality assumptions to discharge some axioms using Kelly's argument [48]. See Appendix C.2 for the general definition.

Example 6.3 Every right-skew monoidal category $\mathcal{C}$ has a $\mathcal{C}$-action[7] on itself using its own skew monoidal tensor: $(a \oslash b) \coloneqq (a \otimes b)$. The action axioms are a subset of the skew monoidal ones.

Example 6.4 Given a sequence of $\mathcal{C}$-actions $\left(\mathcal{A}_i, (\oslash^i), \mathbb{I}^i, \mathbf{a}^i, \mathbf{r}^{i\prime}\right)_i$, we define the *product $\mathcal{C}$-action* componentwise as follows, validating the axioms componentwise:

$$(\oslash) : (\prod_{i \in I} \mathcal{A}_i) \times \mathcal{C} \to \prod_{i \in I} \mathcal{A}_i \qquad (\boldsymbol{x} \oslash a) \coloneqq \left(x_i \oslash^i a\right)_i \qquad \mathbf{a} \coloneqq \left(\mathbf{a}^i\right)_i \qquad \mathbf{r}' \coloneqq \left(\mathbf{r}^{i\prime}\right)_i$$

Example 6.5 If $\mathcal{A}$ is a $\mathcal{C}$-action, then the monoidal category of pointed objects $\mathcal{C}^{\bullet}$ acts on $\mathcal{A}$ by: $(a \oslash_{\bullet} A) \coloneqq (a \oslash \underline{A})$. The action axioms hold by straightforward calculation. We denote this action by $\mathcal{A}_{\bullet}$.

Let $\mathcal{A}, \mathcal{B}$ be two associative unital right $\mathcal{C}$-actions, and consider any functor $F : \underline{\mathcal{A}} \to \underline{\mathcal{B}}$. A *tensorial strength* for $F : \mathcal{A} \to \mathcal{B}$ is a transformation $\mathsf{str} : (Fa) \oslash_{\mathcal{A}} b \to F(a \oslash_{\mathcal{B}} b)$ satisfying the two axioms:

$$\begin{tikzcd}
 & & (Fx) \oslash (a \otimes b) \arrow[drr, "\mathsf{str}"] & & \\
((Fx) \oslash a) \oslash b \arrow[urr, "\mathbf{a}"] \arrow[dr, "\mathsf{str} \otimes \mathsf{id}"'] & & \begin{array}{c}\text{strength}\\ \text{pentagon}\\ =\end{array} & & F(x \oslash (a \otimes b)) \\
 & (F(x \oslash a)) \oslash b \arrow[rr, "\mathsf{str}"'] & & F((x \oslash a) \oslash b) \arrow[ur, "F\mathbf{a}"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & Fx \arrow[dl, "\mathbf{r}'"'] \arrow[dr, "F\mathbf{r}'"] & \\
(Fx) \oslash \mathbb{I} \arrow[rr, "\begin{array}{c}\text{strength}\\ \text{triangle}\\ =\end{array}", "\mathsf{str}"'] & & F(x \oslash \mathbb{I})
\end{tikzcd}$$

A *strong functor* $F : \mathcal{A} \to \mathcal{B}$ is then a functor $\underline{F} : \underline{\mathcal{A}} \to \underline{\mathcal{B}}$ equipped with a strength, str_F, for $\underline{F} : \mathcal{A} \to \mathcal{B}$.

Definition 6.6 Let $\mathbf{R}$ be a sorting system. An $\mathbf{R}$-*signature functor* is an $\mathbf{R}\text{-}\mathbf{Struct}^{\bullet}$-strong functor $\mathbf{O} : \mathbf{R}\text{-}\mathbf{Struct}_{\bullet} \to \mathbf{R}\text{-}\mathbf{Struct}_{\bullet}$ (cf. Ex. 6.5).

Example 6.7 Recall the *scope shift* combinator $(\rhd\Gamma) : \mathbf{R}\text{-}\mathbf{Struct} \to \mathbf{R}\text{-}\mathbf{Struct}$ that we use to describe binding constructs, such as abstraction. We define its strength $\mathsf{str}^{\Gamma\rhd}_{P,A} : (\Gamma \rhd P) \otimes_{\bullet} A \to \Gamma \rhd (P \otimes_{\bullet} A)$ by:

$$\mathsf{str}^{\Gamma\rhd}_{s;\Xi} \left[t \in P_s\Delta, e \in \underline{A}^{\mathrm{Env}}_{\Delta}\Xi\right]_{\Delta} \coloneqq \left[t, \left(e_x \left[[x : s] \xleftarrow{\pi_1} \Xi \mathbin{+\!\!+} \Gamma\right]\right)_{(x:s)\in\Delta} \mathbin{+\!\!+} \left(\llbracket y \rrbracket_A \left[[y : r] \xleftarrow{\pi_{y[\pi_2]}} \Xi \mathbin{+\!\!+} \Gamma\right]\right)_{(y:r)\in\Gamma}\right]_{\Delta \mathbin{+\!\!+} \Gamma}$$

When the operators in a signature do not bind variables themselves but are merely compatible with binding, we can exhibit a strength w.r.t. the simpler action $(\otimes)$ thanks to the following result, generalising an implicit result in Fiore's [26] work (see Appendix C.2 for the straightforward proof):

Lemma 6.8 *Let $\mathcal{C}$ be a skew monoidal category; $\mathcal{A}$,$\mathcal{B}$ two $\mathcal{C}$-actions; and $F : \mathcal{A} \to \mathcal{B}$ a strong functor. Then the following morphisms exhibit $\underline{F}$ as a strong functor: $F_{\bullet} : \mathcal{A}_{\bullet} \to \mathcal{B}_{\bullet}$.*

$$\mathsf{str}^{F_{\bullet}}_{x,a} : (\underline{F}x) \oslash_{\bullet} a = (\underline{F}x) \oslash \underline{a} \xrightarrow{\mathsf{str}^{F}_{x,\underline{a}}} \underline{F}(x \oslash \underline{a}) = \underline{F}(x \oslash_{\bullet} a)$$

The next few examples all use this lemma to derive strengths.

Example 6.9 Recall the *restriction* combinator $(|_I) : \mathbf{R}\text{-}\mathbf{Struct} \to \mathbf{PSh}\left(I \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right)$ and its specialisation $(@s_0) : \mathbf{R}\text{-}\mathbf{Struct} \to \mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}\right)$ (where $s_0 \in \mathbf{R}\text{-}\mathsf{sort}$) that we use it to signify a subterm of sort s_0. Define $\mathsf{str}^{|_I} : P|_I \otimes Q \to (P \otimes Q)|_I$ by $\mathsf{str}^{|_I}_{s,\Gamma} \left[t \in P_{s_0}\Delta, e \in Q^{\mathrm{Env}}_{\Delta}\Gamma\right]_{\Delta} \coloneqq [t, e]_{\Delta}$, and derive $\mathsf{str}^{@s_0} : (P \mathbin{@} s_0) \otimes Q \to (P \otimes Q) \mathbin{@} s_0$ from it. Similarly, recall the *extension* combinator $\looparrowright_I : \mathbf{PSh}\, I \times \mathbf{R}\text{-}\mathsf{Bind}_{\vdash} \to \mathbf{R}\text{-}\mathbf{Struct}$ that lets us construct nodes at a specific subset of sorts $I \subseteq \mathbf{R}\text{-}\mathsf{sort}$.

[6] Overloading the associator notation for tensors and action can always be resolved.

[7] We defined actions assuming associativity and right-unitality, but this fact holds for the actions of Appendix C.2.

Then $((\looparrowright_I P) \otimes Q)_s = \mathbb{0}$ when $s \notin I$, and so we can define the strength $\mathsf{str}^{\looparrowright_I}_{P;Q} : (\looparrowright_P) \otimes Q \to \looparrowright_{P\otimes Q}$ vacuously in those sorts:

$$\mathsf{str}^{\looparrowright_I}_{P;Q;s} : ((\underset{P}{\looparrowright}) \otimes Q)_s = \mathbb{0} \xrightarrow{[]} \underset{I}{\looparrowright}(P \otimes Q)_s \qquad \mathsf{str}^{\looparrowright_I}_{P;Q;r} : ((\underset{P}{\looparrowright}) \otimes Q)_r = (P \otimes Q)_r = \underset{I}{\looparrowright}(P \otimes Q)_r \qquad (s \notin I \ni r)$$

Example 6.10 Let $\left(\mathcal{A}_i\right)_{i\in I}$ be a family of $\mathcal{C}$-actions. The projection functors $\boldsymbol{\pi}_j : \prod_{i\in I} \mathcal{A}_i \to \mathcal{A}_j$ have the identity as a strength $\mathsf{str}^{\boldsymbol{\pi}_j}_{(x_i)_i,a} = \mathsf{id}_{x_i \rtimes_i a}$ since: $\left(\boldsymbol{\pi}_j \left(x_i\right)_i\right) \rtimes_i a = x_j \rtimes a = \boldsymbol{\pi}_j\left(x_i\right)_i \rtimes a$. For a $\mathcal{C}$-action $\mathcal{B}$ and a family of strong functors $F_i : \mathcal{B} \to \mathcal{A}_i$, the tupling functor $\left(\underline{F}_i\right)_i : \mathcal{B} \to \prod_i \mathcal{A}_i$ has the following $\prod_i \mathcal{A}_i$-morphism as strength: $\mathsf{str}^{(F_i)_i}_{(x)_i,a} \coloneqq \left((F_i x) \rtimes a \xrightarrow{\mathsf{str}^{F_i}} F_i(x \rtimes a)\right)_i$.

Example 6.11 Recall the I-ary sums $\coprod_I : \mathcal{C}^I \to \mathcal{C}$ and J-ary products $\prod_J : \mathcal{C}^J \to \mathcal{C}$ that let us alternate between I-indexed operator nodes and include J-ary branching factor in categories with I-coproducts/J-products. The product has strength, and coproducts, if they distribute over $(\otimes)$, also have a strength:

$$\mathsf{str}^{\prod_J} : \left(\prod_{j\in J} x_j\right) \rtimes a \xrightarrow{(\boldsymbol{\pi}_j \rtimes \mathsf{id})_{j\in J}} \prod_{j\in J}(x_j \rtimes a) \qquad \mathsf{str}^{\coprod_I} : \left(\coprod_{i\in I} x_i\right) \rtimes a \xrightarrow{[(i:)\rtimes\mathsf{id}]^{-1}_{i\in I}} \coprod_{i\in I}(x_i \rtimes a)$$

When $\mathcal{C} = \mathbf{R}\textbf{-Struct}$, the substitution tensor $(\otimes)$ distributes over coproducts (cf. Prop. B.1), and so the following pointwise formulae describe these strengths:

$$\mathsf{str}^{\prod_J}_{\boldsymbol{P},\boldsymbol{Q}} [t, e]_\Delta \coloneqq \left(\left[t_j, e\right]_\Delta\right)_{j\in J} \qquad \mathsf{str}^{\coprod_I} : [(i : t), e]_\Delta \coloneqq (i : [t, e]_\Delta)$$

As is well-known, strong functors compose, and their combined strength is given by (see Lemma C.4):

$$\mathsf{str}^{G\circ F}_{x,a} : (\underline{GF}x) \rtimes a \xrightarrow{\mathsf{str}^G} \underline{G}(\underline{F}x \rtimes a) \xrightarrow{\underline{G}\,\mathsf{str}^F} \underline{GF}(x \rtimes a)$$

This fact, together with the signature combinator toolkit, covers a wide range of examples.

Example 6.12 A straightforward calculation derives the strength for the abstraction signature (Ex. 4.5):

$$\mathsf{LamOp} \coloneqq \left\{\!\!\left[(\lambda x : A.) : \underset{A\to B}{\looparrowright}[x : A] \rhd X \mathbin{@} A \,\middle|\, A, B \in \mathsf{SimpleType}\right]\!\!\right\}$$

$$\mathsf{str}^{\mathsf{LamOp}}_{P,A;A\to B,\Gamma} [\lambda x : A.t, e]_\Delta \coloneqq \lambda x : A. \left[t, e\left[\Gamma \xleftarrow{\boldsymbol{\pi}_1} \Gamma + [x : A]\right] + \mathsf{env}^A_{\Gamma + [x:A]}\, x\right]_{\Delta + [x:A]}$$

We similarly derive strengths for the other cbv signature functors from Exs. 4.3 and 4.4:

$$\mathsf{str}^{\mathsf{ValOp}} [\mathsf{val}\, t, e]_\Delta \coloneqq \mathsf{val}\, [t, e]_\Delta \qquad \mathsf{str}^{\mathsf{AppOp}} \left[t_1 \mathbin{@} t_2, e\right]_\Delta \coloneqq \left[t_1, e\right]_\Delta \mathbin{@} \left[t_2, e\right]_\Delta$$

7 Compatible monoids

We define the substitution structure of interpretations of the syntax that ensures they are compatible with the operations in the signature, concluding the tutorial with the Special Representation Thm 7.4.

Definition 7.1 Let $\mathbf{O}$ be an $\mathbf{R}$-signature functor, and $\mathbf{M}$ a substitution monoid. We say that an $\mathbf{O}$-algebra $[\![-]\!] : \underline{\mathbf{OM}} \to \underline{\mathbf{M}}$ is *compatible* with $\mathbf{M}$, when:

$$\begin{tikzcd}
 & \underline{\mathbf{O}}(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[rrd, "\underline{\mathbf{O}}(-[-]_{\mathbf{M}})"] & & \\
(\underline{\mathbf{OM}}) \otimes_\bullet \mathsf{env}^{\mathbf{M}} \arrow[ru, "\mathsf{str}"] \arrow[rd, "{[\![-]\!] \otimes \mathsf{id}}"'] & & \text{compatibility} \atop = & \underline{\mathbf{OM}} \arrow[d, "{[\![-]\!]}"] \\
 & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[rr, "-[-]_{\mathbf{M}}"'] & & \underline{\mathbf{M}}
\end{tikzcd}$$

An $\mathbf{O}$-*compatible substitution monoid* $\mathbf{M} = \left(\underline{\mathbf{M}}, -[-]_{\mathbf{M}}, \mathsf{env}^{\mathbf{M}}, \mathbf{M}[\![-]\!]\right)$, or $\mathbf{O}$-*monoid* for short, consists of a monoid $\mathbf{M}_{\mathrm{mon}} = \left(\underline{\mathbf{M}}, -[-]_{\mathbf{M}}, \mathsf{env}^{\mathbf{M}}\right)$ and an $\underline{\mathbf{O}}$-algebra $\mathbf{M}[\![-]\!] : \underline{\mathbf{O}\mathbf{M}} \to \underline{\mathbf{M}}$, compatible with $\mathbf{M}_{\mathrm{mon}}$.

Example 7.2 Recall the substitution monoid $\mathbf{M}$ for CBV in a Cartesian-closed category $\mathcal{C}$ with chosen finite products (Ex. 5.6). The standard interpretation of the CBV λ-calculus equips it with a LamOp-, ValOp-, and AppOp-algebra structures:

$$\mathbf{M}[\![\lambda x : A.]\!]\left([\![\Gamma, x : A]\!] \xrightarrow{f} [\![B]\!]\right) := \left([\![\Gamma]\!] \xrightarrow{\mathsf{curry}\, f} [\![B]\!]^{[\![A]\!]}\right) \qquad \mathbf{M}[\![\mathsf{val}]\!]\left([\![\Gamma]\!] \xrightarrow{f} [\![A]\!]\right) := f$$

$$\mathbf{M}\left[\!\!\left[\left([\![\Gamma]\!] \xrightarrow{f} [\![B]\!]^{A}\right) @ \left([\![\Gamma]\!] \xrightarrow{a} [\![B]\!]\right)\right]\!\!\right] := \left([\![\Gamma]\!] \xrightarrow{(f,a)} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{\mathsf{eval}} [\![B]\!]\right)$$

The compatibility axiom for LamOp amounts to the following equation, for each $\theta \in \mathcal{C}([\![\Gamma]\!], [\![\Delta]\!]) \cong \mathbf{M}^{\mathrm{Env}}_{\Delta}\Gamma$:

$$\mathsf{curry}\,([\![\lambda x : A.]\!]\, f) \circ \left(\theta_y\right)_y = \mathsf{curry}\left([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{(\theta_y)_y \times \mathsf{id}} [\![\Delta]\!] \xrightarrow{f} [\![B]\!]\right)$$

It follows from the naturality curry. The compatibility axioms for ValOp and AppOp hold immediately.

The following lemma allows us to combine these compatibility to the model structure for the whole CBV language (cf. Appendix C.3 for the proof):

Lemma 7.3 *Let* $\left(\mathbf{O}_i\right)_{i \in I}$ *be a family of* $\mathbf{R}$*-signature functors; let* $\mathbf{M}$ *be a substitution monoid; and let* $\left([\![-]\!]_i : \underline{\mathbf{O}_i\mathbf{M}} \to \underline{\mathbf{M}}\right)_{i \in I}$ *be a family of algebras. The cotupled algebra:* $\left[[\![-]\!]_i\right]_{i \in I} : \coprod_{i \in I} \underline{\mathbf{O}_i\mathbf{M}} \to \underline{\mathbf{M}}$ *is compatible with* $\mathbf{M}$ *iff every algebra* $\mathbf{O}_i$ *is compatible with* $\mathbf{M}$.

An $\mathbf{O}$-monoid *homomorphism* $h : \mathbf{M} \to \mathbf{N}$ is a morphism $h : \underline{\mathbf{M}} \to \underline{\mathbf{N}}$ that is both a $\mathbf{O}$-homomorphism and a monoid homomorphism. Let $\mathbf{H} \in \mathbf{R\text{-}Struct}$ be an $\mathbf{R}$-structure, whose elements we think of as holes. An $\mathbf{O}$-*monoid over* $\mathbf{H}$ is a pair $(\mathbf{M}, \mathsf{menv})$ consisting of a $\mathbf{O}$-monoid $\mathbf{M}$ and a morphism $\mathsf{menv} : \mathbf{H} \to \underline{\mathbf{M}}$, which we will call the *metavariable environment.* A morphism of $\mathbf{O}$-monoids $h : \left(\mathbf{M}, \mathsf{menv}^{\mathbf{M}}\right) \to \left(\mathbf{N}, \mathsf{menv}^{\mathbf{N}}\right)$ over $\mathbf{H}$ is an $\mathbf{O}$-monoid homomorphism $h : \mathbf{M} \to \mathbf{N}$ preserving the metavariable environment.

Theorem 7.4 (special representation) *Let* $\mathbf{O}$ *be an* $\mathbf{R}$*-signature functor, and* $\mathbf{H}$ *an* $\mathbf{R}$*-structure. Consider any initial algebra* $\boldsymbol{\mu}X.(\underline{\mathbf{O}}X) \amalg \mathbb{I} \amalg (\mathbf{H} \otimes X)$ *given by* $\left(\$^{\mathbf{O}}\mathbf{H}, [\![-]\!], \mathsf{var}, \mathsf{meta}\right)$. *There is a unique morphism* $-[-] : \$^{\mathbf{O}}\mathbf{H} \otimes \$^{\mathbf{O}}\mathbf{H} \to \$^{\mathbf{O}}\mathbf{H}$*, called* simultaneous substitution*, satisfying the following equations:*

$$\begin{tikzcd}
 & \underline{\mathbf{O}}(\$\mathbf{H} \otimes \$\mathbf{H}) \arrow[drr, "\underline{\mathbf{O}}(-[-])"] & & \\
(\underline{\mathbf{O}}\$\mathbf{H}) \otimes_\bullet \mathsf{var} \arrow[ur, "\mathsf{str}"] \arrow[dr, "{[\![-]\!] \otimes \mathsf{id}}"'] & \overset{\text{op}}{\underset{=}{\text{case}}} & & \underline{\mathbf{O}}\$\mathbf{H} \arrow[d, "{[\![-]\!]}"] \\
 & \$\mathbf{H} \otimes \$\mathbf{H} \arrow[rr, "{-[-]}"'] & & \$\mathbf{H}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathbb{I} \otimes \$\mathbf{H} \arrow[dl, "\mathsf{var} \otimes \mathsf{id}"'] \arrow[dr, "\boldsymbol{\ell}"] & \\
\$\mathbf{H} \otimes \$\mathbf{H} \arrow[rr, "{-[-]}"'] & \overset{\text{var}}{\underset{=}{\text{case}}} & \$\mathbf{H}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathbf{H} \otimes (\$\mathbf{H} \otimes \$\mathbf{H}) \arrow[rr, "{\mathsf{id} \otimes (-[-])}"] & & \mathbf{H} \otimes \$\mathbf{H} \arrow[dr, "\mathsf{meta}"] & \\
(\mathbf{H} \otimes \$\mathbf{H}) \otimes \$\mathbf{H} \arrow[ur, "\mathbf{a}"] \arrow[drr, "\mathsf{meta} \otimes \mathsf{id}"'] & & \overset{\text{metavariable}}{\underset{=}{\text{case}}} & & \$\mathbf{H} \\
 & & \$\mathbf{H} \otimes \$\mathbf{H} \arrow[urr, "{-[-]}"'] & &
\end{tikzcd}$$

Equipping $\mathbf{F}_{\mathbf{O}}\mathbf{H} := \left(\$^{\mathbf{O}}\mathbf{H}, -[-], \mathsf{var}, [\![-]\!]\right)$ *with* $\mathsf{menv} : \mathbf{H} \xrightarrow{\mathbf{r}'} \mathbf{H} \otimes \mathbb{I} \xrightarrow{\mathsf{id} \otimes \mathsf{var}} \mathbf{H} \otimes \$^{\mathbf{O}}\mathbf{H} \xrightarrow{\mathsf{meta}} \$^{\mathbf{O}}\mathbf{H}$*, yields the free* $\mathbf{O}$*-monoid over* $\mathbf{H}$.

With this theorem we can prove substitution lemmata analogously to Lemma 2.1 and its proof.

8 Case study: CBV

We summarise a case study for MAST, and defer the technical details to Appendix A.

We use algebraic signatures to modularly describe extensions to the CBV type system, and signature functors to modularly describe extensions to the terms. Starting with a semantics based on strong monads, we extend a basic calculus with sequential composition, functions, products, coproducts, an inductive datatype of natural numbers, iteration, and recursion. These require 7 checks that each additional bunch of semantic definitions is well-defined and compatible with the substitution structures in the corresponding algebra. In return, the MAST theory lets us deduce $2^7 = 128$ different substitution lemmata, for each language fragment. Note that not all models can interpret each fragment, and so one cannot deduce the substitution lemma for a fragment from the lemma for the full language.

9 Related work

The two POPL-Mark challenges [11,2] galvanise the programming-language community to hard problems in formalisation. Both challenges emphasise representation and manipulation of syntax with binding and substitution. Our work does not target mechnisation and computational realisation of abstract syntax with binding and substitution, but we note the ample work of this nature. All modern mechanisation systems support libraries or features for abstract syntax with binding [66,10,74,33,68,69,62,35,36,67, e.g.]. These typically generate specialised functions and lemmata given a description of the syntax, rather than appeal to a generic data structure and a general theory of syntax and substitution. We direct the reader to Allais et al.'s related work section [7] which surveys: non-de Bruijn approaches to binding [34,17,15]; alternative binding structure [76,65,16,38,42]; additional work on automation [64,51]; universes of syntax and generic programming [50,49,12,54,77,19,59]. To those we add HOAS [46,61], monadic [8] and functorial [13] representations.

The presheaf approach [26,9,31,30] lends itself to mathematical operational semantics [73,40] and G. structural operational semantics (GSOS) [41]. Fiore and Turi [32] considered a special case of our heterogeneous situation in which the constructs for first-class sorts can be described independently from the second-class sorts in terms of actions. In our situation, we have mutual dependency between terms of first-class and second-class sorts, e.g., CBV terms include values, and functions, which are values, are abstracted terms. McBride varies the indexing category e.g., thinnings/order-preserving embeddings to implement co-de Bruijn representations [57]. In a different direction, Fiore and Saville reduce the free **O**-monoid to the existence of certain list objects [23].

Close to the presheaf approach is the familial approach of Hirschowitz et al. [44,14,43,45], which also uses a skew tensor for technical reasons involving operational semantics and bisimilarity. Ahrens also uses families rather than presheaves [4] including an implementation [5] in UniMath [75]. For more connections between these approaches, see the introductory text by Lamiaux and Ahrens [53], and especially its comprehensive related work section.

Fiore, Gambino, Hyland, and Winskel's Kleisli bicategories [28] follow their earlier work [29] generalising Joyal's species of structure [47]. Olimpieri et al. [60,18] used these, e.g., when studying intersection type systems. More recently, Fiore, Galal, and Paquet introduce a bicategory of stable species [27]. Ahrens et al's aforementioned implementation [5] also uses such bicategorical ideas.

10 Conclusion

We have extended the classical theory of abstract syntax with binding and substitution with second-class sorts through skew structure. This extension is straightforward thanks to existing work on skew monoidal categories for abstract syntax and to the bicategorical perspective. We separated the concepts needed to employ the theory from its underlying technical machinery. We expect similar results from an analogous case study on Levy's CBPV. There the basic calculus comprises of returners and sequencing, and one can extend it with value products and coproducts, thunks, computation products, and functions. Thus, with MAST one could deduce $2^6 = 64$ different substitution lemmata by checking the compatibility conditions for 6 signatures. In the future, we may be able to avoid skewness through monoid actions, generalising Fiore and Turi's semantics for value passing [32]. We could then develop a corresponding algorithmic theory using familial skew structures [25] and implement it.

References

[1] *Monads on symmetric monoidal closed categories*, Archiv der Mathematik **21**, pages 1–10 (1970).
https://doi.org/10.1007/BF01220868

[2] Abel, A., G. Allais, A. Hameer, B. Pientka, A. Momigliano, S. Schäfer and K. Stark, *Poplmark reloaded: Mechanizing proofs by logical relations*, Journal of Functional Programming **29**, page e19 (2019).
https://doi.org/10.1017/S0956796819000170

[3] Aczel, P., *A general Church-Rosser theorem. University of Manchester*, Technical report, Technical report (1978).

[4] Ahrens, B., *Modules over relative monads for syntax and semantics*, Mathematical Structures in Computer Science **26**, page 3–37 (2016).
https://doi.org/10.1017/S0960129514000103

[5] Ahrens, B., R. Matthes and A. Mörtberg, *Implementing a category-theoretic framework for typed abstract syntax*, in: *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2022, page 307–323, Association for Computing Machinery, New York, NY, USA (2022), ISBN 9781450391825.
https://doi.org/10.1145/3497775.3503678

[6] Allais, G., R. Atkey, J. Chapman, C. McBride and J. McKinna, *A type and scope safe universe of syntaxes with binding: their semantics and proofs*, Proc. ACM Program. Lang. **2** (2018).
https://doi.org/10.1145/3236785

[7] Allais, G., R. Atkey, J. Chapman, C. Mcbride and J. Mckinna, *A type- and scope-safe universe of syntaxes with binding: their semantics and proofs*, Journal of Functional Programming **31** (2021).
https://doi.org/10.1017/S0956796820000076

[8] Altenkirch, T. and B. Reus, *Monadic presentations of lambda terms using generalized inductive types*, in: J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL, Madrid, Spain, September 20-25, 1999, Proceedings*, volume 1683 of *Lecture Notes in Computer Science*, pages 453–468, Springer (1999).
https://doi.org/10.1007/3-540-48168-0_32

[9] Arkor, N. and D. McDermott, *Abstract Clones for Abstract Syntax*, in: N. Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021)*, volume 195 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:19, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021), ISBN 978-3-95977-191-7, ISSN 1868-8969.
https://doi.org/10.4230/LIPIcs.FSCD.2021.30

[10] Aydemir, B., A. Charguéraud, B. C. Pierce, R. Pollack and S. Weirich, *Engineering formal metatheory*, in: *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, page 3–15, Association for Computing Machinery, New York, NY, USA (2008), ISBN 9781595936899.
https://doi.org/10.1145/1328438.1328443

[11] Aydemir, B. E., A. Bohannon, M. Fairbairn, J. N. Foster, B. C. Pierce, P. Sewell, D. Vytiniotis, G. Washburn, S. Weirich and S. Zdancewic, *Mechanized metatheory for the masses: The poplmark challenge*, in: J. Hurd and T. Melham, editors, *Theorem Proving in Higher Order Logics*, pages 50–65, Springer Berlin Heidelberg, Berlin, Heidelberg (2005), ISBN 978-3-540-31820-0.

[12] Benton, N., C.-K. Hur, A. J. Kennedy and C. Mcbride, *Strongly typed term representations in coq*, J. Autom. Reason. **49**, page 141–159 (2012), ISSN 0168-7433.
https://doi.org/10.1007/s10817-011-9219-0

[13] Blanchette, J. C., L. Gheri, A. Popescu and D. Traytel, *Bindings as bounded natural functors*, Proc. ACM Program. Lang. **3**, pages 22:1–22:34 (2019).
https://doi.org/10.1145/3290335

[14] Borthelle, P., T. Hirschowitz and A. Lafont, *A cellular howe theorem*, in: H. Hermanns, L. Zhang, N. Kobayashi and D. Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 273–286, ACM (2020).
https://doi.org/10.1145/3373718.3394738

[15] Charguéraud, A., *The locally nameless representation*, Journal of automated reasoning **49**, pages 363–408 (2012).

[16] Cheney, J., *Toward a general theory of names: binding and scope*, in: *Proceedings of the 3rd ACM SIGPLAN Workshop on Mechanized Reasoning about Languages with Variable Binding*, MERLIN '05, page 33–40, Association for Computing Machinery, New York, NY, USA (2005), ISBN 1595930728.
https://doi.org/10.1145/1088454.1088459

[17] Chlipala, A., *Parametric higher-order abstract syntax for mechanized semantics*, in: *Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming*, ICFP '08, page 143–156, Association for Computing Machinery, New York, NY, USA (2008), ISBN 9781595939197.
https://doi.org/10.1145/1411204.1411226

[18] Clairambault, P., F. Olimpieri and H. Paquet, *From Thin Concurrent Games to Generalized Species of Structures* , in: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–14, IEEE Computer Society, Los Alamitos, CA, USA (2023).
https://doi.org/10.1109/LICS56636.2023.10175681

[19] Copello, E., N. Szasz and Á. Tasistro, *Formalization of metatheory of the lambda calculus in constructive type theory using the barendregt variable convention*, Mathematical Structures in Computer Science **31**, page 341–360 (2021).
https://doi.org/10.1017/S0960129521000335

[20] Crole, R. L., *The representational adequacy of Hybrid*, Math. Struct. Comput. Sci. **21**, pages 585–646 (2011).
https://doi.org/10.1017/S0960129511000041

[21] de Bruijn, N., *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem*, Indagationes Mathematicae (Proceedings) **75**, pages 381–392 (1972), ISSN 1385-7258.
https://doi.org/https://doi.org/10.1016/1385-7258(72)90034-0

[22] Eilenberg, S. and G. M. Kelly, *Closed categories*, in: S. Eilenberg, D. K. Harrison, S. MacLane and H. Röhrl, editors, *Proceedings of the Conference on Categorical Algebra*, pages 421–562, Springer Berlin Heidelberg, Berlin, Heidelberg (1966), ISBN 978-3-642-99902-4.

[23] Fiore, M. and P. Saville, *List Objects with Algebraic Structure*, in: D. Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2017), ISBN 978-3-95977-047-7, ISSN 1868-8969.
https://doi.org/10.4230/LIPIcs.FSCD.2017.16

[24] Fiore, M. and D. Szamozvancev, *Formal metatheory of second-order abstract syntax*, Proc. ACM Program. Lang. **6** (2022).
https://doi.org/10.1145/3498715

[25] Fiore, M. and D. Szamozvancev, *Familial model of second-order abstract syntax* (2025). Unpublished.

[26] Fiore, M. P., *Second-order and dependently-sorted abstract syntax*, in: *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 57–68, IEEE Computer Society (2008).
https://doi.org/10.1109/LICS.2008.38

[27] Fiore, M. P., Z. Galal and H. Paquet, *Stabilized profunctors and stable species of structures*, Log. Methods Comput. Sci. **20** (2024).
https://doi.org/10.46298/LMCS-20(1:17)2024

[28] Fiore, M. P., N. Gambino, M. Hyland and G. Winskel, *Relative pseudomonads, Kleisli bicategories, and substitution monoidal structures*, Selecta Mathematica New Series **24**, pages 2791–2830 (2018).
https://doi.org/10.1007/s00029-017-0361-3

[29] Fiore, M. P., N. Gambino, M. H. Hyland and G. Winskel, *The cartesian closed bicategory of generalised species of structures*, Journal of the London Mathematical Society **77**, pages 203–220 (2008).

[30] Fiore, M. P. and M. Hamana, *Multiversal polymorphic algebraic theories: Syntax, semantics, translations, and equational logic*, in: *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, page 520–529, IEEE Computer Society, USA (2013), ISBN 9780769550206.

[31] Fiore, M. P., G. D. Plotkin and D. Turi, *Abstract syntax and variable binding (extended abstract)*, in: *Proc. 14$^{\text{th}}$ LICS Conf.*, pages 193–202, IEEE, Computer Society Press (1999).

[32] Fiore, M. P. and D. Turi, *Semantics of name and value passing*, in: *16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings*, pages 93–104, IEEE Computer Society (2001).
https://doi.org/10.1109/LICS.2001.932486

[33] Forster, Y. and K. Stark, *Coq à la carte — a practical approach to modular syntax with binders*, in: *9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, USA*, ACM (2020).

[34] Gabbay, M. J. and A. M. Pitts, *A new approach to abstract syntax with variable binding*, Form. Asp. Comput. **13**, page 341–363 (2002), ISSN 0934-5043.
https://doi.org/10.1007/s001650200016

[35] Gacek, A., *A Framework for Specifying, Prototyping, and Reasoning about Computational Systems*, Ph.D. thesis, University of Minnesota (2009).

[36] Gacek, A., D. Miller and G. Nadathur, *A two-level logic approach to reasoning about computations*, Journal of Automated Reasoning **49**, pages 241–273 (2012).

[37] Garner, R. and J.-S. P. Lemay, *Cartesian differential categories as skew enriched categories*, Applied Categorical Structures **29**, pages 1099–1150 (2021).
https://doi.org/https://doi.org/10.1007/s10485-021-09649-7

[38] Ghani, N., M. Hamana, T. Uustalu and V. Vene, *Representing cyclic structures as nested datatypes* pages 173–188 (2006).

[39] Goguen, J. A., J. W. Thatcher, E. G. Wagner and J. B. Wright, *Initial algebra semantics and continuous algebras*, J. ACM **24**, page 68–95 (1977), ISSN 0004-5411.
https://doi.org/10.1145/321992.321997

[40] Goncharov, S., S. Milius, L. Schröder, S. Tsampas and H. Urbat, *Towards a higher-order mathematical operational semantics*, Proc. ACM Program. Lang. **7**, pages 632–658 (2023).
https://doi.org/10.1145/3571215

[41] Goncharov, S., S. Tsampas and H. Urbat, *Abstract operational methods for call-by-push-value*, Proc. ACM Program. Lang. **9**, pages 1013–1039 (2025).
https://doi.org/10.1145/3704871

[42] Hamana, M., *Initial algebra semantics for cyclic sharing structures*, in: P.-L. Curien, editor, *Typed Lambda Calculi and Applications*, pages 127–141, Springer Berlin Heidelberg, Berlin, Heidelberg (2009), ISBN 978-3-642-02273-9.

[43] Hirschowitz, A., T. Hirschowitz and A. Lafont, *Modules over monads and operational semantics*, in: Z. M. ndré, editor, *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, volume 167 of *LIPIcs*, pages 12:1–12:23, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020).
https://doi.org/10.4230/LIPICS.FSCD.2020.12

[44] Hirschowitz, A., T. Hirschowitz, A. Lafont and M. Maggesi, *Variable binding and substitution for (nameless) dummies*, CoRR **abs/2209.02614** (2022). 2209.02614.
https://doi.org/10.48550/ARXIV.2209.02614

[45] Hirschowitz, A. and M. Maggesi, *Modules over monads and initial semantics*, Information and Computation **208**, pages 545–564 (2010), ISSN 0890-5401. Special Issue: 14th Workshop on Logic, Language, Information and Computation (WoLLIC 2007).
https://doi.org/https://doi.org/10.1016/j.ic.2009.07.003

[46] Hofmann, M., *Semantical analysis of higher-order abstract syntax*, in: *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, pages 204–213 (1999).
https://doi.org/10.1109/LICS.1999.782616

[47] Joyal, A., *Une théorie combinatoire des séries formelles*, Advances in Mathematics **42**, pages 1–82 (1981), ISSN 0001-8708.
https://doi.org/https://doi.org/10.1016/0001-8708(81)90052-9

[48] Kelly, G. M., *On MacLane's conditions for coherence of natural associativities, commutativities, etc.*, Journal of Algebra **1**, pages 397–402 (1964), ISSN 0021-8693.
https://doi.org/https://doi.org/10.1016/0021-8693(64)90018-3

[49] Keuchel, S., *Generic programming with binders and scope* (2011).

[50] Keuchel, S. and J. T. Jeuring, *Generic conversions of abstract syntax representations*, in: *Proceedings of the 8th ACM SIGPLAN Workshop on Generic Programming*, WGP '12, page 57–68, Association for Computing Machinery, New York, NY, USA (2012), ISBN 9781450315760.
https://doi.org/10.1145/2364394.2364403

[51] Keuchel, S., S. Weirich and T. Schrijvers, *Needle & knot: Binder boilerplate tied up*, in: *Proceedings of the 25th European Symposium on Programming Languages and Systems - Volume 9632*, page 419–445, Springer-Verlag, Berlin, Heidelberg (2016), ISBN 9783662494974.
https://doi.org/10.1007/978-3-662-49498-1_17

[52] Lack, S. and R. Street, *On monads and warpings*, Cahiers de topologie et géométrie différentielle catégoriques **LV**, pages 244–266 (2014), ISSN 1245-530X.

[53] Lamiaux, T. and B. Ahrens, *An introduction to different approaches to initial semantics* (2024). 2401.09366.
https://arxiv.org/abs/2401.09366

[54] Lee, G., B. C. D. S. Oliveira, S. Cho and K. Yi, *Gmeta: A generic formal metatheory framework for first-order representations*, in: H. Seidl, editor, *Programming Languages and Systems*, pages 436–455, Springer Berlin Heidelberg, Berlin, Heidelberg (2012), ISBN 978-3-642-28869-2.

[55] Levy, P. B., *Call-by-push-value: A subsuming paradigm*, in: *Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications*, TLCA '99, page 228–242, Springer-Verlag, Berlin, Heidelberg (1999), ISBN 3540657630.

[56] Levy, P. B., *Call-By-Push-Value: A Functional/Imperative Synthesis*, volume 2 of *Semantics Structures in Computation*, Springer (2004).

[57] McBride, C., *Everybody's got to be somewhere*, in: R. Atkey and S. Lindley, editors, *Proceedings of the 7th Workshop on Mathematically Structured Functional Programming, MSFP@FSCD 2018, Oxford, UK, 8th July 2018*, volume 275 of *EPTCS*, pages 53–69 (2018).
https://doi.org/10.4204/EPTCS.275.6

[58] McDermott, D. and T. Uustalu, *What makes a strong monad?*, Electronic Proceedings in Theoretical Computer Science **360**, page 113–133 (2022), ISSN 2075-2180.
https://doi.org/10.4204/eptcs.360.6

[59] Morris, P., T. Altenkirch and C. McBride, *Exploring the regular tree types*, in: J.-C. Filliâtre, C. Paulin-Mohring and B. Werner, editors, *Types for Proofs and Programs*, pages 252–267, Springer Berlin Heidelberg, Berlin, Heidelberg (2006), ISBN 978-3-540-31429-5.

[60] Olimpieri, F., *Intersection Types and Resource Calculi in the Denotational Semantics of Lambda-Calculus*, Theses, Aix-Marseille Universite (2020).
https://theses.hal.science/tel-03123485

[61] Pfenning, F. and C. Elliott, *Higher-order abstract syntax*, in: *Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation*, PLDI '88, page 199–208, Association for Computing Machinery, New York, NY, USA (1988), ISBN 0897912691.
https://doi.org/10.1145/53990.54010

[62] Pientka, B., *Beluga: Programming with dependent types, contextual data, and contexts*, in: M. Blume, N. Kobayashi and G. Vidal, editors, *Functional and Logic Programming*, pages 1–12, Springer Berlin Heidelberg, Berlin, Heidelberg (2010), ISBN 978-3-642-12251-4.

[63] Plotkin, G. D., *An illative theory of relations*, Situation Theory and its Applications pages 133–146 (1990).

[64] Polonowski, E., *Automatically generated infrastructure for de bruijn syntaxes*, in: S. Blazy, C. Paulin-Mohring and D. Pichardie, editors, *Interactive Theorem Proving*, pages 402–417, Springer Berlin Heidelberg, Berlin, Heidelberg (2013), ISBN 978-3-642-39634-2.

[65] Poulsen, C., A. Rouvoet, A. Tolmach, R. Krebbers and E. Visser, *Intrinsically-typed definitional interpreters for imperative languages*, Proceedings of the ACM on Programming Languages **2**, pages 1–34 (2018), ISSN 2475-1421.
https://doi.org/10.1145/3158104

[66] Schäfer, S., G. Smolka and T. Tebbi, *Completeness and decidability of de bruijn substitution algebra in coq*, in: *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 67–73, ACM (2015).

[67] Sewell, P., F. Z. Nardelli, S. Owens, G. Peskine, T. Ridge, S. Sarkar and R. Strniša, *Ott: Effective tool support for the working semanticist*, Journal of Functional Programming **20**, page 71–122 (2010).
https://doi.org/10.1017/S0956796809990293

[68] Stark, K., *Mechanising Syntax with Binders in Coq*, Ph.D. thesis, Saarland University (2020).

[69] Stark, K., S. Schäfer and J. Kaiser, *Autosubst 2: Reasoning with multi-sorted de bruijn terms and vector substitutions*, 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019 (2019).

[70] Street, R., *Elementary cosmoi i*, in: G. M. Kelly, editor, *Category Seminar*, pages 134–180, Springer Berlin Heidelberg, Berlin, Heidelberg (1974), ISBN 978-3-540-37270-7.

[71] Swierstra, W., *Data types á la carte*, Journal of Functional Programming **18**, pages 423–436 (2008).
https://doi.org/10.1017/S0956796808006758

[72] Szlachányi, K., *Skew-monoidal categories and bialgebroids*, Advances in Mathematics **231**, pages 1694–1730 (2012), ISSN 0001-8708.
https://doi.org/https://doi.org/10.1016/j.aim.2012.06.027

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x:A}\qquad \frac{\Gamma\vdash V:A}{\Gamma\vdash \mathsf{val}\ V:\mathsf{comp}\ A}\qquad \frac{\Gamma,x_1:A_1,\ldots,x_n:A_n\vdash N:\mathsf{comp}\ B\qquad \text{for all } i<n\text{: } \Gamma,x_1:A_1,\ldots,x_i:A_i\vdash M_{i+1}:\mathsf{comp}\ A_{i+1}}{\Gamma\vdash \mathbf{let}\ x_1=M_1;\ldots;x_n=M_n\,\mathbf{in}\,N:\mathsf{comp}\ B}$$

$$\frac{\Gamma,x:A\vdash M:\mathsf{comp}\ B}{\Gamma\vdash \lambda x:A.M:A\to B}\qquad \frac{\Gamma\vdash M:\mathsf{comp}(A\to B)\qquad \Gamma\vdash N:\mathsf{comp}\ A}{\Gamma\vdash M\,@\,N:\mathsf{comp}\ B}$$

$$\frac{\text{for all } 1\le i\le n\text{: } \Gamma\vdash M_i:A_i}{\Gamma\vdash\left(C_1:\ M_1,\ldots,C_n:\ M_n\right):\mathsf{comp}\,\llparenthesis C_1:\ A_1,\ldots,C_n:\ A_n\rrparenthesis}\qquad \frac{\Gamma\vdash M:\mathsf{comp}\,\llparenthesis C_1:\ A_1,\ldots,C_n:\ A_n\rrparenthesis\qquad \Gamma,x_1:A_1,\ldots,x_n:A_n\vdash N:\mathsf{comp}\ B}{\Gamma\vdash \mathbf{case}\ M\,\mathbf{of}\left(C_1x_1,\ldots,C_nx_n\right)\Rightarrow N:\mathsf{comp}\ B}$$

$$\frac{A=\lBrace C_i:\ A_i\,\big|\,i\in I\rBrace\qquad \Gamma\vdash M:\mathsf{comp}\ A_i}{\Gamma\vdash A.C_i\,M:\mathsf{comp}\ A}\qquad \frac{\Gamma\vdash M:\lBrace C_i:\ A_i\,\big|\,i\in I\rBrace\qquad \text{for all } 1\le i\le n\text{: } \Gamma,x_i:A_i\vdash M_i:\mathsf{comp}\ B}{\Gamma\vdash \mathbf{case}\ M\,\mathbf{of}\left\{C_ix_i\Rightarrow M_i\,\big|\,i\in I\right\}N:\mathsf{comp}\ B}$$

$$\frac{}{\Gamma\vdash n:\mathbb{N}}\qquad \frac{\Gamma\vdash M:\mathsf{comp}\,\mathbb{N}}{\Gamma\vdash \mathbf{unroll}\,M:\lBrace 0:\ \llparenthesis\textvisiblespace\rrparenthesis,\ (1+):\ \mathbb{N}\rBrace}\qquad \frac{\Gamma\vdash M:\mathsf{comp}\lBrace 0:\ \llparenthesis\textvisiblespace\rrparenthesis,\ (1+):\ \mathbb{N}\rBrace}{\Gamma\vdash \mathbf{roll}\,M:\mathsf{comp}\,\mathbb{N}}\qquad \frac{\Gamma\vdash M:\mathsf{comp}\,\mathbb{N}\qquad \Gamma,x:\lBrace 0:\ \llparenthesis\textvisiblespace\rrparenthesis,\ (1+):\ A\rBrace\vdash N:\mathsf{comp}\ A}{\Gamma\vdash \mathbf{fold}\ M\ \mathbf{by}\{x\Rightarrow N\}:\mathsf{comp}\ A}$$

$$\frac{\Gamma\vdash M:\mathsf{comp}\ A\qquad \Gamma,i:A\vdash N:\mathsf{comp}\lBrace\mathbf{done}:\ B,\mathbf{continue}:\ A\rBrace}{\Gamma\vdash \mathbf{for}\ i=M\,\mathbf{do}\,N:\mathsf{comp}\ B}\qquad \frac{\Gamma,f_1:\llparenthesis\Gamma_1\rrparenthesis\to A_1,\ldots,f_n:\llparenthesis\Gamma_n\rrparenthesis\to A_n\vdash N:B\qquad \text{for all } 1\le i\le n\text{: } \Gamma\mathbin{+\!\!+}\Gamma_n\vdash M_n:A_n}{\Gamma\vdash \mathbf{let\ \ rec}\ f_1\Gamma_1:\ A_1=M_1;\ldots;f_n\Gamma_n:\ A_n=M_n\mathbf{in}N:B}$$

Fig. A.1. Type system of CBV, omitting analogous rules for value records and variants.

[73] Turi, D. and G. D. Plotkin, *Towards a mathematical operational semantics*, in: *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 280–291 (1997).
https://doi.org/10.1109/LICS.1997.614955

[74] Urban, C., *Nominal techniques in Isabelle/HOL*, J. Autom. Reason. **40**, pages 327–356 (2008).
https://doi.org/10.1007/S10817-008-9097-2

[75] Voevodsky, V., B. Ahrens, D. Grayson *et al.*, *Unimath — a computer-checked library of univalent mathematics*, available at http://unimath.org.
https://doi.org/10.5281/zenodo.10849216

[76] Weirich, S., B. A. Yorgey and T. Sheard, *Binders unbound*, SIGPLAN Not. **46**, page 333–345 (2011), ISSN 0362-1340.
https://doi.org/10.1145/2034574.2034818

[77] Érdi, G., *Generic description of well-scoped, well-typed syntaxes* (2018). 1804.00119.
https://arxiv.org/abs/1804.00119

A Detailed case study: the Call-by-Value λ-calculus

We evaluate applying MAST to detailed calculi that contain simply-typed features that might appear in semantics work, emphasising modularity in both syntax and semantics. The development is similar in spirit to Swierstra's á la carte methodology [71,33], but it also provides semantic substitution lemmata. We use this opportunity to also summarise the standard denotational semantics for these features.

A.1 The full calculus

Fig 1 presents the abstract syntax of all the features we will consider without explicating their typing judgements nor their binding structure. Fig A.1 presents the typing judgements. Our base calculus include a construct for sequencing, which evaluates the intermediate results in order, binding them to variables. Extending the calculus with records adds tuples of labelled fields, which we eliminate with a pattern matching construct. Extending the calculus with variants adds tagged sums, which we eliminate with a pattern matching construct. Extending the calculus with natural numbers adds natural number literals as values, the iso-recursive constructor **roll** and deconstructor **unroll**, and a bounded iteration eliminator. We further extend the calculus with unbounded iteration $\mathbf{for}\ i = M\ \mathbf{do}\ N$, which initialises i to M, and then iterates N until it is **done**.

A.2 Simple types á la carte

To develop these calculi the and their models fully modularly, we first need to treat their sets of types modularly. We will use the classical á la carte methodology, for initial algebra semantics in **Set**, to mix and choose the collection of simple types we work with in each case. We will work with respect to an ordinary signature functor $\mathbf{L} : \mathbf{Set} \to \mathbf{Set}$ that has an initial algebra $\mathbf{L}(\mu\mathbf{L}) \to \mu\mathbf{L}$. This functor specifies the signature for the simple types given by $\mathsf{Type}_{\mathbf{L}} \coloneqq \mu\mathbf{L}$. For any set Type, whether inductively given by such a signature functor or not, define the sorting system $\textsc{cbv}_{\mathsf{Type}}$:

$$\textsc{cbv}_{\mathsf{Type}}\text{-}\mathsf{sort} \coloneqq \{A, \mathsf{comp}\,A | A \in \mu\mathsf{Type}\} \qquad A \vDash \mathsf{bnd} \qquad \mathsf{comp}\,A \nvDash \mathsf{bnd} \qquad (A \in \mu\mathsf{Type})$$

We will also work with respect to a $\textsc{mast}$ $\textsc{cbv}_{\mathsf{Type}}$-signature functor $\mathbf{O}$, and define an $\mathbf{O}$-monoid, which by the Special Representation Thm 7.4 satisfies the substitution lemma.

Example A.1 In the simplest case, all we have are base types. The signature functor for types is the constant functor $\underline{\mathsf{Base}} : \mathbf{Set} \to \mathbf{Set}$, $\underline{\mathsf{Base}}\,X \coloneqq \mathsf{Base}$. It has the identity function $\mathsf{id} : \mathsf{Base} \to \mathsf{Base}$ as its initial algebra, thus $\mathsf{Base} = \mu\,\underline{\mathsf{Base}} =: \mathsf{Type}$. Summarising, the set of types is the set of base types.

Example A.2 To accommodate function types, consider the functor $\mathsf{FunTy} \coloneqq X \mapsto X \times X : \mathbf{Set} \to \mathbf{Set}$. Then:

$$\underline{\mathsf{Base}} \amalg \mathsf{FunTy}\,X \cong \{\!\!|\beta | \beta \in \mathsf{Base}|\!\!\} \amalg \{\!\!|(\to) : X \times X|\!\!\}$$

Therefore, the simple types of the $\textsc{stlc}$ is the initial algebra $[\![-]\!] : (\underline{\mathsf{Base}} \amalg \mathsf{FunTy})\mathsf{Type} \to \mathsf{Type}$ qua:

$$[\![\beta]\!] \coloneqq \beta \qquad [\![(\to)]\!]\,(A, B) \coloneqq A \to B$$

Example A.3 Next, we deal with records and variants uniformly. Let X be a set. A *row* in X is a function $\left(C_i \mapsto x_i\right)_{i \in I}$ from a finite set of *field/constructor labels* $\left\{C_i | i \in I\right\}$ to X. Letting Label be the set of field labels, define $\mathsf{Row} : \mathbf{Set} \to \mathbf{Set}$ by $\mathsf{Row}\,X \coloneqq \coprod_{I \subseteq_{\mathrm{fin}} \mathsf{Label}} X^I$. The functors for record and variant types are: $\mathsf{RecordTy}, \mathsf{VariantTy} \coloneqq \mathsf{Row} : \mathbf{Set} \to \mathbf{Set}$.

The collection of types $\textsc{cbv}\mathsf{Type}_{\mathsf{Base}}$ defined inductively in Fig 1 is the initial algebra for the functor:

$$\mathsf{FullCBV} \coloneqq \underline{\mathsf{Base}} \amalg ((\to) : \mathsf{FunTy}) \amalg (\llparenthesis - \rrparenthesis : \mathsf{RecordTy}) \amalg (\{\!\!|-|\!\!\} : \mathsf{VariantTy}) \amalg (\mathbb{N} : \mathsf{NatTy}) : \mathbf{Set} \to \mathbf{Set}$$

We want to work with more than $2^4 = 16$ collections of types, given by various restrictions of this collection, depending on the typing needs of each language fragment. To deal with such sub-collections, we define a *typing need* to be a signature functor $\mathbf{R} : \mathbf{Set} \to \mathbf{Set}$. A *fulfillment* of a typing need $\mathbf{R}$ by a signature functor $\mathbf{L} : \mathbf{Set} \to \mathbf{Set}$ is a natural transformation $\alpha : \mathbf{R} \to \mathbf{L}$, which we write as $\mathbf{L} \vDash \alpha : \mathbf{R}$. We will work with the typing needs given by each type-constructor FunTy, $\mathsf{RecordTy}$, $\mathsf{VariantTy}$ and $\mathsf{NatTy} \coloneqq \mathbb{1}$. We will also sometimes only need the existence of specific record/variant types, and use these typing needs:

- When we need the unit type, we impose the typing need $\mathsf{EmptyRecord} \coloneqq \mathbb{1}$, typically fulfilled by the coproduct injection $\mathbf{L} \amalg (\llparenthesis\_\rrparenthesis : \mathbb{1}) \vDash \llparenthesis\_\rrparenthesis : \mathsf{EmptyRecord}$.
- When we need to deconstruct natural numbers, we impose the typing need:

  $$\mathsf{NatConstVariant} \coloneqq (0 : \mathbb{1}) \amalg ((1+) : \mathsf{Id})$$

  We will typically fulfill it by the coproduct injection $\mathbf{L} \amalg (\iota_2 : \{\!\!|0 : \llparenthesis\_\rrparenthesis, (1+) : -|\!\!\}) \vDash \iota_2 : \mathsf{NatConstVariant}$. This fulfillment will allow us to interpret the variant types $\{\!\!|0 : \llparenthesis\_\rrparenthesis, (1+) : -|\!\!\}$.
- Similarly, to type the bodies of while-loops, we impose the typing need:

  $$\mathsf{NatConstVariant} \coloneqq (\mathbf{done} : \mathsf{Id}) \times (\mathbf{continue} : \mathsf{Id})$$

We only use fulfillments given by coproduct injections $\coprod_{i \in I}(C : \mathbf{L}_i) \vDash C_j : \mathbf{L}_j$. We use general natural transformations for the simplicity in defining needs and fulfillments.

A fullfillment $\mathbf{L} \vDash \alpha : \mathbf{R}$ induces a functor from the category of $\mathbf{L}$-algebras to the category of $\mathbf{R}$-algebras:

$$\left(\mathbf{L}A \xrightarrow{a} A\right) \mapsto \left(\mathbf{R}A \xrightarrow{\alpha} \mathbf{L}A \xrightarrow{a} A\right) \qquad \text{since}$$
$$\left(h : (A, a) \to (B, b)\right)\big) \mapsto \left(h : (A, a \circ \alpha) \to (B, b \circ \alpha)\right)$$

$$\begin{tikzcd} \mathbf{L}A \arrow[r, "a"] \arrow[d, "Lh"'] \arrow[dr, phantom, "="] & A \arrow[d, "h"] \\ \mathbf{L}B \arrow[r, "b"'] & B \end{tikzcd} \implies \begin{tikzcd} \mathbf{R}A \arrow[r, "\alpha"] \arrow[d, "Rh"'] \arrow[dr, phantom, "\overset{\alpha\text{-nat.}}{=}"] & \mathbf{L}A \arrow[r, "a"] \arrow[d, "Lh"'] \arrow[dr, phantom, "="] & A \arrow[d, "h"] \\ \mathbf{R}B \arrow[r, "\alpha"'] & \mathbf{L}B \arrow[r, "b"'] & B \end{tikzcd}$$

Therefore, a fulfillment lets us interpret the type constructors provided by the typing need $\mathbf{R}$ as operations on the types provided by $\mathbf{L}$.

A.3 The substitution monoids

Let Type be a set whose elements represent types. A *strong-monad model* $\left(\mathcal{C}, [\![-]\!], \mathrm{T}\right)$ consists of:

- A locally-small Cartesian category $\mathcal{C}$ with chosen finite products.
- An interpretation function $[\![-]\!] : \mathsf{Type} \to \mathcal{C}$.
- A strong monad [1] T over $\mathcal{C}$, i.e., an assignment of:
  - an object $\mathrm{T}x$ to every $x \in \mathcal{C}$;
  - a morphism $\mathsf{return}_x : x \to \mathrm{T}x$ to every $x \in \mathcal{C}$;
  - a morphism ${\gg\!=}_{a,x,y} f : a \times \mathrm{T}x \to \mathrm{T}y$ to every $a, x, y \in \mathcal{C}$ and $f : a \times x \to \mathrm{T}y$;

  satisfying the following four equations [58], phrased using the cartesian monoidal structure $(\mathcal{C}, (\times), \mathbb{1}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r})$:

$$\begin{tikzcd}[column sep=huge] \mathbb{1} \times \mathrm{T}x \arrow[r, bend left, "{\gg\!=}\left(\mathbb{1} \times x \xrightarrow{\boldsymbol{\pi}_2} x \xrightarrow{\mathrm{return}} \mathrm{T}x\right)"] \arrow[r, bend right, "\boldsymbol{\pi}_2"'] \arrow[r, phantom, "\substack{\text{strong monad}\\ \text{cartesian unit}\\ =}"] & \mathrm{T}x \end{tikzcd} \qquad \begin{tikzcd} a \times \mathrm{T}x \arrow[d, "u \times \mathsf{id}"'] \arrow[dr, bend left, "{\gg\!=}\left(a \times x \xrightarrow{u \times \mathsf{id}} b \times x \xrightarrow{f} \mathrm{T}y\right)"] \arrow[dr, phantom, "\substack{\text{strong monad}\\ \text{naturality}}"] & \\ b \times \mathrm{T}x \arrow[r, "{\gg\!=} f"', "="] & Ty \end{tikzcd} \qquad \begin{tikzcd} & a \times x \arrow[dl, bend right, "\mathsf{id} \times \mathsf{return}"'] \arrow[d, bend left, "f"] \\ a \times \mathrm{T}x \arrow[r, "{\gg\!=} f"', "\substack{\text{strong monad}\\ \text{monadic unit}\\ =}"] & \mathrm{T}y \end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge] (a \times b) \times \mathrm{T}x \arrow[rr, "{\gg\!=}\left((a \times b) \times x \xrightarrow{\mathbf{a}} a \times (b \times x) \xrightarrow{\mathsf{id} \times f} a \times \mathrm{T}y \xrightarrow{\gg= g} \mathrm{T}z\right)"] \arrow[d, "\mathbf{a}"'] \arrow[drr, phantom, "\substack{\text{strong monad}\\ \text{associativity}\\ =}"] & & \mathrm{T}z \\ a \times (b \times \mathrm{T}x) \arrow[rr, "\mathsf{id} \times ({\gg\!=} f)"'] & & a \times \mathrm{T}y \arrow[u, "{\gg\!=} g"'] \end{tikzcd}$$

Each such strong-monad model induces a substitution monoid $\mathbf{M}$ in $\textsc{cbv}_{\mathsf{Type}}$-structures. First, let:

$$[\![\Gamma]\!] \coloneqq [\![-]\!]^{\mathrm{Env}}_{\Gamma} \coloneqq \prod_{(x:A) \in \Gamma} [\![A]\!] \qquad \left[\!\!\left[\Gamma \xrightarrow{\rho} \Delta\right]\!\!\right] \coloneqq [\![-]\!]^{\mathrm{Env}}_{\rho} : [\![\Delta_1]\!] \xrightarrow{(\boldsymbol{\pi}_y)_{(y:B) \in \Delta_2}} [\![\Delta_2]\!]$$

And then define the carrier for $\mathbf{M}$ by:

$$[\![\mathsf{comp}\, A]\!] \coloneqq \mathrm{T}\, [\![A]\!] \qquad \underline{\mathbf{M}}_s \Gamma \coloneqq \mathcal{C}([\![\Gamma]\!], [\![s]\!]) \qquad \underline{\mathbf{M}}_s \left(\Gamma \xleftarrow{\rho} \Delta\right) : \left([\![\Gamma]\!] \xrightarrow{f} [\![s]\!]\right) \mapsto \left([\![\Delta]\!] \xrightarrow{[\![\rho]\!]} [\![\Gamma]\!] \xrightarrow{f} [\![s]\!]\right)$$

I.e., $\underline{\mathbf{M}}_{\mathsf{comp}\,A}\Gamma \coloneqq \mathcal{C}([\![\Gamma]\!], \mathrm{T}\,[\![A]\!])$ so the semantics of computations are Kleisli arrows for the monad T. The monoid's unit interprets variables by projecting the appropriate component:

$$\mathsf{env} : \mathbb{I} \to \underline{\mathbf{M}} \qquad \mathsf{env}_{B,\Gamma}\, y \coloneqq \left([\![\Gamma]\!] = \left(\prod\nolimits_{(x:A) \in \Gamma} [\![A]\!]\right) \xrightarrow{\boldsymbol{\pi}_y} [\![B]\!]\right)$$

$$\begin{tikzcd}
(\mathbb{I}\otimes\underline{\mathbf{M}})_s\Gamma \ni \left[(x : A)\in\Delta, \left(\overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta]\!]}\right)\right]_\Delta \arrow[r, mapsto, "\mathsf{env}\otimes\mathsf{id}"] \arrow[dr, mapsto, "\ell"'] & \left[[\![\Delta]\!]\xrightarrow{\pi_x}[\![A]\!], \left(\overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta]\!]}\right)\right]_\Delta \in (\underline{\mathbf{M}}\otimes\underline{\mathbf{M}})_s\Gamma \arrow[d, mapsto, "-[-]_{\mathbf{M}}"] \\
& \left([\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta]\!]\xrightarrow{\pi_x}[\![A]\!]\right)\in\underline{\mathbf{M}}_s\Gamma
\end{tikzcd}$$

Fig. A.2. Substitution monoid left neutrality

To define substitution, we use the isomorphism which internalises tupling:

$$\overline{(-)} \coloneqq \left(\pi_y\circ(-)\right)_{(y:B)\in\Delta} : \mathcal{C}([\![\Gamma]\!], [\![\Delta]\!]) \xrightarrow{\cong} \left(\prod_{(y:B)\in\Delta}\mathcal{C}([\![\Gamma]\!], [\![B]\!])\right) = \underline{\mathbf{M}}^{\mathrm{Env}}_{\Delta}\Gamma$$

We express the functorial action of $\underline{\mathbf{M}}^{\mathrm{Env}}$ through this isomorphism as follows:

$$\underline{\mathbf{M}}^{\mathrm{Env}}_{\Delta_1\xrightarrow{\rho}\Delta_2,\Gamma} : \left(\overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta_1]\!]}\right) \mapsto \left(\overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta_1]\!]\xrightarrow{[\![\rho]\!]}[\![\Delta_2]\!]}\right)$$

We then define substitution $-[-]_{\mathbf{M}} : \mathbf{M}\otimes\mathbf{M}\to\mathbf{M}$ by pre-composition:

$$\left([\![\Delta]\!]\xrightarrow{f}[\![s]\!]\right)\left[\overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta]\!]}\right]_{\mathbf{M},s,\Gamma} \coloneqq \left([\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta]\!]\xrightarrow{f}[\![s]\!]\right)$$

It respects the coend quotient. Indeed, for $\rho : \Delta_1\to\Delta_2$, chase a generic element $(f,\theta)\in\underline{\mathbf{M}}_s\Delta_2\times\underline{\mathbf{M}}^{\mathrm{Env}}_{\Delta_1}\Gamma$:

$$\begin{tikzcd}
& \left([\![\Delta_1]\!]\xrightarrow{[\![\rho]\!]}[\![\Delta_2]\!]\xrightarrow{f}[\![s]\!], \overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta_1]\!]}\right)\in\underline{\mathbf{M}}_s\Delta_1\times\underline{\mathbf{M}}^{\mathrm{Env}}_{\Delta_1}\Gamma \arrow[dr, mapsto, "-[-]_{\mathbf{M}}"] & \\
\underline{\mathbf{M}}_s\Delta_2\times\underline{\mathbf{M}}^{\mathrm{Env}}_{\Delta_1}\Gamma\ni\left([\![\Delta_2]\!]\xrightarrow{f}[\![s]\!], \overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta_1]\!]}\right) \arrow[ur, mapsto, "\underline{\mathbf{M}}_s\rho\times\mathsf{id}"] \arrow[dr, mapsto, "\mathsf{id}\times\underline{\mathbf{M}}^{\mathrm{Env}}_{\rho,\Gamma}"'] & & \left([\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta_1]\!]\xrightarrow{[\![\rho]\!]}[\![\Delta_2]\!]\xrightarrow{f}[\![s]\!]\right)\in\underline{\mathbf{M}}_s\Gamma \\
& \left([\![\Delta_2]\!]\xrightarrow{f}[\![s]\!], \overline{[\![\Gamma]\!]\xrightarrow{\theta}[\![\Delta_1]\!]\xrightarrow{[\![\rho]\!]}[\![\Delta_2]\!]}\right)\in\underline{\mathbf{M}}_s\Delta_1\times\underline{\mathbf{M}}^{\mathrm{Env}}_{\Delta_1}\Gamma \arrow[ur, mapsto, "-[-]_{\mathbf{M}}"'] &
\end{tikzcd}$$

It is natural in Γ since $[\![\Gamma]\!]$ is natural in Γ, and pre-/post-composition is natural. Summarising, we have construct the data of a substitution monoid:

$$\mathbb{I}\xrightarrow{\mathsf{env}}\underline{\mathbf{M}}\xleftarrow{-[-]_{\mathbf{M}}}\underline{\mathbf{M}}\otimes\underline{\mathbf{M}}$$

This structure satisfies the monoid axioms. Straightforward calculation shows the monoid axioms (see Figures A.2, A.3, and A.4).

A.4 The CBV *customisation menu*

We will now consider each of the following fragments of the full CBV calculus. Fig A.5 list the constructs in each fragment, which fragments of the type system they require, and what model structure they need.

$$\begin{tikzcd}
\underline{\mathbf{M}}_s\Gamma \ni \left(\llbracket\Gamma\rrbracket \xrightarrow{f} \llbracket A\rrbracket\right) \arrow[rr, mapsto, "\mathbf{r}'"] \arrow[d, equal] & & \left[\llbracket\Gamma\rrbracket \xrightarrow{f} \llbracket A\rrbracket\,, (x)_{(x:A)\in\Gamma}\right]_{\Gamma} \in (\underline{\mathbf{M}} \otimes \mathbb{I})_s\Gamma \arrow[d, "\mathsf{id} \otimes \mathsf{env}"] \\
\llbracket\Gamma\rrbracket \arrow[r, "f"] \arrow[dd, bend right=60, "{\underline{\mathbf{M}}_s\Gamma \ni (\boldsymbol{\pi}_x)_{x\in\Gamma}}"'] \arrow[dd, equal, bend left] & \llbracket A\rrbracket & \left[\llbracket\Gamma\rrbracket \xrightarrow{f} \llbracket A\rrbracket\,, \left(\overline{\llbracket\Gamma\rrbracket \xrightarrow{(\boldsymbol{\pi}_x)_{(x:A)\in\Gamma}} \llbracket\Gamma\rrbracket}\right)\right]_{\Gamma} \in (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}})_s\Gamma \arrow[dl, mapsto, "{-[-]_{\mathbf{M}}}"] \\
\arrow[r, phantom, "\overset{\text{product}}{=}"] & f \arrow[u, phantom, "="] & \\
\llbracket\Gamma\rrbracket \arrow[uur, bend right, "f"'] & &
\end{tikzcd}$$

Fig. A.3. Substitution monoid right neutrality

$$\begin{tikzcd}
((\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \otimes \underline{\mathbf{M}})_s\Gamma \ni \left[\left[\llbracket\Delta\rrbracket \xrightarrow{f} \llbracket A\rrbracket\,, \left(\overline{\llbracket\Xi\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket}\right)\right]_{\Delta}\,, \left(\overline{\llbracket\Gamma\rrbracket \xrightarrow{\sigma} \llbracket\Xi\rrbracket}\right)\right]_{\Xi} \arrow[d, mapsto, "{(-[-]_{\mathbf{M}}) \otimes \mathsf{id}}"'] \arrow[dddd, mapsto, bend left=60, "\mathbf{a}"] \\
(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}})_s\Gamma \ni \left[\llbracket\Xi\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket \xrightarrow{f} \llbracket A\rrbracket\,, \left(\overline{\llbracket\Gamma\rrbracket \xrightarrow{\sigma} \llbracket\Xi\rrbracket}\right)\right]_{\Xi} \arrow[d, mapsto, "{-[-]_{\mathbf{M}}}"'] \\
\left(\llbracket\Gamma\rrbracket \xrightarrow{\sigma} \llbracket\Xi\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket \xrightarrow{f} \llbracket A\rrbracket\right) \in \underline{\mathbf{M}}_s\Gamma \\
\left[\llbracket\Delta\rrbracket \xrightarrow{f} \llbracket A\rrbracket\,, \left(\overline{\llbracket\Gamma\rrbracket \xrightarrow{\sigma} \llbracket\Xi\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket}\right)\right]_{\Delta} \in (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}})_s\Gamma \arrow[u, mapsto, "{-[-]_{\mathbf{M}}}"] \\
\left[\llbracket\Delta\rrbracket \xrightarrow{f} \llbracket A\rrbracket\,, \left(\left[\llbracket\Xi\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket \xrightarrow{\boldsymbol{\pi}_x} \llbracket A\rrbracket\,, \llbracket\Gamma\rrbracket \xrightarrow{\sigma} \llbracket\Xi\rrbracket\right]_{\Xi}\right)_{(x:A)\in\Delta}\right]_{\Delta} \in (\underline{\mathbf{M}} \otimes (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}))_s\Gamma \arrow[u, mapsto, "{\mathsf{id} \otimes (-[-]_{\mathbf{M}})}"]
\end{tikzcd}$$

Fig. A.4. Substitution monoid associativity

We define each fragment, and then, in §A.5, explain how to combine fragments together into one calculus and its model class of interest. Together, these combinations describe $2^7 = 128$ different calculi, and their denotational semantics. Thanks to MAST and the Special Representation Thm 7.4, the substitution operation and denotational semantics for each combination satisfy a substitution lemma. For each fragment, we specify

- a typing need $\mathbf{R}$.

Then, we fix any fulfillment $\mathbf{L} \vDash \alpha : \mathbf{R}$. It induces a sorting system $\mathrm{CBV}_{\mathbf{L}} := \mathrm{CBV}_{\mathsf{Type}_{\mathbf{L}}}$. This sorting system has a type constructor for each $\mathbf{R}$ operator. We then further specify:

- a $\mathrm{CBV}_{\mathbf{L}}$-signature functor $\mathbf{O}$ describing the syntactic constructs in this fragment;
- additional structure or properties we require of the substitution monoid $\mathbf{M}$. These requirements will typically be requirements of its type interpretation $\llbracket - \rrbracket : \mathsf{Type}_{\mathbf{L}} \to \mathbf{M}$, typically in the form of universal properties using the postulated categorical structure on $\mathbf{M}$.

name	syntactic constructs	typing needs	additional model needs
base	returning a value: val		strong monad over a Cartesian category
sequential	sequencing: $\mathbf{let}$		
functions	abstraction and application $(\lambda x.\,:\,A), (@)$	function $(\to)$	Kleisli exponentials
records	constructors and pattern match $\left(C_1 : -, \ldots, C_n : -\right)$ $\mathbf{case} - \mathbf{of}\ \left(C_1 x_1, \ldots, C_n x_n\right)\ \Rightarrow -$	record $\llparenthesis C_i\ :\ - \middle\vert i \in I \rrparenthesis$	
variants	constructors and pattern match $A.C_i -,\ \ \mathbf{case} - \mathbf{of}\ \left\{C_i x_i \Rightarrow - \middle\vert i \in I\right\}$	variant $\{\!\vert C_i\ :\ - \middle\vert i \in I \vert\!\}$	distributive category
natural numbers	the zero and successor constructors, literals, empty record, (de)constructors, and bounded iteration, the pattern matching $0, (1+), n, (\textvisiblespace)\ , \mathbf{unroll}, \mathbf{roll}$ $\mathbf{fold} - \mathbf{by}\ \{x \Rightarrow\ -\}$ $\mathbf{case} - \mathbf{of}\ \{0 \Rightarrow\ -, 1{+}x \Rightarrow\ -\}$	naturals, empty record, and the variants $\mathbb{N}$ $\left\{\!\!\left[\begin{matrix}0 :\ \llparenthesis \textvisiblespace \rrparenthesis\ , \\ (1+) :\ -\end{matrix}\right]\!\!\right\}$	binary coproducts distributed over by the products, and a natural numbers object
while	the constructors, unbounded iteration $\mathbf{done}, \mathbf{continue}, \mathbf{for}\ i = - \ \mathbf{do}\ -$	the variants $\left\{\!\!\left[\begin{matrix}\mathbf{done}\ :\ -, \\ \mathbf{continue}\ :\ -\end{matrix}\right]\!\!\right\}$	binary coproducts, distributive products, and the monad has a complete Elgot structure
recursion	relevant record constructor, function application, recursion $(-)\ , (@), \mathbf{let\ rec}$	the functions $(\llparenthesis x_i\ :\ - \middle\vert i \in I \rrparenthesis \to)$	uniform parameterised monadic fixed-points, Kleisli exponentials

Fig. A.5. A customisation menu of cbv fragments

We then fix such a substitution monoid $\mathbf{M}$, and further specify:

- an $\mathbf{O}$-algebra structure for $\mathbf{M}$ making it an $\mathbf{O}$-monoid.

Base fragment

The typing need here is empty, i.e., the initial functor $\mathbb{0}$. Every type-signature functor $\mathbf{L}$ has a unique fulfillment $\mathbf{L} \vDash [] : \mathbb{0}$ given by the unique natural transformation $[] : \mathbb{0} \to \mathbf{L}$. The base calculus has, for each type $A \in \mathsf{Type}_{\mathbf{L}}$, one operator coercing A-values to A-computations. Its binding signature functor and its derived strength are:

$$\mathsf{Base}\, X \coloneqq \coprod_{A \in \mathsf{Type}_{\mathbf{L}}} (\mathsf{val}_A : \underset{\mathsf{comp}\, A}{\looparrowright} (X \mathbin{@} A)) \qquad \mathsf{str}^{\mathsf{Base}}_{P,A,\Gamma} \left[\mathsf{val}_A(p \in P_A \Delta), \theta \in A^{\mathrm{Env}}_{\Delta}\Gamma\right] \coloneqq \mathsf{val}_A[p, \theta]$$

Given a strong-monad model $\mathbf{M} \coloneqq \big(\mathcal{C}, \llbracket - \rrbracket, \mathrm{T}\big)$, we require no additional semantic structure of it. Define its Base-algebra structure by:

$$\mathbf{M} \left\llbracket \mathsf{val}_A \; \llbracket \Gamma \rrbracket \xrightarrow{f} \llbracket A \rrbracket \right\rrbracket \coloneqq \left(\llbracket \Gamma \rrbracket \xrightarrow{f} \llbracket A \rrbracket \xrightarrow{\mathrm{return}} \mathrm{T}\, \llbracket A \rrbracket \right)$$

and validate the compatibility axiom:

$$\begin{tikzcd}
 & \mathsf{val}_A\,[f, \theta] \in \big(\underline{\mathsf{Base}}(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}})\big) \arrow[dr, mapsto, "\underline{\mathsf{Base}}(-[-]_{\mathbf{M}})"] & \\
\begin{array}{c}\big((\underline{\mathsf{Base}\,\mathbf{M}}) \otimes_{\bullet} \mathsf{env}^{\mathbf{M}}\big)_{\mathsf{comp}\,A,\Gamma} \ni \\ \left[\mathsf{val}_A \left(\llbracket \Delta \rrbracket \xrightarrow{f} \llbracket A \rrbracket \right), \llbracket \Gamma \rrbracket \xrightarrow{\theta} \llbracket \Delta \rrbracket \right]\end{array} \arrow[ur, mapsto, "\mathsf{str}"] \arrow[d, mapsto, "\llbracket - \rrbracket \otimes \mathsf{id}"'] & & \begin{array}{c}\mathsf{val}_A \left(\llbracket \Gamma \rrbracket \xrightarrow{\theta} \llbracket \Delta \rrbracket \xrightarrow{f} \llbracket A \rrbracket \right) \\ \in \big(\underline{\mathsf{Base}\mathbf{M}}\big)\end{array} \arrow[d, mapsto, "\llbracket - \rrbracket"] \\
\left[\llbracket \Delta \rrbracket \xrightarrow{f} \llbracket A \rrbracket \xrightarrow{\mathrm{return}} \mathrm{T}\, \llbracket A \rrbracket, \theta \right] \in \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[rr, mapsto, "-[-]_{\mathbf{M}}"'] & & \left(\llbracket \Gamma \rrbracket \xrightarrow{\theta} \llbracket \Delta \rrbracket \xrightarrow{f} \llbracket A \rrbracket \xrightarrow{\mathrm{return}} \mathrm{T}\, \llbracket A \rrbracket \right) \in \underline{\mathbf{M}}_{\mathsf{comp}\,A}\Gamma
\end{tikzcd}$$

Sequential fragment

The typing need here is also empty, and similarly has a unique fulfillment. The sequential fragment has, for every tuple of types $A_0, \dots, A_n, B \in \mathsf{Type}_{\mathbf{L}}$, one sequencing operator $\mathbf{let}\; x_0 = M_0; \dots; x_n = M_n \;\mathbf{in}\; N$:

$$\mathsf{Seq}\, X \coloneqq \coprod_{n \in \mathbb{N}} \coprod_{A_0, \dots, A_n, B \in \mathsf{Type}_{\mathbf{L}}} \left(\begin{array}{l} (\mathbf{let}\; x_0 : A_0 = \_; \dots; x_n : A_n = \_ \;\mathbf{in}\; \_ : B) : \\ \qquad \underset{\mathsf{comp}\, B}{\looparrowright} \left(\begin{array}{c} \left(\prod_{i=0}^{n} [x_0 : A_1, \dots, x_{i-1} : A_{i-1}] \rhd X \mathbin{@} \mathsf{comp}\, A_i \right) \\ \times([x_0 : A_1, \dots, x_n : A_n] \rhd X \mathbin{@} \mathsf{comp}\, B) \end{array} \right) \end{array} \right)$$

with its induced strength:

$$\mathsf{str}^{\mathsf{Seq}}_{P,A,\Gamma}\left[\begin{array}{l}\mathbf{let}\; x_0 : A_0 = (p_0 \in P_{\mathsf{comp}\,A_0}\Delta)\\ \quad x_1 : A_1 = (p_1 \in P_{\mathsf{comp}\,A_1}(\Delta, x_0 : A_0))\\ \qquad\vdots\\ \quad x_n : A_n = (p_n \in P_{\mathsf{comp}\,A_n}(\Delta, x_0 : A_0, \dots, x_{n-1} : A_{n-1}))\\ \mathbf{in}\; (q \in P_{\mathsf{comp}\,B}(\Delta, x_0 : A_0, \dots, x_n : A_n)), \theta \in A^{\mathsf{Env}}_{\Delta}\Gamma\end{array}\right]_{\Delta}$$

$$:= \begin{array}{l}\mathbf{let}\; x_0 : A_0 = \left[p_0, \theta\right]_{\Delta}\\ \quad x_1 : A_1 = \left[p_1, \theta + \left(x_0 : \mathsf{env}_A\, x_0\right)\right]_{\Delta, x_0 : A_0}\\ \qquad\vdots\\ \quad x_n : A_n = \left[p_n, \theta + \left(x_0 : \mathsf{env}_A\, x_0, \dots, x_{n-1} : \mathsf{env}_A\, x_{n-1}\right)\right]_{\Delta, x_0 : A_0, \dots, x_{n-1} : A_{n-1}}\\ \mathbf{in}\; \left[q \in P_{\mathsf{comp}\,B}, \theta + \left(x_0 : \mathsf{env}_A\, x_0, \dots, x_n : \mathsf{env}_A\, x_n\right)\right]_{\Delta, x_0 : A_0, \dots, x_n : A_n}\end{array}$$

Given a strong-monad model $\mathbf{M} := \left(\mathcal{C}, [\![-]\!], \mathrm{T}\right)$, we require no additional semantic structure of it. We use the monadic bind to interpret the **let** construct, and to ease dealing with the intermediate results bound to $x_1, \dots, x_n$, we use the following derived semantic structure. The *strength* of the monad is given by:

$$\mathsf{str}_{a,b} : a \times \mathrm{T}b \xrightarrow{\gg\!=_{a,b,a\times b}\mathsf{id}} \mathrm{T}(a \times b)$$

We use it to define, for every $f : a \times x \to \mathrm{T}y$ a morphism that keeps the intermediate result:

$$\lessdot f : a \times x \xrightarrow{(\pi_2, f)} x \times \mathrm{T}y \xrightarrow{\mathsf{str}} \mathrm{T}(x \times y) \qquad\qquad \gg\!=\!\lessdot f : a \times \mathrm{T}x \xrightarrow{\gg\!=(\lessdot f)} \mathrm{T}(x \times y)$$

Define the Seq-algebra structure as follows. Given $A_0, \dots, A_n, B$, let:

$$\Delta := [x_0 : A_0, \dots, x_n : A_n] \qquad \Delta_i := [x_0 : A_0, \dots, x_{i-1} : A_{i-1}] \qquad\qquad \text{for all } i = 1, \dots, n$$

and then define, suppressing canonical isomorphisms such as $\Delta_i \times A_i \cong \Delta_{i+1}$:

$$\left[\!\!\left[\begin{array}{l}\mathbf{let}\; x_0 : A_0 = [\![\Gamma]\!] \xrightarrow{f_0} \mathrm{T}\,[\![A_0]\!]\\ \quad x_1 : A_1 = [\![\Gamma]\!] \times [\![\Delta_1]\!] \xrightarrow{f_1} \mathrm{T}\,[\![A_1]\!]\\ \qquad\vdots\\ \quad x_n : A_n = [\![\Gamma]\!] \times [\![\Delta_n]\!] \xrightarrow{f_n} \mathrm{T}\,[\![A_n]\!]\; \mathbf{in}\; [\![\Gamma + \Delta]\!] \xrightarrow{g} \mathrm{T}\,[\![B]\!] : B\end{array}\right]\!\!\right] :=$$

$$[\![\Gamma]\!] \xrightarrow{(\mathsf{id}, f_0)} [\![\Gamma]\!] \times \mathrm{T}\,[\![\Delta_1]\!] \xrightarrow{(\mathsf{id}, \gg\!=\!\lessdot f_1)} [\![\Gamma]\!] \times \mathrm{T}\,[\![\Delta_2]\!] \to \cdots \to [\![\Gamma]\!] \times \mathrm{T}\,[\![\Delta_n]\!] \xrightarrow{(\mathsf{id}, \gg\!=\!\lessdot f_n)} [\![\Gamma]\!] \times \mathrm{T}\,[\![\Delta]\!] \xrightarrow{\gg\!= g} \mathrm{T}\,[\![B]\!]$$

To show the compatibility condition for this algebra with the substitution monoid structure, take any:

$$\left[\begin{array}{l}\mathbf{let}\; x_0 : A_0 = [\![\Xi]\!] \xrightarrow{f_0} \mathrm{T}\,[\![A_0]\!]\\ \quad x_1 : A_1 = [\![\Xi]\!] \times [\![\Delta_1]\!] \xrightarrow{f_1} \mathrm{T}\,[\![A_1]\!]\\ \qquad\vdots\\ \quad x_n : A_n = [\![\Xi]\!] \times [\![\Delta_n]\!] \xrightarrow{f_n} \mathrm{T}\,[\![A_n]\!]\\ \mathbf{in}\; [\![\Xi]\!] \times [\![\Delta]\!] \xrightarrow{g} \mathrm{T}\,[\![B]\!], [\![\Gamma]\!] \xrightarrow{\theta} [\![\Xi]\!]\end{array}\right]_{\Xi} \in ((\mathsf{Seq}\,\mathbf{M}) \otimes \mathbf{M})_{\mathsf{comp}\,B}\Gamma$$

The compatibily condition then amounts to the following diagram:

$$\begin{tikzcd}
 & & & \llbracket\Gamma\rrbracket\times \mathrm{T}\,\llbracket\Delta_i\rrbracket \arrow[r, bend left, "{\left(\mathsf{id}, \gg\!=\!\lessdot\left(f_i\circ(\theta\times\mathsf{id})\right)\right)}"] \arrow[r, bend right, "{\left(\mathsf{id}, \gg\!=\left((\lessdot f_i)\circ(\theta\times\mathsf{id})\right)\right)}"'] \arrow[r, phantom, "\overset{(*)}{=}"] \arrow[dddd, "\theta\times\mathsf{id}"'] & \llbracket\Gamma\rrbracket\times \mathrm{T}\,\llbracket\Delta_{i+1}\rrbracket \arrow[dr] \arrow[dddd, "\theta\times\mathsf{id}"] & & & \\
\llbracket\Gamma\rrbracket \arrow[r, "{(\mathsf{id}, f_0\circ\theta)}"] \arrow[dd, "\theta"'] \arrow[ddr, phantom, "\overset{\text{products}}{=}"] & \llbracket\Gamma\rrbracket\times \mathrm{T}\,\llbracket\Delta_1\rrbracket \arrow[r] \arrow[dd, "\theta\times\mathsf{id}"] & \cdots \arrow[ur] & & & \cdots \arrow[r] & \llbracket\Gamma\rrbracket\times \mathrm{T}\,\llbracket\Delta\rrbracket \arrow[dr, "\gg\!=(g\circ(\theta\times\mathsf{id}))"] \arrow[dd, "\theta\times\mathsf{id}"'] & \\
 & & & \overset{\substack{\text{products and}\\ \text{strong monad}\\ \text{naturality}}}{=} & & & \overset{\substack{\text{strong monad}\\ \text{naturality}}}{=} & \mathrm{T}\,\llbracket B\rrbracket \\
\llbracket\Xi\rrbracket \arrow[r, "{(\mathsf{id}, f_0)}"'] & \llbracket\Xi\rrbracket\times \mathrm{T}\,\llbracket\Delta_1\rrbracket \arrow[r] & \cdots \arrow[dr] & & & \cdots \arrow[r] & \llbracket\Xi\rrbracket\times \mathrm{T}\,\llbracket\Delta\rrbracket \arrow[ur, "\gg\!= g"'] & \\
 & & & \llbracket\Xi\rrbracket\times \mathrm{T}\,\llbracket\Delta_i\rrbracket \arrow[r, bend right, "{\left(\mathsf{id}, \gg\!=\!\lessdot f_i\right)}"'] & \llbracket\Xi\rrbracket\times \mathrm{T}\,\llbracket\Delta_{i+1}\rrbracket \arrow[ur] & & &
\end{tikzcd}$$

The remaining proof obligation $(*)$ follows by applying $\left(\mathsf{id}, \gg\!= -\right)$ to the following equation:

$$\begin{tikzcd}
\llbracket\Gamma\rrbracket\times\llbracket\Delta_i\rrbracket \arrow[rr, bend left, "\lessdot\left(f_i\circ(\theta\times\mathsf{id})\right)"] \arrow[dr, "{\left(\boldsymbol{\pi}_2, f_i\circ(\theta\times\mathsf{id})\right)}"] \arrow[dd, "\theta\times\mathsf{id}"'] \arrow[ddr, phantom, "\overset{\text{products}}{=}"] & \overset{=:}{} & \\
 & \llbracket\Delta_i\rrbracket\times\mathrm{T}\,\llbracket A_i\rrbracket \arrow[r, "\mathsf{str}"'] & \mathrm{T}(\llbracket\Delta_i\rrbracket\times\llbracket A_i\rrbracket) \\
\llbracket\Xi\rrbracket\times\llbracket\Delta_i\rrbracket \arrow[ur, "{(\boldsymbol{\pi}_2, f_i)}"'] \arrow[urr, bend right, "\lessdot f_i"'] & \overset{=:}{} &
\end{tikzcd} \tag{$*$}$$

Functional fragment
This fragment has the typing need $\mathsf{FunTy} : \mathbf{Set} \to \mathbf{Set}$, $\mathsf{FunTy}\,X \coloneqq X \times X$ as in Ex. A.2. Then, for every fullfillment $\mathbf{L} \vDash (\to) : \mathsf{FunTy}$ of this typing need, we then define the corresponding signature functor:

$$\mathsf{Fun}\,X \coloneqq \coprod_{A,B\in\mathsf{Type}_{\mathbf{L}}} \begin{array}{l}\left((\lambda x : A.) : \underset{A\to B}{\looparrowright}[x : A] \rhd X \mathbin{@} \mathsf{comp}\,B\right)\\ \qquad \amalg \left((@) : \underset{B}{\looparrowright}(X \mathbin{@} \mathsf{comp}(A \to B)) \times (X \mathbin{@} \mathsf{comp}\,A)\right)\end{array}$$

Its derived strength is given by:

$$\begin{aligned}
\mathsf{str}^{\mathsf{Fun}}_{P,C,A\to B,\Gamma}\left[\lambda x : A.\left(p \in P_{\mathsf{comp}\,B}(\Delta, x : A)\right), \theta \in C^{\mathsf{Env}}_{\Delta}\Gamma\right]_{\Delta} &\coloneqq \lambda x : A.\,[p, (\theta, x : \mathsf{env}\,x)]_{\Delta, x:A}\\
\mathsf{str}^{\mathsf{Fun}}_{P,C,B,\Gamma}\left[\left(p \in P_{\mathsf{comp}\,A\to B}\Delta\right) @ \left(q \in P_{\mathsf{comp}\,A}\Delta\right), \theta \in C^{\mathsf{Env}}_{\Delta}\Gamma\right]_{\Delta} &\coloneqq [p, \theta]_{\Delta} @ [q, \theta]_{\Delta}
\end{aligned}$$

Let $\mathbf{M}$ be a model of the base fragment for $\mathbf{L}$, equipped with a type interpretation $\llbracket - \rrbracket : \mathsf{Type}_{\mathbf{L}} \to \mathbf{M}$. To model the functional fragment, we require, for each $A, B \in \mathsf{Type}_{\mathbf{L}}$, the structure of a Kleisli exponential, of $\mathrm{T}\,\llbracket B\rrbracket$ by $\llbracket A\rrbracket$, over $\llbracket A \to B\rrbracket$. Itn amounts to specifying morphisms $\mathsf{eval} : \llbracket A \to B\rrbracket \times \llbracket A\rrbracket \to \mathrm{T}\,\llbracket B\rrbracket$

satisfying the appropriate universal property. We then define the Fun-algebra structure by setting:

$$\left[\!\!\left[\lambda x : A. \left(\llbracket\Gamma\rrbracket \times \llbracket A\rrbracket \xrightarrow{f} \mathrm{T}\,\llbracket B\rrbracket\right)\right]\!\!\right] \coloneqq \left(\llbracket\Gamma\rrbracket \xrightarrow{\mathsf{curry}\,f} \llbracket A \to B\rrbracket\right)$$

$$\left[\!\!\left[\left(\llbracket\Gamma\rrbracket \xrightarrow{f} \mathrm{T}\,\llbracket A \to B\rrbracket\right) @ \left(\llbracket\Gamma\rrbracket \xrightarrow{a} \mathrm{T}\,\llbracket A\rrbracket\right)\right]\!\!\right] \coloneqq$$

$$\llbracket\Gamma\rrbracket \xrightarrow{(\mathsf{id}, f)} \llbracket\Gamma\rrbracket \times \mathrm{T}\,\llbracket A \to B\rrbracket \xrightarrow{\gg\!=\!\lessdot\left(\llbracket\Gamma\rrbracket \times \llbracket A \to B\rrbracket \xrightarrow{\pi_1} \llbracket\Gamma\rrbracket \xrightarrow{a} \mathrm{T}\,\llbracket A\rrbracket\right)} \mathrm{T}\left(\llbracket A \to B\rrbracket \times \llbracket A\rrbracket\right) \xrightarrow{\gg\!=\mathsf{eval}} \mathrm{T}\,\llbracket B\rrbracket$$

This algebra structure is compatible with the substitution monoid structure. For abstraction, it amounts to the following equation (cf. Ex. 7.2):

$$\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[dr, "\mathsf{curry}\left(\llbracket\Gamma\rrbracket \times \llbracket A\rrbracket \xrightarrow{\theta \times \mathsf{id}} \llbracket\Delta\rrbracket \times \llbracket A\rrbracket \xrightarrow{f} \mathrm{T}\,\llbracket B\rrbracket\right)"] \arrow[dd, "\theta"'] & \\
\arrow[r, phantom, "\substack{\text{curry-nat.}\\ ="}] & \llbracket A \to B\rrbracket \\
\llbracket\Delta\rrbracket \arrow[ur, "\mathsf{curry}\left(\llbracket\Delta\rrbracket \times \llbracket A\rrbracket \xrightarrow{f} \mathrm{T}\,\llbracket B\rrbracket\right)"'] &
\end{tikzcd}$$

For application, it amounts to the following calculation:

$$\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[r, "(\mathsf{id}, f \circ \theta)"] \arrow[dd, "\theta"'] \arrow[ddr, phantom, "\substack{\text{products}\\ ="}] & \llbracket\Gamma\rrbracket \times \llbracket A \to B\rrbracket \arrow[r, "\gg\!=\!\lessdot\,(a \circ \theta \circ \pi_1)"] \arrow[dd, "\theta \times \mathsf{id}"'] \arrow[ddr, bend left, "\gg\!=\begin{pmatrix}(\lessdot(a \circ \pi_1))\\ \circ(\theta \times \mathsf{id})\end{pmatrix}" description] & \mathrm{T}\left(\llbracket A \to B\rrbracket \times \llbracket A\rrbracket\right) \arrow[dd, equal, "\substack{(*)\\ ="}'] \arrow[dr, "\gg\!=\mathsf{eval}"] & \\
& \arrow[r, phantom, "\substack{\text{strong}\\ \text{monad nat.}\\ ="}] & \arrow[r, phantom, "="] & \mathrm{T}\left(\llbracket B\rrbracket\right) \\
\llbracket\Delta\rrbracket \arrow[r, "(\mathsf{id}, f)"'] & \llbracket\Delta\rrbracket \times \llbracket A \to B\rrbracket \arrow[r, "\gg\!=\!\lessdot\,(a \circ \pi_1)"'] & \mathrm{T}\left(\llbracket A \to B\rrbracket \times \llbracket A\rrbracket\right) \arrow[ur, "\gg\!=\mathsf{eval}"'] &
\end{tikzcd}$$

The remaining proof obligation $(*)$ is similar to $(*)$ in the sequential case.

We have demonstrated all the moving parts in using MAST to define syntax and semantics á la carte. The other fragments follow a similar treatment, defining signature functors Record, Variant, Nat, While, and Rec, each with their typing needs and model structure and properties, as in Fig A.5. We omit these technical details.

A.5 Syntax and semantics á la carte

We can put these fragments together modularly. Let ϵ range over the $2^7 = 128$ subsets of fragments:

$$\varepsilon \subseteq \{\mathsf{Seq}, \mathsf{Fun}, \mathsf{Record}, \mathsf{Variant}, \mathsf{Nat}, \mathsf{While}, \mathsf{Rec}\}$$

For each fragment ε we define:

- A simple-type signature functor for each fragment, taking the typing needs into account. For example:
  - We include all function types when we include the functional fragment, and n-ary function types in the recursive fragment even if we exclude arbitrary function types or records:

$$\mathsf{Fun}_\varepsilon\, X \coloneqq \begin{cases} \mathsf{Fun} \in \varepsilon : & \mathsf{Fun}\, X \\ (\mathsf{Fun} \notin \varepsilon \ni \mathsf{Rec}) \text{ or } (\mathsf{Record} \notin \varepsilon \ni \mathsf{Rec}) : & \coprod_{I \subseteq_{\mathrm{fin}} \mathbb{N}} (\llparenthesis x_i : - \mid i \in I \rrparenthesis \to - : X^I \times X) \\ \text{otherwise:} & \emptyset \end{cases}$$

- We include all variant types in the variants fragment, and only the relevant variants for natural numbers and iteration:

$$\mathsf{Variant}_{\varepsilon} \coloneqq \begin{cases} \mathsf{Variant} \in \varepsilon : & \mathsf{Variant} \\ \mathsf{Variant} \notin \varepsilon : & \left(\coprod_{\mathsf{Nat}\in\varepsilon}\left(\left\{\!\!\left[\begin{matrix} 0 : & (\!|\_|\!), \\ (1+) : & - \end{matrix}\right]\!\!\right\} : X\right)\right) \amalg \left(\coprod_{\mathsf{While}\in\varepsilon}\left(\left\{\!\!\left[\begin{matrix} \mathbf{done} : & -, \\ \mathbf{continue} : & - \end{matrix}\right]\!\!\right\} : X \times X\right)\right) \end{cases}$$

Then we take the overall signature functor for types to be:

$$\mathbf{L}_{\varepsilon} \coloneqq \mathsf{FunTy}_{\varepsilon} \amalg \mathsf{RecordTy}_{\varepsilon} \amalg \mathsf{VariantTy}_{\varepsilon} \amalg \mathsf{NatTy}_{\varepsilon}$$

This signature functor defines the set of types in this fragment $\mathsf{Type}_{\varepsilon}$, and the corresponding binding structure $\mathrm{CBV}_{\varepsilon} \coloneqq \mathrm{CBV}_{\mathsf{Type}_{\varepsilon}}$.

- This definition supports a fulfillment $\mathbf{L}_{\varepsilon} \vDash \alpha_{\varepsilon} : \mathbf{R}$ for each typing need of each fragment. For example, when $\mathsf{Rec} \in \varepsilon$, the functor $\mathbf{L}_{\varepsilon}$ supports n-ary function types whether or not $\mathsf{FunTy} \in \varepsilon$.
- Similarly, we define a binding signature functor for each subset of construct depending on the features we include in ε, collecting them via coproducts into a binding signature functor $\mathbf{O}_{\varepsilon}$.
- For each ε, the right-most column in Fig A.5 specifies the semantic structure and property for the model, and we impose the conjunction of structures/properties for each row in ε, chosen coherently when two typing requirements overlap. Each such structure derives a corresponding algebra structure on the model for each fragment, and correspondingly an $\mathbf{O}$-monoid structure.
- We then derive through the Representation Theorem: an abstract syntax, substitution operation, a denotational semantics for each model, and a substitution lemma for this model.

B Technical development outline

Our development is relatively straightforward thanks to several abstractions: skew monoidal categories (Appendix C), skew bi-categories and the right-closed skew-monoidal structure of substitution (Appendix D) and the full proof of the General Representation Thm B.2 (Appendix E). In this section, we summarise how these abstract components intertwine, and relegate the remaining details to the remaining sections of the appendix.

B.1 Bicategorical development

Fiore, Gambino, Hyland, and Winskel [28] package the sophistication involved in the classical substitution tensor product in a bicategory we denote by $\mathbf{Prof}_{\vdash}$. Its vertices/0-cells are small categories $\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots$. The category $S_{\vdash}$ of S-sorted contexts is a the the finite-product completion of S, and extends to categories. The arrows/1-cells $P : \mathbb{A} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbb{B}$ in $\mathbf{Prof}_{\vdash}$ are Kleisli profunctors $P : \mathbb{A} \nrightarrow \mathbb{B}_{\vdash}$, i.e. presheaves $P \in \mathbf{PSh}\,(\mathbb{A}^{\mathsf{op}} \times \mathbb{B}_{\vdash})$, equivalently functors $P : \mathbb{A}{\times}\mathbb{B}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$. Its faces/2-cells $\alpha : P \Rightarrow Q$ are natural transformations, with their usual vertical and horizontal composition. Arrows $P : \mathbb{A} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbb{B}$ and $Q : \mathbb{B} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbb{C}$ compose diagrammatically using the same formula as the substitution tensor product, and the profunctor of variables, suitably generalised, is the identity $\mathbb{V} : \mathbb{A} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbb{A}$. One recovers the classical setting by restricting to the full sub-bicategory over a set of sorts $\mathbf{R}$, since a one vertex bicategory is a monoidal category.

Our skew theory utilises the following bicategory $\mathbf{SortSys}$. Its vertices are heterogeneous sorting systems $\mathbf{R}$. A 1-cell $P : \mathbf{R} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbf{S}$ is then a Kleisli profunctor $P : \mathbf{R} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbf{S}\text{-}\mathsf{Bind}$, and 2-cells are again natural transformations with horizontal and vertical composition. We compose 1-cells $P : \mathbf{R} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbf{S}$ and $Q : \mathbf{S} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbf{T}$ diagrammatically using the same formula as our skewed tensors, equivalently by $P \otimes Q \coloneqq P \circledS (Q|_{\mathsf{bnd}})$, with the heterogeneous structures of variables as identities $\mathbb{I} : \mathbf{R} \twoheadrightarrow\!\!\!\!\!\!\!\!\;\;| \;\; \mathbf{R}$. The remaining bicategorical structure for $\mathbf{SortSys}$ lifts by noticing that $(P \otimes Q) \cong (P|_{\mathsf{bnd}}) \circledS (\otimes Q)$. See Appendix D for the full details. Restricting to the full sub-bicategory of a given sorting system $\mathbf{R}$ yields the skew monoidal bicategory of $\mathbf{R}$-structures.

B.2 The representation theorem

$$\begin{tikzcd} & (x \mathbin{\leftarrow\!\!\otimes} y)\otimes y \arrow[d, "\mathsf{eval}"] \\ z\otimes y \arrow[ru, "\mathsf{curry}\, f\otimes\mathsf{id}"] \arrow[r, "f"'] \arrow[ru, phantom, shift right=3, "="] & x \end{tikzcd}$$

Adapting the classical proof for the representation theorem is straightforward thanks to its level of generality. Let $\mathcal{C}$ be a skew monoidal category, and $x, y \in \mathcal{C}$. Recall that a *right exponential of x by y* is an object $x \mathbin{\leftarrow\!\!\otimes} y$ equipped with a universal morphism $\mathsf{eval} : ((x \mathbin{\leftarrow\!\!\otimes} y)\otimes y) \to x$, i.e., for every arrow $f : z\otimes y \to x$ there is a unique arrow $\mathsf{curry}\, f : z \to (x \mathbin{\leftarrow\!\!\otimes} y)$ making the diagram on the right commute. A skew monoidal category $\mathcal{C}$ is *right-closed* when all right-exponentials exist. In that case, $(\otimes)$ distributes over coproducts. We prove the following result in Appendix D.

Proposition B.1 *The category of* $\mathbf{R}$*-structures is right-closed, with:* $(P \mathbin{\leftarrow\!\!\otimes} Q)_s\Gamma := \int_{\Delta\in\mathbf{R}\text{-}\mathsf{Bind}_{\vdash}} (P_s\Delta)^{Q^{\mathsf{Env}\,\Gamma}_{\Delta}}$.

The following result (cf. Appendix E) provides the following generalisation of the classical theory:

Theorem B.2 (general representation) *Let $\mathcal{C}$ be an assocciative, right-closed skew monoidal category, $\mathbf{O} : \mathcal{C}_\bullet \to \mathcal{C}_\bullet$ be a pointed-strong functor, $x \in \mathcal{C}$ an object, and $\left(\$^{\mathbf{O}}x, [\![-]\!], \mathsf{var}, \mathsf{meta}\right)$ be an initial algebra over $\$^{\mathbf{O}} := \mu z.(\underline{\mathbf{O}}z)\amalg\mathbb{I}\amalg(x\otimes z)$. There is a unique morphism $-[-] : \$^{\mathbf{O}}x\otimes\$^{\mathbf{O}}x \to \$^{\mathbf{O}}x$, called* simultaneous substitution*, satisfying analogous equations to Thm 7.4. Equipping $\mathbf{F_O}x := \left(\$^{\mathbf{O}}x, -[-], \mathsf{var}, [\![-]\!]\right)$ with the arrow $\mathsf{menv} : x \xrightarrow{\mathbf{r}'} x\otimes\mathbb{I} \xrightarrow{\mathsf{id}\otimes\mathsf{var}} x\otimes\$^{\mathbf{O}}x \xrightarrow{\mathsf{meta}} \$^{\mathbf{O}}x$ yields the free $\mathbf{O}$-monoid over x.*

C Skew monoidal categories

We relegate the formal definitions of skew monoidal category and structure to this appendix as their precise definition is standard, and not needed to work with MAST for the tutorial and technical development. We include them, and omitted proofs, for ease of reference of the readers.

C.1 Monoidal structure

Recall that a *right-skew monoidal category* [72] $\left(\mathcal{C}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$ consists of a category $\mathcal{C}$ equipped with a functor $(\otimes) : \mathcal{B}\times\mathcal{C} \to \mathcal{C}$, an object $\mathbb{I} \in \mathcal{C}$, and three transformations called the *associator*, natural in $a, b, c \in \mathcal{C}$ and the *left/right unitors*, natural in $a \in \mathcal{C}$:

$$\mathbf{a} : (a\otimes b)\otimes c \to a\otimes(b\otimes c) \qquad \boldsymbol{\ell} : \mathbb{I}\otimes a \to a \qquad \mathbf{r}' : a \to a\otimes\mathbb{I}$$

satisfying the following equations:

$$\begin{tikzcd} & (a\otimes b)\otimes(c\otimes d) \arrow[rd, "\mathbf{a}"] & \\ ((a\otimes b)\otimes c)\otimes d \arrow[ru, "\mathbf{a}"] \arrow[d, "\mathbf{a}\otimes\mathsf{id}"'] & \text{monoidal pentagon} = & a\otimes(b\otimes(c\otimes d)) \\ (a\otimes(b\otimes c))\otimes d \arrow[rr, "\mathbf{a}"'] & & a\otimes((b\otimes c)\otimes d) \arrow[u, "\mathsf{id}\otimes\mathbf{a}"'] \end{tikzcd}$$

$$\begin{tikzcd} (\mathbb{I}\otimes a)\otimes b \arrow[rr, "\mathbf{a}"] \arrow[rd, "\boldsymbol{\ell}\otimes\mathsf{id}"'] & \text{skew left} = & \mathbb{I}\otimes(a\otimes b) \arrow[ld, "\boldsymbol{\ell}"] \\ & a\otimes b & \end{tikzcd}$$

$$\begin{tikzcd} & a\otimes b \arrow[ld, "\mathbf{r}'"'] \arrow[rd, "\mathsf{id}\otimes\mathbf{r}'"] & \\ (a\otimes b)\otimes\mathbb{I} \arrow[rr, "\mathbf{a}"'] & \text{skew right} = & a\otimes(b\otimes\mathbb{I}) \end{tikzcd}$$

$$\begin{tikzcd} (a\otimes\mathbb{I})\otimes b \arrow[r, "\mathbf{a}"] & a\otimes(\mathbb{I}\otimes b) \arrow[d, "\mathsf{id}\otimes\boldsymbol{\ell}"] \\ a\otimes b \arrow[u, "\mathbf{r}'\otimes\mathsf{id}"] \arrow[r, equal] \arrow[ru, phantom, "\text{monoidal rectangle} ="] & a\otimes b \end{tikzcd}$$

$$\begin{tikzcd} \mathbb{I} \arrow[rr, equal] \arrow[rd, "\mathbf{r}'"'] & \text{skew triangle} = & \mathbb{I} \\ & \mathbb{I}\otimes\mathbb{I} \arrow[ru, "\boldsymbol{\ell}"'] & \end{tikzcd}$$

The two left axioms generalise the monoidal pentagon and triangle axioms.

[7] Somewhat confusingly, the structure we call right-skew monoidal category here is called a *left-monoidal* category by Szlachányi [72]. We adopt what appears to be the persisting later terminology [37, e.g.].

$$\begin{tikzcd}
 & & \underline{\mathbb{I}^\bullet\otimes^\bullet A} \arrow[d, equal] \\
 & \mathbb{I}\otimes\mathbb{I} \arrow[r, "\mathsf{id}\otimes\mathsf{env}"] \arrow[d, "\boldsymbol{\ell}"] \arrow[r, phantom, shift left=6, ":="] & \mathbb{I}\otimes\underline{A} \arrow[dd, "\boldsymbol{\ell}"] \arrow[ddl, phantom, "\boldsymbol{\ell}\text{-nat} ="] \\
\mathbb{I} \arrow[uurr, bend left, "\mathsf{env}"] \arrow[ur, "\mathbf{r}'"] \arrow[r, equal] \arrow[ur, phantom, shift right=4, "\text{skew triangle} ="] \arrow[drr, bend right, "\mathsf{env}"'] & \mathbb{I} \arrow[dr, "\mathsf{env}"] \arrow[d, phantom, "\text{id} ="] & \\
 & & \underline{A}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & \underline{A} \arrow[dd, "\mathbf{r}'"] \\
\mathbb{I} \arrow[urr, "\mathsf{env}"] \arrow[dr, "\mathbf{r}'"] \arrow[ddrr, bend right, "\mathsf{env}"'] & \mathbf{r}'\text{ nat} = & \\
 & \mathbb{I}\otimes\mathbb{I} \arrow[r, "\mathsf{env}\otimes\mathsf{id}"'] \arrow[r, phantom, shift right=6, ":="] & \underline{A}\otimes\mathbb{I} \arrow[d, equal] \\
 & & \underline{B^\bullet\otimes^\bullet\mathbb{I}^\bullet}
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & \underline{(A\otimes^\bullet B)\otimes^\bullet C} \arrow[d, equal] \\
 & \mathbb{I}\otimes\mathbb{I} \arrow[r, "\mathbf{r}'\otimes\mathsf{id}"'] \arrow[rr, bend left, "\mathsf{env}\otimes\mathsf{env}"] \arrow[dd, equal] \arrow[dd, phantom, shift right=6, "\text{id} ="] & (\mathbb{I}\otimes\mathbb{I})\otimes\mathbb{I} \arrow[r, "(\mathsf{env}\otimes\mathsf{env})\otimes\mathsf{env}"] \arrow[dd, "\mathbf{a}"] \arrow[dd, phantom, shift right=10, "\text{skew right} ="] & (\underline{A}\otimes\underline{B})\otimes\underline{C} \arrow[dd, "\mathbf{a}"] \arrow[ddl, phantom, "\mathbf{a}\text{ nat} ="] \\
\mathbb{I} \arrow[uurrr, bend left, "\mathsf{env}"] \arrow[ur, "\mathbf{r}'"] \arrow[dr, "\mathbf{r}'"'] \arrow[ddrrr, bend right, "\mathsf{env}"'] & & & \\
 & \mathbb{I}\otimes\mathbb{I} \arrow[r, "\mathsf{id}\otimes\mathbf{r}'"] \arrow[rr, bend right, "\mathsf{env}\otimes\mathsf{env}"'] & \mathbb{I}\otimes(\mathbb{I}\otimes\mathbb{I}) \arrow[r, "\mathsf{env}\otimes(\mathsf{env}\otimes\mathsf{env})"] & \underline{A}\otimes(\underline{B}\otimes\underline{C}) \arrow[d, equal] \\
 & & & \underline{A\otimes^\bullet(B\otimes^\bullet C)}
\end{tikzcd}$$

Fig. C.1. Lifting the mediators from a skew monoidal category to its pointed objects

The functorial action of the pointed tensor $(\otimes^\bullet)$ from §6, and the initial map out of $\mathbb{I}^\bullet$ are well-defined due to the following arguments:

$$\begin{tikzcd}
 & & \underline{A}\otimes\underline{A'} \arrow[dd, bend left, "f\otimes^\bullet g"] \\
\mathbb{I} \arrow[urr, bend left, "\mathsf{env}^{A\otimes^\bullet A'}"] \arrow[r, "\mathbf{r}'"] \arrow[drr, bend right, "\mathsf{env}^{B\otimes^\bullet B'}"'] & \mathbb{I}\otimes\mathbb{I} \arrow[ur, "\mathsf{env}^A\otimes\mathsf{env}^{A'}"] \arrow[dr, "\mathsf{env}^B\otimes\mathsf{env}^{B'}"'] \arrow[u, phantom, ":="] \arrow[d, phantom, ":="] \arrow[r, phantom, "\text{pt pres}\otimes\text{pt pres} ="] & \\
 & & \underline{B}\otimes\underline{B'}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \underline{\mathbb{I}^\bullet} \arrow[d, "{[]} := \mathsf{env}^A"] \\
\mathbb{I} \arrow[ur, bend left, "\mathsf{env}^{\mathbb{I}^\bullet}"] \arrow[ur, equal] \arrow[r, "\mathsf{env}^A"'] \arrow[ur, phantom, shift left=6, ":="] \arrow[r, phantom, shift left=4, "\text{id} ="] & \underline{A}
\end{tikzcd}$$

Since the primary source for the following proposition is unpublished, we include its proof. However, we make no claim to its originality. Its suffix concerning associativity and unitality is a simple consequences of the unpublished result.

Proposition C.1 (Fiore and Szamozvancev [25]) *Every right-skew monoidal category $\mathcal{C}$ yields a right-skew monoidal category of pointed objects $\mathcal{C}^\bullet := \left(\underline{\mathcal{C}}^\bullet, (\otimes^\bullet), \mathbb{I}^\bullet, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$. I.e., the skew mediators preserve points, hence lift to $\mathcal{C}^\bullet$. The category $\mathcal{C}^\bullet$ associative, or left/right unital iff $\mathcal{C}$ is.*

Proof. Straightforward calculation as in Fig C.1. By appealing to the faithfulness of $\underline{-} : \underline{\mathcal{C}}^\bullet \to \underline{\mathcal{C}}$ we deduce the skew monoidal axioms for $\mathcal{C}^\bullet$ from those of $\mathcal{C}$. An invertible pointed arrow is a pointed isomorphism, i.e., this functor reflects isomorphisms, and so $\mathcal{C}^\bullet$ has any associativity or unitality property iff $\mathcal{C}$ has it. □

C.2 Actions

In §6 we defined associative and unital right-actions for associative and unital monoidal categories. The general definitions postulates one more axiom that associativity and unitality help discharge.

Let $\mathcal{C} = \left(\mathcal{B}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$ be a right-skew monoidal category. A *right action* $\mathcal{A} = \left(\mathcal{A}, (\otimes\!\!\!>), \mathbf{a}, \mathbf{r}'\right)$ consists of a category $\mathcal{A}$ equipped with a functor $(\otimes\!\!\!>) : \underline{\mathcal{A}} \times \underline{\mathcal{C}} \to \underline{\mathcal{A}}$, an *associator* transformation $\mathbf{a} : (x \otimes\!\!\!> a) \otimes\!\!\!> b \to x \otimes\!\!\!> (a \otimes b)$ natural in x, a, b and a *right unitor* transformation $\mathbf{r}' : x \to x \otimes\!\!\!> \mathbb{I}$ natural in x, satisfying the conditions:

$$\begin{tikzcd}
& (x \oslash a) \oslash (b \otimes c) \arrow[rr, "\mathbf{a}"] & & x \oslash (a \otimes (b \otimes c)) \\
((x \oslash a) \oslash b) \oslash c \arrow[ur, "\mathbf{a}"] \arrow[dr, "\mathbf{a} \otimes \mathsf{id}"'] & & \overset{\text{action pentagon}}{=} & \\
& (x \oslash (a \otimes b)) \oslash c \arrow[r, "\mathbf{a}"'] & x \oslash ((a \otimes b) \otimes c) \arrow[uur, "\mathsf{id} \otimes \mathbf{a}"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
(x \oslash \mathbb{I}) \oslash a \arrow[r, "\mathbf{a}"] & x \oslash (\mathbb{I} \otimes a) \arrow[d, "\mathsf{id} \otimes \boldsymbol{\ell}"] \\
x \oslash a \arrow[u, "\mathbf{r}' \otimes \mathsf{id}"] \arrow[r, equal] \arrow[ur, phantom, "\overset{\text{action rectangle}}{=}"] & x \oslash a
\end{tikzcd}
\qquad
\begin{tikzcd}
& x \oslash a \arrow[dl, "\mathbf{r}'"'] \arrow[dr, "\mathsf{id} \otimes \mathbf{r}'"] & \\
(x \oslash a) \oslash \mathbb{I} \arrow[rr, "\mathbf{a}"'] \arrow[rr, phantom, "\overset{\text{action skew right}}{=}", yshift=12pt] & & x \oslash (a \otimes \mathbb{I})
\end{tikzcd}$$

The last axiom follows from the others in associative right-unital structures:

Lemma C.2 (Kelly's argument [48]) *Let $\mathcal{C} = \left(\mathcal{B}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}'\right)$ be a right-skew associative right-unital monoidal category, $\left(\mathcal{A}, (\oslash), \mathbf{a}, \mathbf{r}'\right)$ be the data for a right $\mathcal{C}$-action. If the action's associator* **a** *and unitor* $\mathbf{r}'$ *are invertible and the pentagon and rectangle axioms hold, then so does the skew right axiom.*

Proof. We adapt Kelly's proof for his Thm 7. We maintain the convention that $\mathbf{r}' = \mathbf{r}^{-1}$. Instantiate the pentagon axiom as follows, leaving the starred face uncommuted:

$$\begin{tikzcd}
& & & (x \oslash a) \oslash (\mathbb{I} \otimes \mathbb{I}) \arrow[dl, "(\mathsf{id} \oslash \mathsf{id}) \oslash \boldsymbol{\ell}"'] \arrow[rrdd, bend left, "\mathbf{a}"] & & \\
& \overset{\text{assumption}}{=} & (x \oslash a) \oslash \mathbb{I} \arrow[dl, equal] \arrow[dr, "\mathbf{a}"] & & \overset{\mathbf{a}\text{-nat}}{=} & \\
((x \oslash a) \oslash \mathbb{I}) \oslash \mathbb{I} \arrow[rrruu, bend left, "\mathbf{a}"] \arrow[dr, "\mathbf{a} \oslash \mathsf{id}"'] & (x \oslash a) \oslash \mathbb{I} \arrow[l, "\mathbf{r}' \oslash \mathsf{id}"'] \arrow[d, "(\mathsf{id} \oslash \mathbf{r}') \oslash \mathsf{id}"] & & x \oslash (a \otimes \mathbb{I}) \arrow[r, equal] \arrow[dr, "\mathsf{id} \oslash (\mathbf{r}' \otimes \mathsf{id})"] & x \oslash (a \otimes \mathbb{I}) & x \oslash (a \otimes (\mathbb{I} \otimes \mathbb{I})) \arrow[l, "\mathsf{id} \oslash (\mathsf{id} \otimes \boldsymbol{\ell})"'] \\
(*) & (x \oslash (a \otimes \mathbb{I})) \oslash \mathbb{I} \arrow[rrr, "\mathbf{a}"'] & \overset{\mathbf{a}\text{-nat}}{=} & & x \oslash ((a \otimes \mathbb{I}) \otimes \mathbb{I}) \arrow[ur, "\mathsf{id} \oslash \mathbf{a}"'] & \overset{\text{skew rectangle}}{=}
\end{tikzcd}$$

By inverting the two bottom right associators and the arrow $\mathsf{id} \oslash (\mathbf{r}' \otimes \mathsf{id})$, and using the commutativity of the outer face, we deduce that $(*)$ commutes. The following calculation proves action skew right axiom:

$$\begin{tikzcd}
(x \oslash a) \oslash \mathbb{I} \arrow[rrrr, "\mathbf{a}"] & & & & x \oslash (a \otimes \mathbb{I}) \\
& ((x \oslash a) \oslash \mathbb{I}) \oslash \mathbb{I} \arrow[rr, "\mathbf{a} \oslash \mathsf{id}"] & \overset{(*)}{=} & (x \oslash (a \otimes \mathbb{I})) \oslash \mathbb{I} \arrow[ur, "\mathbf{r}"] & \\
& \overset{\mathbf{r}\text{-nat}}{=} & (x \oslash a) \oslash \mathbb{I} \arrow[ul, "\mathbf{r}' \oslash \mathsf{id}"'] \arrow[ur, "(\mathsf{id} \oslash \mathbf{r}') \oslash \mathsf{id}"] & \overset{\mathbf{r}\text{-nat}}{=} & \\
& & x \oslash a \arrow[u, "\mathbf{r}'"] \arrow[uuull, bend left, "\mathbf{r}'"] \arrow[uuurr, bend right, "\mathsf{id} \oslash \mathbf{r}'"'] & &
\end{tikzcd}$$

□

In §6 we claimed Lemma 6.8 has a straightforward proof:

Lemma C.3 *Let $\mathcal{C}$ be a skew monoidal category; $\mathcal{A}$,$\mathcal{B}$ two $\mathcal{C}$-actions; and $F : \mathcal{A} \to \mathcal{B}$ a strong functor. Then the following exhibits $\underline{F}$ as a strong functor: $F_\bullet : \mathcal{A}_\bullet \to \mathcal{B}_\bullet$.*

$$\mathsf{str}^{F_\bullet}_{x,a} : (\underline{F}x) \oslash_\bullet a = (\underline{F}x) \oslash \underline{a} \xrightarrow{\mathsf{str}^{F}_{x,\underline{a}}} \underline{F}(x \oslash \underline{a}) = \underline{F}(x \oslash_\bullet a)$$

Proof. Consider a strong functor $F : \mathcal{A} \to \mathcal{B}$ and define a putative $\mathsf{str}^{F_\bullet}$ as in the lemma statement.

For naturality, take any $f : x \to x$ in $\mathcal{C}$ and $g : a \to b$ in $\mathcal{C}^{\bullet}$:

$$\begin{tikzcd}
(\underline{F}x) \rtimes a \arrow[rrr, "\mathsf{str}^{F_\bullet}"] \arrow[dddd, "\underline{F}f \rtimes_{\bullet} g"'] \arrow[dr, equal] & & & \underline{F}(x \rtimes a) \arrow[dddd, "\underline{F}(f \otimes^{\bullet} g)"] \\
& & \text{def.} = & \\
& (\underline{F}x) \otimes \underline{a} \arrow[r, "\mathsf{str}"'] \arrow[d, "\underline{F}f \rtimes g"] \arrow[l, phantom, "\text{def.} ="] & \underline{F}(x \otimes \underline{a}) \arrow[d, "\underline{F}(f \otimes g)"'] \arrow[uur, equal] \arrow[r, phantom, "\text{def.} ="] \arrow[dl, phantom, "\text{assumption} ="] & \\
& (\underline{F}y) \otimes \underline{b} \arrow[r, "\mathsf{str}"] & \underline{F}(x \otimes \underline{b}) \arrow[dr, equal] & \\
(\underline{F}y) \rtimes b \arrow[rrr, "\mathsf{str}^{F_\bullet}"'] \arrow[ur, equal] & & \text{def.} = & \underline{F}(y \rtimes b)
\end{tikzcd}$$

For the strength pentagon and rectangle, calculate:

$$\begin{tikzcd}
& & \underline{F}(x \rtimes_{\bullet} (a \otimes^{\bullet} b)) \arrow[rrd, "\underline{F}\mathbf{a}"] \arrow[d, equal] & & \\
\underline{F}x \rtimes_{\bullet} (a \otimes^{\bullet} b) \arrow[urr, "\mathsf{str}^{F_\bullet}"] \arrow[dr, equal] \arrow[dddr, "\mathbf{a}"'] & \text{def} = & \underline{F}(x \otimes (\underline{a} \rtimes \underline{b})) \arrow[dr, "\mathbf{Oa}"'] & \text{def} = & \underline{F}((x \rtimes_{\bullet} a) \rtimes_{\bullet} b) \\
& \underline{F}x \otimes (\underline{a} \rtimes \underline{b}) \arrow[ur, "\mathsf{str}"'] \arrow[d, "\mathbf{a}"] & \text{assumption} = & \underline{F}((x \otimes \underline{a}) \rtimes \underline{b}) \arrow[ur, equal] & \\
\text{def} = & (\underline{F}x \otimes \underline{a}) \rtimes \underline{b} \arrow[rr, bend right, "\mathsf{str} \rtimes \mathsf{id}"'] \arrow[d, equal] & & (\underline{F}(x \otimes \underline{a})) \rtimes \underline{b} \arrow[u, "\mathsf{str}"] \arrow[d, equal] & \text{def} = \\
& (\underline{F}x \rtimes_{\bullet} a) \rtimes_{\bullet} b \arrow[rr, bend right, "\mathsf{str}^{F_\bullet} \rtimes_{\bullet} \mathsf{id}"'] & \text{def} = & (\underline{F}(x \rtimes_{\bullet} a)) \rtimes_{\bullet} b \arrow[uuur, "\mathsf{str}^{F_\bullet}"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
& & \underline{F} \arrow[ddll, bend right, "\mathbf{r}"'] \arrow[dl, "\mathbf{r}'"'] \arrow[dr, "\underline{F}\mathbf{r}'"] \arrow[ddrr, bend left, "\underline{F}\mathbf{r}'"] & & \\
& (\underline{F}x) \rtimes \mathbb{I} \arrow[rr, bend left, "\mathsf{str}"'] \arrow[dl, equal] \arrow[l, phantom, "\text{def.} ="] & \text{assumption} = & \underline{F}(x \rtimes \mathbb{I}) \arrow[dr, equal] \arrow[r, phantom, "\text{def.} ="] & \\
(\underline{F}x) \rtimes_{\bullet} \mathsf{id}_{\mathbb{I}} \arrow[rrrr, bend right, "\mathsf{str}"'] & & \text{def.} = & & \underline{F}(x \rtimes_{\bullet} \mathsf{id}_{\mathbb{I}})
\end{tikzcd}$$

as we wanted. □

The fact that strong functors compose has been known in some form of another since the early years of category theory [22, e.g.]. The argument remains unchanged in the skew setting:

Lemma C.4 (well-known) *Let $\mathcal{A}_1 \xrightarrow{F} \mathcal{A}_2 \xrightarrow{G} \mathcal{A}_3$ be $\mathcal{C}$-strong functors. Then composing the strengths yields a strong functor $G \circ F : \mathcal{A}_1 \to \mathcal{A}_3$:*

$$\mathsf{str}^{G \circ F}_{x,a} : (GFx) \rtimes a \xrightarrow{\mathsf{str}^{G}} G(Fx \rtimes a) \xrightarrow{G\,\mathsf{str}^{F}} GF(x \rtimes a)$$

Proof. Straightforward calculation:

$$\begin{tikzcd}
 & \underline{G}\underline{F}x \arrow[ddl, bend right, "\mathbf{r}'"'] \arrow[ddr, bend left, "\underline{G}\underline{F}\mathbf{r}'"] \arrow[d, "\underline{G}\mathbf{r}'"'] & \\
\overset{\mathsf{str}^G\text{ triangle}}{=} & \underline{G}((\underline{F}x)\rtimes\mathbb{I}) \arrow[dr, "\underline{G}\,\mathsf{str}^F"] \arrow[d, phantom, ":="] & \overset{\substack{G\text{ functoriality}\\ \mathsf{str}^F\text{ triangle}}}{=} \\
(\underline{G}\underline{F}x)\rtimes\mathbb{I} \arrow[ur, "\mathsf{str}^G"] \arrow[rr, "\mathsf{str}^{F\circ G}"'] & & \underline{G}\underline{F}(x\rtimes\mathbb{I})
\end{tikzcd}$$

$$\begin{tikzcd}
 & & (\underline{G}\underline{F}x)\rtimes(a\otimes b) \arrow[dr, "\mathsf{str}^G"'] \arrow[ddrr, bend left, "\mathsf{str}^{G\circ F}"] & & \\
 & \overset{\mathsf{str}^G\text{ pentagon}}{=} & & \underline{G}((\underline{F}x)\rtimes(a\otimes b)) \arrow[dr, "\underline{G}\,\mathsf{str}^F"'] & =: \\
((\underline{G}\underline{F}x)\rtimes a)\rtimes b \arrow[uurr, bend left, "\mathbf{a}"] \arrow[dr, "\mathsf{str}^G\otimes\mathsf{id}"] \arrow[ddr, bend right, "\mathsf{str}^{G\circ F}\otimes\mathsf{id}"'] & & \underline{G}(((\underline{F}x)\rtimes a)\rtimes b) \arrow[ur, "\underline{G}\mathbf{a}"] \arrow[dd, "\underline{G}(\mathsf{str}^F\otimes\mathsf{id})"] & & \underline{G}\underline{F}(x\rtimes(a\otimes b)) \\
:= & (\underline{G}((\underline{F}x)\rtimes a))\rtimes b \arrow[ur, bend left, "\mathsf{str}^G"] \arrow[d, "\underline{G}\,\mathsf{str}^F\otimes\mathsf{id}"] & \overset{\mathsf{str}^G\text{-nat.}}{=} & \overset{\substack{G\text{ functoriality}\\ \mathsf{str}^F\text{ pentagon}}}{=} & \\
 & (\underline{G}\underline{F}(x\rtimes a))\rtimes b \arrow[r, "\mathsf{str}^G"'] \arrow[rr, bend right, "\mathsf{str}^{G\circ F}"'] & \underline{G}((\underline{F}(x\rtimes a))\rtimes b) \arrow[r, "\underline{G}\,\mathsf{str}^F"'] & \underline{G}\underline{F}((x\rtimes a)\rtimes b) \arrow[uur, "\underline{G}\underline{F}\mathbf{a}"'] & \\
 & & := & &
\end{tikzcd}$$

as we wanted. □

C.3 Compatible monoids

We discharge Lemma 7.3:

Lemma C.5 *Let $\left(\mathbf{O}_i\right)_{i\in I}$ be a family of* **R***-signature functors; let* **M** *be a substitution monoid; and let $\left(\llbracket-\rrbracket_i : \underline{\mathbf{O}_i\mathbf{M}} \to \underline{\mathbf{M}}\right)_{i\in I}$ be a family of algebras. The cotupled algebra: $\left[\llbracket-\rrbracket_i\right]_{i\in I} : \coprod_{i\in I}\underline{\mathbf{O}_i\mathbf{M}} \to \underline{\mathbf{M}}$ is compatible with* **M** *iff every algebra $\mathbf{O}_i$ is compatible with* **M**.

Proof. For each $j \in I$, consider the following diagram:

$$\begin{tikzcd}
 & & \coprod_i \underline{\mathbf{O}}_i(\underline{\mathbf{M}}\otimes\underline{\mathbf{M}}) \arrow[ddrrr, bend left, "\coprod_i \underline{\mathbf{O}}_i(-[-]_{\mathbf{M}})"] & & & \\
 & := & \coprod_i\left((\underline{\mathbf{O}_i\mathbf{M}})\otimes\underline{\mathbf{M}}\right) \arrow[u, "\coprod_i \mathsf{str}^{\mathbf{O}_i}"'] & & \overset{(j\ :)\text{-nat}}{=} & \\
\left(\coprod_i \underline{\mathbf{O}_i\mathbf{M}}\right)\otimes_{\bullet}\mathsf{env}^{\mathbf{M}} \arrow[uurr, bend left, "\mathsf{str}^{\coprod_i \mathbf{O}_i}"] \arrow[urr, "{\left[(i\ :)\otimes\mathsf{id}\right]_i^{-1}}"] \arrow[ddr, bend right, "{\left[\llbracket-\rrbracket_i\right]_i\otimes\mathsf{id}}"'] & \overset{\text{coproducts}}{=} & & \underline{\mathbf{O}}_j(\underline{\mathbf{M}}\otimes\underline{\mathbf{M}}) \arrow[dr, "\underline{\mathbf{O}}_j(-[-]_{\mathbf{M}})"] & & \coprod_i \underline{\mathbf{O}}_i\underline{\mathbf{M}} \arrow[ddl, bend left, "{\left[\llbracket-\rrbracket_i\right]_i}"] \\
\overset{\text{coproducts}}{=} & \left(\underline{\mathbf{O}_j\mathbf{M}}\right)\otimes_{\bullet}\mathsf{env}^{\mathbf{M}} \arrow[ul, "(j\ :)\otimes\mathsf{id}"] \arrow[uur, "(j\ :)"] \arrow[urr, "\mathsf{str}^{\mathbf{O}_j}"'] \arrow[d, "\llbracket-\rrbracket_j\otimes\mathsf{id}"] & & & \underline{\mathbf{O}}_j\underline{\mathbf{M}} \arrow[ur, "(j\ :)"'] \arrow[d, "\llbracket-\rrbracket_j"'] & \overset{\text{coproducts}}{=} \\
 & \underline{\mathbf{M}}\otimes\underline{\mathbf{M}} \arrow[rrr, "-[-]_{\mathbf{M}}"'] & & & \underline{\mathbf{M}} &
\end{tikzcd}$$

The outer face is the compatibility condition for the cotupled algebra, and the inner face is the compatibility condition for the j-th algebra. Since **R-Struct** is distributive, the outer face commutes iff all inner faces commute, as we wanted. □

D Skew bicategorical development

For completeness, we recall the full definition of a skew bicategory. Readers already familiar with this definition may skip to page 38.

D.1 Definition

A *right-skew bicategory* [52] $\mathcal{B}$ consists of:

- *Vertices/*0*-cells* $a, b, c, \ldots \in \mathcal{B}$
- For every pair of vertices a, b, a category $\mathcal{B}(a, b)$ whose:
  - objects are the *arrows/*1*-cells* $f, g, h, \ldots : a \to b$; and whose
  - morphisms are the *faces/*2*-cells* $\alpha, \beta, \gamma, \ldots : f \Rightarrow g$;
  - identities are the *strictly commuting faces/identity* 2*-cells* $\mathsf{id} : f = g$;
  - composition operations sending two *vertically composable* 2-cells to their *vertical composition*:

$$\begin{tikzcd} a \arrow[r, bend left=50, "f"{name=F}] \arrow[r, "g"{name=G, description}] \arrow[r, bend right=50, "h"'{name=H}] & b \arrow[r, phantom, "\mapsto"] & a \arrow[r, bend left=50, "f"{name=F2}] \arrow[r, bend right=50, "h"'{name=H2}] & b \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=G, to=H, "\beta"] \arrow[Rightarrow, from=F2, to=H2, "\beta\circ\alpha"] \end{tikzcd}$$

  The category axioms amount to requiring vertical composition to be associative and the identity 2-cells to compose neutrally on both sides.
- For every $a, b, c \in \mathcal{B}$, a functor $\mathcal{B}(b, c) \times \mathcal{B}(a, b) \to \mathcal{B}(a, c)$ whose action on:
  - objects sends every pair of *composable* arrows/1-cells $a \xrightarrow{f} b \xrightarrow{g} c$ to their *arrow/*1*-cell composition* which we write in *application order* as $g \circ f : a \to c$ and in *diagram order* as $f \otimes g : a \to c$.
  - morphisms sends every pair of *horizontally composable* 2-cells to their *horizontal composition*:

$$\begin{tikzcd} a \arrow[r, bend left=40, "f"{name=F}] \arrow[r, bend right=40, "g"'{name=G}] & b \arrow[r, bend left=40, "h"{name=H}] \arrow[r, bend right=40, "k"'{name=K}] & c \arrow[r, phantom, "\mapsto"] & a \arrow[r, bend left=40, "f\otimes h"{name=FH}] \arrow[r, bend right=40, "g\otimes k"'{name=GK}] & c \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=H, to=K, "\beta"] \arrow[Rightarrow, from=FH, to=GK, "\alpha\otimes\beta"] \end{tikzcd}$$

  The functoriality axioms amount to the so called *interchange laws*, meaning $\mathsf{id}_f \otimes \mathsf{id}_g = \mathsf{id}_{f \otimes g}$ so that:

$$\begin{tikzcd} a \arrow[r, bend left=40, "f"{name=F}] \arrow[r, bend right=40, "f'"'{name=F1}] & b \arrow[r, bend left=40, "g"{name=G}] \arrow[r, bend right=40, "g'"'{name=G1}] & c \arrow[r, phantom, "\Longrightarrow"] & a \arrow[r, bend left=40, "f\otimes g"{name=FG}] \arrow[r, bend right=40, "f'\otimes g'"'{name=FG1}] & c \arrow[from=F, to=F1, phantom, "="] \arrow[from=G, to=G1, phantom, "="] \arrow[from=FG, to=FG1, phantom, "="] \end{tikzcd}$$

  and $(\beta \circ \alpha) \otimes (\delta \circ \gamma) = (\beta \otimes \gamma) \circ (\alpha \otimes \gamma)$, so that:

$$\begin{tikzcd} a \arrow[r, bend left=50, "f"{name=F}] \arrow[r, "g"{name=G, description}] \arrow[r, bend right=50, "h"'{name=H}] & b \arrow[r, bend left=50, "k"{name=K}] \arrow[r, "p"{name=P, description}] \arrow[r, bend right=50, "q"'{name=Q}] & c \arrow[r, phantom, "\mapsto"] & a \arrow[r, bend left=50, "f\otimes k"{name=FK}] \arrow[r, bend right=50, "h\otimes q"'{name=HQ}] & c \arrow[r, phantom, "="] & a \arrow[r, bend left=50, "f\otimes k"{name=FK2}] \arrow[r, bend right=50, "h\otimes q"'{name=HQ2}] & c \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=G, to=H, "\beta"] \arrow[Rightarrow, from=K, to=P, "\gamma"] \arrow[Rightarrow, from=P, to=Q, "\delta"] \arrow[Rightarrow, from=FK, to=HQ, "(\beta\otimes\gamma)\circ(\alpha\otimes\gamma)"] \arrow[Rightarrow, from=FK2, to=HQ2, "(\beta\circ\alpha)\otimes(\delta\circ\gamma)"] \end{tikzcd}$$

- For every $a \in \mathcal{B}$, a specified *identity* 1-cell $\mathbb{I} : a \to a$.

- For every $a \xrightarrow{f} b \xrightarrow{g} c \xrightarrow{h} d$, an *associator* 2-cell $\mathbf{a} : (f \otimes g) \otimes h \Rightarrow f \otimes (g \otimes h)$, natural in f, g, and h in the sense that:

$$\begin{tikzcd}
a \arrow[r, bend left=40, "f"] \arrow[r, bend right=40, "f'"'] \arrow[r, phantom, "\Downarrow\alpha"] & b \arrow[r, bend left=40, "g"] \arrow[r, bend right=40, "g'"'] \arrow[r, phantom, "\Downarrow\beta"] & c \arrow[r, bend left=40, "h"] \arrow[r, bend right=40, "h'"'] \arrow[r, phantom, "\Downarrow\gamma"] & d
\end{tikzcd}$$

$$\Longrightarrow \quad \begin{tikzcd}[column sep=huge]
a \arrow[r, bend left=50, "(f\otimes g)\otimes h"] \arrow[r, "f\otimes(g\otimes h)" description] \arrow[r, bend right=50, "f'\otimes(g'\otimes h')"'] \arrow[r, phantom, shift left=4, "\Downarrow\mathbf{a}"] \arrow[r, phantom, shift right=4, "\Downarrow\alpha\otimes(\beta\otimes\gamma)"] & d
\end{tikzcd} = \begin{tikzcd}[column sep=huge]
a \arrow[r, bend left=50, "(f\otimes g)\otimes h"] \arrow[r, "(f'\otimes g')\otimes h'" description] \arrow[r, bend right=50, "f'\otimes(g'\otimes h')"'] \arrow[r, phantom, shift left=4, "\Downarrow(\alpha\otimes\beta)\otimes\gamma"] \arrow[r, phantom, shift right=4, "\Downarrow\mathbf{a}"] & d
\end{tikzcd}$$

- For every $a \xrightarrow{f} b$, *left* and *right* unitor 2-cells $\boldsymbol{\ell} : \mathbb{I} \otimes f \Rightarrow f$ and $\mathbf{r}' : f \Rightarrow f \otimes \mathbb{I}$, natural in f in the sense that:

$$\begin{tikzcd}
a \arrow[r, bend left=40, "f"] \arrow[r, bend right=40, "f'"'] \arrow[r, phantom, "\Downarrow\alpha"] & b
\end{tikzcd} \Longrightarrow \begin{tikzcd}
a \arrow[r, bend left=50, "\mathbb{I}\otimes f"] \arrow[r, "f" description] \arrow[r, bend right=50, "f'"'] \arrow[r, phantom, shift left=4, "\Downarrow\boldsymbol{\ell}"] \arrow[r, phantom, shift right=4, "\Downarrow\alpha"] & b
\end{tikzcd} = \begin{tikzcd}
a \arrow[r, bend left=50, "\mathbb{I}\otimes f"] \arrow[r, "\mathbb{I}\otimes f'" description] \arrow[r, bend right=50, "f'"'] \arrow[r, phantom, shift left=4, "\Downarrow\mathbb{I}\otimes\alpha"] \arrow[r, phantom, shift right=4, "\Downarrow\boldsymbol{\ell}"] & b
\end{tikzcd}$$

$$\text{and} \quad \begin{tikzcd}
a \arrow[r, bend left=50, "f"] \arrow[r, "f\otimes\mathbb{I}" description] \arrow[r, bend right=50, "f'\otimes\mathbb{I}"'] \arrow[r, phantom, shift left=4, "\Downarrow\mathbf{r}'"] \arrow[r, phantom, shift right=4, "\Downarrow\alpha\otimes\mathbb{I}"] & b
\end{tikzcd} = \begin{tikzcd}
a \arrow[r, bend left=50, "f"] \arrow[r, "f'" description] \arrow[r, bend right=50, "f'\otimes\mathbb{I}"'] \arrow[r, phantom, shift left=4, "\Downarrow\alpha"] \arrow[r, phantom, shift right=4, "\Downarrow\mathbf{r}'"] & b
\end{tikzcd}$$

- such that the following 2-cell equations hold:

$$\begin{tikzcd}[column sep=small]
& (f\otimes g)\otimes(h\otimes k) \arrow[dr, "\mathbf{a}"] & \\
((f\otimes g)\otimes h)\otimes k \arrow[ur, "\mathbf{a}"] \arrow[d, "\mathbf{a}\otimes\mathsf{id}"'] & \substack{\text{bicategorical}\\ \text{pentagon}\\ =} & f\otimes(g\otimes(h\otimes k)) \\
(f\otimes(g\otimes h))\otimes k \arrow[rr, "\mathbf{a}"'] & & f\otimes((g\otimes h)\otimes k) \arrow[u, "\mathsf{id}\otimes\mathbf{a}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
(f\otimes\mathbb{I})\otimes g \arrow[r, "\mathbf{a}"] & f\otimes(\mathbb{I}\otimes g) \arrow[d, "\mathsf{id}\otimes\boldsymbol{\ell}"] \\
f\otimes g \arrow[u, "\mathbf{r}'\otimes\mathsf{id}"] \arrow[r, equal] \arrow[ur, phantom, "\substack{\text{bicategorical}\\ \text{rectangle}\\ =}"] & f\otimes g
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=small]
(\mathbb{I}\otimes f)\otimes g \arrow[rr, "\mathbf{a}"] \arrow[dr, "\boldsymbol{\ell}\otimes\mathsf{id}"'] & \substack{\text{skew left}\\ =} & \mathbb{I}\otimes(f\otimes g) \arrow[dl, "\boldsymbol{\ell}"] \\
& f\otimes g &
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
& f\otimes g \arrow[dl, "\mathbf{r}'"'] \arrow[dr, "\mathsf{id}\otimes\mathbf{r}'"] & \\
(f\otimes g)\otimes\mathbb{I} \arrow[rr, "\mathbf{a}"'] & \substack{\text{skew right}\\ =} & f\otimes(g\otimes\mathbb{I})
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
\mathbb{I} \arrow[rr, equal] \arrow[dr, "\mathbf{r}'"'] & \substack{\text{skew}\\ \text{triangle}\\ =} & \mathbb{I} \\
& \mathbb{I}\otimes\mathbb{I} \arrow[ur, "\boldsymbol{\ell}"'] &
\end{tikzcd}$$

We say that a skew bicategory is *associative* when the associator is invertible, and similarly say it is *left/right-unital* when the corresponding unitor is invertible. Lack and Street [52] prove that every bicategory is an associative and unital skewed bicategory by generalising Kelly's techniques [48] from the monoidal to the bicategorical setting.

D.2 A bicategory of sorting systems

Let $\mathbf{Prof}_{\vdash} \coloneqq \big(\mathbf{Prof}, (\not\twoheadrightarrow), (\Rightarrow), (\circ), \mathsf{id}, (\olessthan), \mathbb{V}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}\big)$ be Fiore, Gambino, Hyland, and Winskel's Kleisli bicategory of profunctors equipped with the substitution tensor product as diagrammatic composition, as we described in §B.1. Continuing with the same notation, we write $P : \mathbf{R} \twoheadrightarrow \mathbf{S}$ when $P : \mathbf{R}\text{-}\mathsf{sort} \not\twoheadrightarrow \mathbf{S}\text{-}\mathsf{Bind}$ in $\mathbf{Prof}_{\vdash}$, i.e., a functor $P : \mathbf{R}\text{-}\mathsf{sort} \times \mathbf{S}\text{-}\mathsf{Bind}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$.

First, we prove Lemma 5.4, generalising it to our candidate skew bicategory:

Lemma D.1 *We have a coend-preserving isomorphism, natural in the* **R***-structures* $P, Q \in \mathbf{R\text{-}Struct}$*:*

$$[p, e]_\Delta \mapsto [p, e]_\Delta \,:\, (P \otimes Q)|_{\mathsf{bnd}} \xrightarrow{\cong} (P|_{\mathsf{bnd}}) \olessthan (Q|_{\mathsf{bnd}})$$

Proof. For each bindable sort $s \vDash \mathsf{bnd}$, both coends are for the same functor $\Delta \mapsto P_s\Delta \times Q^{\mathrm{Env}}_\Delta$ □

Theorem D.2 *Sorting systems* **R***, their* 1*-cells* $P : \mathbf{R} \twoheadrightarrow \mathbf{S}$*, and natural transformations between them* $\alpha : P \Rightarrow Q$ *form a skew, associative, and right-unital bicategory:*

$$\mathbf{SortSys} \coloneqq \left(\mathbf{SubSet}, (\twoheadrightarrow), (\Rightarrow), (\circ), \mathsf{id}, (\otimes), \mathbb{I}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r}^{-1}\right)$$

Proof. Natural transformations with vertical composition and identities form a category $\mathbf{SortSys}(P, Q)$. Diagrammatic composition of 1-cells is the composite functor:

$$\begin{tikzcd}
\mathbf{SortSys}(\mathbf{R}, \mathbf{S}) \times \mathbf{SortSys}(\mathbf{S}, \mathbf{T}) \arrow[rr, "(\otimes)"] \arrow[d, "\mathsf{Id} \times -|_{\mathsf{bnd}}"'] & \arrow[d, phantom, ":="] & \mathbf{SortSys}(\mathbf{R}, \mathbf{T}) \arrow[d, equal] \\
\mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{sort} \times \mathbf{S}\text{-}\mathsf{Bind}^{\mathrm{op}}_{\vdash}\right) \times \mathbf{PSh}\left(\mathbf{S}\text{-}\mathsf{Bind} \times \mathbf{T}\text{-}\mathsf{Bind}^{\mathrm{op}}_{\vdash}\right) \arrow[rr, "(\olessthan)"'] & & \mathbf{PSh}\left(\mathbf{R}\text{-}\mathsf{sort} \times \mathbf{T}\text{-}\mathsf{Bind}^{\mathrm{op}}_{\vdash}\right)
\end{tikzcd}$$

The naturality of the associator and unitors amounts to their naturality as natural transformations as applied to the components of the natural transformations α, and β and γ (in the notation of page 38). The right-unitor and the associator are invertible.

The bicategorical pentagon does not mention $\mathbb{I}$ at all, and so reduces to the bicategorical pentagon for $(\olessthan)$ using Lemma D.1:

$$\begin{tikzcd}[column sep=tiny]
 & & & & (P \otimes Q) \otimes (L \otimes K) \arrow[rrrrd, "\mathbf{a}"] \arrow[d, equal] & & & & \\
((P \otimes Q) \otimes L) \otimes K \arrow[rrrru, "\mathbf{a}"] \arrow[rrd, equal] \arrow[rrrdd, bend right, "\mathsf{a} \otimes \mathsf{id}"'] & & \overset{\mathbf{a}\text{ def}}{=} & & (P \olessthan Q|_{\mathsf{bnd}}) \olessthan (L|_{\mathsf{bnd}} \olessthan K|_{\mathsf{bnd}}) \arrow[rrd, "\mathbf{a}"] & & \overset{\mathbf{a}\text{ def}}{=} & & P \otimes (Q \otimes (L \otimes K)) \arrow[lld, equal] \\
 & & ((P \olessthan Q|_{\mathsf{bnd}}) \olessthan L|_{\mathsf{bnd}}) \olessthan K|_{\mathsf{bnd}} \arrow[rru, "\mathbf{a}"] \arrow[rd, "\mathbf{a} \olessthan \mathsf{id}"] & & \overset{\text{bicategorical pentagon}}{=} & & P \olessthan (Q|_{\mathsf{bnd}} \olessthan (L|_{\mathsf{bnd}} \olessthan K|_{\mathsf{bnd}})) & & \\
 & \overset{\mathbf{a}\text{ def}}{=} & & (P \olessthan (Q|_{\mathsf{bnd}} \olessthan L|_{\mathsf{bnd}})) \olessthan K|_{\mathsf{bnd}} \arrow[rr, bend right, "\mathbf{a}"'] \arrow[ld, equal] & & P \olessthan ((Q|_{\mathsf{bnd}} \olessthan L|_{\mathsf{bnd}}) \olessthan K|_{\mathsf{bnd}}) \arrow[ru, "\mathsf{id} \olessthan \mathbf{a}"] \arrow[rd, equal] & & \overset{\mathbf{a}\text{ def}}{=} & \\
 & & (P \otimes (Q \otimes L)) \otimes K \arrow[rrrr, bend right, "\mathbf{a}"'] & & \overset{\mathbf{a}\text{ def}}{=} & & P \otimes ((Q \otimes L) \otimes K) \arrow[rruuu, bend right, "\mathsf{id} \otimes \mathbf{a}"'] & &
\end{tikzcd}$$

The bicategorical rectangle only involves $\mathbb{I}$, but it always appears to the right of $(\otimes)$, and so becomes the neutral element for $(\olessthan)$ and the rectangle axiom reduces to that of $\mathbf{Prof}_{\vdash}$:

$$\begin{tikzcd}
(P \otimes \mathbb{I}) \otimes Q \arrow[rrr, "\mathbf{a}"] \arrow[rd, equal] & & & P \otimes (\mathbb{I} \otimes Q) \arrow[ld, equal] \arrow[ddd, bend left, "\mathsf{id} \otimes \boldsymbol{\ell}"] \\
\overset{\text{def}}{=} & (P \olessthan \mathbb{V}) \olessthan Q|_{\mathsf{bnd}} \arrow[r, "\overset{\text{def}}{=}", "\mathbf{a}"'] & P \olessthan (\mathbb{V} \olessthan Q|_{\mathsf{bnd}}) \arrow[d, bend left, "\mathsf{id} \olessthan \boldsymbol{\ell}"'] & \overset{\text{def}}{=} \\
 & P \olessthan Q|_{\mathsf{bnd}} \arrow[u, bend left, "\mathbf{r}^{-1} \olessthan \mathsf{id}"'] \arrow[r, equal, "\overset{\text{bicategorical rectangle}}{=}"] & P \olessthan Q|_{\mathsf{bnd}} & \\
P \otimes Q \arrow[uuu, bend left, "\mathbf{r}^{-1} \otimes \mathsf{id}"] \arrow[ru, equal] \arrow[rrr, equal, "\overset{\text{id}}{=}"] & & & P \otimes Q \arrow[lu, equal]
\end{tikzcd}$$

The skew right axiom similarly only contains $\mathbb{I}$ to the right of $(\otimes)$ and so reduces to the bicategorical one:

$$\begin{tikzcd}
 & & P \otimes Q \arrow[lld, bend right, "\mathbf{r}'"'] \arrow[rrd, bend left, "\mathsf{id} \otimes \mathbf{r}'"] \arrow[d, equal] & & \\
 & \overset{\text{def}}{=} & P \olessthan Q|_{\mathsf{bnd}} \arrow[ld, "\mathbf{r}'"'] \arrow[rd, "\mathsf{id} \olessthan \mathbf{r}'"] & \overset{\text{def}}{=} & \\
 & (P \olessthan Q|_{\mathsf{bnd}}) \olessthan \mathbb{V} \arrow[rr, "\mathbf{a}"'] \arrow[r, phantom, "\overset{\text{skew right}}{=}"] & & P \olessthan (Q|_{\mathsf{bnd}} \olessthan \mathbb{V}) & \\
(P \otimes Q) \otimes \mathbb{I} \arrow[ru, equal] \arrow[rrrr, "\mathbf{a}"', "\overset{\text{def}}{=}"] & & & & P \otimes (Q \otimes \mathbb{I}) \arrow[lu, equal]
\end{tikzcd}$$

For the skew left and skew triangle axioms, we do have $\mathbb{I}$ appearing to the left of $(\otimes)$, and so we need to split into cases depending on the sort $s \in \mathbf{R}\text{-}\mathsf{sort}$. When $s \nvDash \mathsf{bnd}$, we have $(\mathbb{I} \otimes P)_s = \mathbb{0}$, and so the diagram denotes two morphisms out of an initial object, and so commutes. For $s \vDash \mathsf{bnd}$, we have:

$$\begin{tikzcd}
((\mathbb{I} \otimes P) \otimes Q)_s \arrow[rrrr, bend left, "\mathbf{a}", "\overset{\text{def}}{=}"'] \arrow[rd, equal] \arrow[rrd, bend right, "\boldsymbol{\ell} \otimes \mathsf{id}"'] & & & & (\mathbb{I} \otimes (P \otimes Q))_s \arrow[ld, equal] \arrow[lldd, bend left, "\boldsymbol{\ell}"] \\
 & (\mathbb{V}_s \olessthan P|_{\mathsf{bnd}}) \olessthan Q|_{\mathsf{bnd}} \arrow[rr, bend left, "\mathbf{a}"] \arrow[rd, "\boldsymbol{\ell} \olessthan \mathsf{id}"'] & & \mathbb{V}_s \olessthan (P|_{\mathsf{bnd}} \olessthan Q|_{\mathsf{bnd}}) \arrow[ld, "\boldsymbol{\ell}"] & \\
 & \overset{\text{def}}{=} & P_s \olessthan Q|_{\mathsf{bnd}} \arrow[d, equal] \arrow[u, phantom, "\overset{\text{skew left}}{=}"] & \overset{\text{def}}{=} & \\
 & & (P \otimes Q)_s & &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbb{I}_s \arrow[rrrr, equal] \arrow[rd, equal] \arrow[rrddd, bend right, "\mathbf{r}'"'] & & & & \mathbb{I}_s \\
 & \mathbb{I}_s \arrow[rr, equal, "\overset{\text{def}}{=}"] \arrow[rd, "\mathbf{r}'"'] & & \mathbb{I}_s \arrow[ru, equal] & \\
 & \overset{\text{def}}{=} & \mathbb{V}_s \olessthan \mathbb{V} \arrow[ru, "\boldsymbol{\ell}"'] \arrow[d, equal] \arrow[u, phantom, "\overset{\text{skew triangle}}{=}"] & \overset{\text{def}}{=} & \\
 & & (\mathbb{I} \otimes \mathbb{I})_s \arrow[rruuu, bend right, "\boldsymbol{\ell}"'] & &
\end{tikzcd}$$

completing the proof. □

D.3 Exponentiation qua right Kan lifts

In a skew bicategory $\mathcal{B}$, every 1-cell $g : b \to c$ induces the *post-composition* functor $(- \otimes g) : \mathcal{B}(a, b) \to \mathcal{B}(a, c)$. Given any 1-cell $f : a \to c$, a *right Kan lift* of f along g is a universal arrow from $(- \otimes g)$ to f, i.e., a 1-cell $(f \mathbin{\leftarrow\!\otimes} g) : a \to b$ and a 2-cell $\mathsf{eval} : (f \mathbin{\leftarrow\!\otimes} g) \otimes g \Rightarrow f$ that is terminal among these 2-cells:

$$\begin{tikzcd}
a \arrow[r, "f \leftarrow\otimes g"] \arrow[rd, "f"'] & b \arrow[d, "g"] \arrow[ld, phantom, "\Leftarrow \mathsf{eval}"] \\
 & c
\end{tikzcd}
\quad \forall \quad
\begin{tikzcd}
a \arrow[r, "h"] \arrow[rd, "f"'] & b \arrow[d, "g"] \arrow[ld, phantom, "\Leftarrow \alpha"] \\
 & c
\end{tikzcd}
\ \exists! \
\begin{tikzcd}
a \arrow[r, bend left, "h"] \arrow[r, bend right, "f \leftarrow\otimes g"'] \arrow[r, phantom, "\Downarrow\beta"] & b
\end{tikzcd}
. \quad \alpha =
\begin{tikzcd}
a \arrow[r, bend left, "h"] \arrow[r, "f \leftarrow\otimes g"'] \arrow[r, phantom, bend left=20, "\Downarrow\beta"] \arrow[rd, bend right, "f"'] & b \arrow[d, "g"] \arrow[ld, phantom, "\Leftarrow \mathsf{eval}"] \\
 & c
\end{tikzcd}$$

We say that a skew bicategory is *(right)-liftal* when it has all right Kan lifts along all 1-cells. This terminology is not standard. We derive it from Street [70], who called the dual notion of a bicategory with all right Kan extensions *extensional*.

The bicategory of profunctors $\mathbf{Prof} := (\mathbf{Cat}, (\nrightarrow), (\Rightarrow), (\otimes), \mathbb{1}, \mathbf{a}, \boldsymbol{\ell}, \mathbf{r})$ is well-known to be liftal and extensional. Fig D.1 presents the right Kan lifts explicitly. We derive the right Kan lifts for the other two

Prof : $$(P \leftarrow\!\!\otimes Q)_a b \coloneqq \int_{c \in \mathbb{C}} (P_a c)^{Q_b c}$$

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "P \leftarrow\!\!\otimes Q", "\shortmid" marking] \arrow[dr, "P"', "\shortmid" marking] & \mathbb{B} \arrow[d, "Q", "\shortmid" marking] \arrow[d, phantom, "\Leftarrow \text{eval}"', shift right=4] \\
 & \mathbb{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\left(\int_{c' \in \mathbb{C}} (P_a c')^{Q_b c'}\right) \times Q_b c \arrow[r, "{[-]_b}"] \arrow[d, "\pi_c \times \mathsf{id}"'] \arrow[dr, phantom, "=:"] & ((P \leftarrow\!\!\otimes Q) \otimes Q)_a c \arrow[d, "\mathsf{eval}"] \\
(P_a c)^{Q_b c} \times Q_b c \arrow[r, "\mathsf{eval}"'] & Q_a c
\end{tikzcd}$$

Fig. D.1. Right Kan lifts in the bicategory of profunctors.

Prof$_\vdash$: $$(P \leftarrow\!\!\varolessthan Q) \coloneqq (P \leftarrow\!\!\otimes Q^{\mathrm{Env}}) \quad \text{i.e.: } (P \leftarrow\!\!\varolessthan Q)_a \Gamma \coloneqq \int_{\Delta \in \mathbb{C}_\vdash} (P_a \Delta)^{Q^{\mathrm{Env}}_\Delta \Gamma}$$

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "P \leftarrow\!\!\varolessthan Q", "\shortmid\shortmid" marking] \arrow[dr, "P"', "\shortmid\shortmid" marking] & \mathbb{B} \arrow[d, "Q", "\shortmid\shortmid" marking] \arrow[d, phantom, "\Leftarrow \text{eval}"', shift right=4] \\
 & \mathbb{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\left(\int_{\Xi \in \mathbb{C}_\vdash} (P_a \Xi)^{Q^{\mathrm{Env}}_\Delta \Xi}\right) \times Q^{\mathrm{Env}}_\Delta \Gamma \arrow[r, "{[-]_\Delta}"] \arrow[d, "\pi_\Gamma \times \mathsf{id}"'] \arrow[dr, phantom, "=:"] & ((P \leftarrow\!\!\varolessthan Q) \varolessthan Q)_a \Gamma \arrow[d, "\mathsf{eval}"] \\
(P_a \Gamma)^{Q^{\mathrm{Env}}_\Delta \Gamma} \times Q^{\mathrm{Env}}_\Delta \Gamma \arrow[r, "\mathsf{eval}"'] & Q_a \Gamma
\end{tikzcd}$$

SortSys : $$(P \leftarrow\!\!\otimes Q) \coloneqq (P \leftarrow\!\!\varolessthan Q|_{\mathsf{bnd}}) \quad \text{i.e.: } (P \leftarrow\!\!\otimes Q)_a \Gamma \coloneqq \int_{\Delta \in \mathbb{C}_\vdash} (P_a \Delta)^{(Q|_{\mathsf{bnd}})^{\mathrm{Env}}_\Delta \Gamma}$$

$$\begin{tikzcd}
\mathbf{R} \arrow[r, "P \leftarrow\!\!\otimes Q", "\times" marking] \arrow[dr, "P"', "\times" marking] & \mathbf{S} \arrow[d, "Q", "\times" marking] \arrow[d, phantom, "\Leftarrow \text{eval}"', shift right=4] \\
 & \mathbf{T}
\end{tikzcd}
\qquad
\begin{tikzcd}
\left(\int_{\Xi \in \mathbb{C}_\vdash} (P_a \Xi)^{(Q|_{\mathsf{bnd}})^{\mathrm{Env}}_\Delta \Xi}\right) \times (Q|_{\mathsf{bnd}})^{\mathrm{Env}}_\Delta \Gamma \arrow[r, "{[-]_\Delta}"] \arrow[d, "\pi_\Gamma \times \mathsf{id}"'] \arrow[dr, phantom, "=:"] & ((P \leftarrow\!\!\otimes (Q|_{\mathsf{bnd}})) \varolessthan (Q|_{\mathsf{bnd}}))_a \Gamma \arrow[d, "\mathsf{eval}"] \\
(P_a \Gamma)^{(Q|_{\mathsf{bnd}})^{\mathrm{Env}}_\Delta \Gamma} \times (Q|_{\mathsf{bnd}})^{\mathrm{Env}}_\Delta \Gamma \arrow[r, "\mathsf{eval}"'] & (Q|_{\mathsf{bnd}})_a \Gamma
\end{tikzcd}$$

Fig. D.2. Deriving right Kan lifts in the Kleisli bicategory of profunctors; and in the bicategory of sorting systems.

bicategories from the lifts of bimodules:

Theorem D.3 *The Kleisli bicategory* **Prof**$_\vdash$ *and the skew bicategory of sorting systems* **SortSys** *are liftal. Fig D.2 has their right Kan lifts.*

Proof. The Kan lifts of **Prof**$_\vdash$ are known, but for completeness's sake, we prove their existence too. To define the situation as on the left, use the situation on the right:

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "P \leftarrow\!\!\varolessthan Q", "\shortmid\shortmid" marking] \arrow[dr, "P"', "\shortmid\shortmid" marking] & \mathbb{B} \arrow[d, "Q", "\shortmid\shortmid" marking] \arrow[d, phantom, "\Leftarrow \text{eval}"', shift right=4] \\
 & \mathbb{C}
\end{tikzcd}
\quad \text{take:} \quad
\begin{tikzcd}
\mathbb{A} \arrow[r, "P \leftarrow\!\!\otimes Q^{\mathrm{Env}}", "\shortmid" marking] \arrow[dr, "P"', "\shortmid" marking] & \mathbb{B}_\vdash \arrow[d, "Q^{\mathrm{Env}}", "\shortmid" marking] \arrow[d, phantom, "\Leftarrow \text{eval}"', shift right=4] \\
 & \mathbb{C}_\vdash
\end{tikzcd}$$

To show that this lift is universal, consider any other lift:

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "L", "\shortparallel"{marking}] \arrow[dr, "P"', "\shortparallel"{marking}] & \mathbb{B} \arrow[d, "Q", "\shortparallel"{marking}] \arrow[dl, Rightarrow, shorten <=6pt, shorten >=14pt, "\alpha"] \\
& \mathbb{C}
\end{tikzcd}
\text{ i.e.: }
\begin{tikzcd}
\mathbb{A} \arrow[r, "L", "\shortmid"{marking}] \arrow[dr, "P"', "\shortmid"{marking}] & \mathbb{B}_{\vdash} \arrow[d, "Q^{\mathrm{Env}}", "\shortmid"{marking}] \arrow[dl, Rightarrow, shorten <=6pt, shorten >=14pt, "\alpha"] \\
& \mathbb{C}_{\vdash}
\end{tikzcd}
\implies \exists!\
\begin{tikzcd}
\mathbb{A} \arrow[r, bend left, "L", "\shortmid"{marking}] \arrow[r, bend right, "P \mathrel{\leftarrow\!\otimes} Q"', "\shortmid"{marking}] \arrow[r, phantom, "\Downarrow\beta"] & \mathbb{B}_{\vdash}
\end{tikzcd}
.\alpha =
\begin{tikzcd}
\mathbb{A} \arrow[rr, bend left, "L", "\shortmid"{marking}] \arrow[rr, bend right, "P \mathrel{\leftarrow\!\otimes} Q"', "\shortmid"{marking}] \arrow[rr, phantom, "\Downarrow\beta"] \arrow[drr, bend right, "P"', "\shortmid"{marking}] & & \mathbb{B}_{\vdash} \arrow[d, "Q^{\mathrm{Env}}", "\shortmid"{marking}] \arrow[dl, Rightarrow, "\mathsf{eval}"'] \\
& & \mathbb{C}_{\vdash}
\end{tikzcd}$$

For such a β we therefore have:

$$\begin{aligned} \alpha &= \mathsf{eval}\circ(\beta \otimes Q^{\mathrm{Env}}) \\ &= \mathsf{eval}\circ(\beta \olessthan Q) \end{aligned}
\qquad \text{i.e.: } \alpha =
\begin{tikzcd}
\mathbb{A} \arrow[rr, bend left, "L", "\shortparallel"{marking}] \arrow[rr, bend right, "P \mathrel{\leftarrow\!\olessthan} Q"', "\shortparallel"{marking}] \arrow[rr, phantom, "\Downarrow\beta"] \arrow[drr, bend right, "P"', "\shortparallel"{marking}] & & \mathbb{B}_{\vdash} \arrow[d, "Q^{\mathrm{Env}}", "\shortparallel"{marking}] \arrow[dl, Rightarrow, "\mathsf{eval}"'] \\
& & \mathbb{C}_{\vdash}
\end{tikzcd}$$

and therefore $(P \mathrel{\leftarrow\!\olessthan} Q, \mathsf{eval})$ is a right Kan lift in $\mathbf{Prof}_{\vdash}$. The formula in Fig D.2 spells this definition out in detail.

Next, we show that **SortSys** has right Kan lifts, inherited from $\mathbf{Prof}_{\vdash}$.

$$\text{For: }
\begin{tikzcd}
\mathbf{R} \arrow[r, "P \mathrel{\leftarrow\!\otimes} Q", "\times"{marking}] \arrow[dr, "P"', "\times"{marking}] & \mathbf{S} \arrow[d, "Q", "\times"{marking}] \arrow[dl, Rightarrow, shorten <=6pt, shorten >=14pt, "\mathsf{eval}"] \\
& \mathbf{T}
\end{tikzcd}
\qquad \text{take: }
\begin{tikzcd}
\mathbf{R}\text{-}\mathsf{sort} \arrow[r, "P \mathrel{\leftarrow\!\olessthan} Q|_{\mathsf{bnd}}", "\shortparallel"{marking}] \arrow[dr, "P"', "\shortparallel"{marking}] & \mathbf{S}\text{-}\mathsf{Bind} \arrow[d, "Q|_{\mathsf{bnd}}", "\shortparallel"{marking}] \arrow[dl, Rightarrow, shorten <=6pt, shorten >=14pt, "\mathsf{eval}"] \\
& \mathbf{T}\text{-}\mathsf{Bind}
\end{tikzcd}$$

To show that this lift is universal, consider any other lift:

$$\begin{tikzcd}
\mathbf{R} \arrow[r, "L", "\times"{marking}] \arrow[dr, "P"', "\times"{marking}] & \mathbf{S} \arrow[d, "Q", "\times"{marking}] \arrow[dl, Rightarrow, shorten <=6pt, shorten >=14pt, "\alpha"] \\
& \mathbf{T}
\end{tikzcd}
\qquad \text{i.e.: }
\begin{tikzcd}
\mathbf{R}\text{-}\mathsf{sort} \arrow[r, "L", "\shortparallel"{marking}] \arrow[dr, "P"', "\shortparallel"{marking}] & \mathbf{S}\text{-}\mathsf{Bind} \arrow[d, "Q|_{\mathsf{bnd}}", "\shortparallel"{marking}] \arrow[dl, Rightarrow, shorten <=6pt, shorten >=14pt, "\alpha"] \\
& \mathbf{T}\text{-}\mathsf{Bind}
\end{tikzcd}$$

$$\implies \exists!\
\begin{tikzcd}
\mathbf{R} \arrow[r, bend left, "L", "\shortparallel"{marking}] \arrow[r, bend right, "P \mathrel{\leftarrow\!\olessthan} Q"', "\shortparallel"{marking}] \arrow[r, phantom, "\Downarrow\beta"] & \mathbf{S}\text{-}\mathsf{Bind}
\end{tikzcd}
.\alpha =
\begin{tikzcd}
\mathbf{R} \arrow[rr, bend left, "L", "\shortparallel"{marking}] \arrow[rr, bend right, "P \mathrel{\leftarrow\!\olessthan} Q|_{\mathsf{bnd}}"', "\shortparallel"{marking}] \arrow[rr, phantom, "\Downarrow\beta"] \arrow[drr, bend right, "P"', "\shortparallel"{marking}] & & \mathbf{S}\text{-}\mathsf{Bind} \arrow[d, "Q|_{\mathsf{bnd}}", "\shortparallel"{marking}] \arrow[dl, Rightarrow, "\mathsf{eval}"'] \\
& & \mathbf{T}\text{-}\mathsf{Bind}
\end{tikzcd}$$

But for such a β we therefore have:

$$\begin{aligned} \alpha &= \mathsf{eval}\circ(\beta \olessthan Q|_{\mathsf{bnd}}) \\ &= \mathsf{eval}\circ(\beta \otimes Q) \end{aligned}
\qquad \text{i.e.: } \alpha =
\begin{tikzcd}
\mathbf{R} \arrow[rr, bend left, "L", "\times"{marking}] \arrow[rr, bend right, "P \mathrel{\leftarrow\!\otimes} Q"', "\times"{marking}] \arrow[rr, phantom, "\Downarrow\beta"] \arrow[drr, bend right, "P"', "\times"{marking}] & & \mathbf{S} \arrow[d, "Q", "\times"{marking}] \arrow[dl, Rightarrow, "\mathsf{eval}"'] \\
& & \mathbf{T}
\end{tikzcd}$$

and therefore ($P \mathbin{\leftarrow\!\otimes} Q, \mathsf{eval}$) is a right Kan lift in **SortSys**. The formula in Fig D.2 spells this definition out in detail. □

Unfolding the definition of skew monoidal exponentials, we deduce Prop. B.1.

E Proving the representation theorem

We use a skew version of a well known parameterized induction principle. We include an indexed variation that we use when the inductive structure F is given by a coproduct $\coprod_i F_i$ of alternative structutres:

Lemma E.1 *Let $\mathcal{C}$ be a right-closed skew monoidal category, $F : \underline{\mathcal{C}} \to \underline{\mathcal{C}}$ a functor, and $F\boldsymbol{\mu}F \xrightarrow{\mathsf{roll}} \boldsymbol{\mu}F$ an initial algebra for F. Each $f : (F(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \to x$ has a unique $\mathsf{fold} f : (\boldsymbol{\mu}F) \otimes p \to x$ satisfying:*

$$\begin{tikzcd}
(F\boldsymbol{\mu}F) \otimes p \arrow[r, "F(\mathsf{curry}(\mathsf{fold} f)) \otimes \mathsf{id}"] \arrow[d, "\mathsf{roll} \otimes \mathsf{id}"'] \arrow[dr, phantom, "\text{parameterised induction} \atop ="] & (F(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \arrow[d, "f"] \\
(\boldsymbol{\mu}F) \otimes p \arrow[r, dashed, "\mathsf{fold} f"'] & x
\end{tikzcd}$$

When $\underline{\mathcal{C}}$ has I-indexed coproducts, $F = \{\!|C_i : F_i \,|\, i \in I|\!\}$, let $\mathsf{roll} = \left[\underline{C_i}\right]_i$ and then for every I-indexed family of arrows $f = \left(f_i : (F_i(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \to x\right)_i$ there is a unique arrow $\mathsf{fold} f : (\boldsymbol{\mu}F) \otimes p \to x$ satisfying:

$$\begin{tikzcd}
(F_i\boldsymbol{\mu}F) \otimes p \arrow[r, "F_i(\mathsf{curry}(\mathsf{fold} f)) \otimes \mathsf{id}"] \arrow[d, "\underline{C_i} \otimes \mathsf{id}"'] \arrow[dr, phantom, "\text{indexed} \atop {\text{parameterised induction} \atop =}"] & (F_i(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \arrow[d, "f_i"] \\
(\boldsymbol{\mu}F) \otimes p \arrow[r, "\mathsf{fold} f"'] & x
\end{tikzcd} \qquad \text{for all } i \in I$$

Proof. The parameterised induction principle amounts to appealing to initiality w.r.t. the following algebra and uncurrying:

$$\mathsf{curry} f : F(x \mathbin{\leftarrow\!\otimes} p) \to (x \mathbin{\leftarrow\!\otimes} p)$$

For the indexed version, assume $\mathcal{C}$ has I-ary coproducts. Then $(\otimes p)$ distributes over these coproducts, and so the following diagrams are both coproduct diagrams:

$$\left(\underline{C_i} \otimes \mathsf{id} : (F_i x) \otimes p \to (Fx) \otimes p\right)_{i\in I} \qquad \left(\underline{C_i} \otimes \mathsf{id} : (F_i(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \to (F(x \mathbin{\leftarrow\!\otimes} p)) \otimes p\right)_{i\in I}$$

We therefore have:

$$\begin{tikzcd}
(F\boldsymbol{\mu}F) \otimes p \arrow[r, "\mathsf{curry} g \otimes \mathsf{id}"] \arrow[d, "\mathsf{roll} \otimes \mathsf{id}"'] \arrow[dr, phantom, "="] & (F(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \arrow[d, "{[f_i]_i}"] \\
(\boldsymbol{\mu}F) \otimes p \arrow[r, "g"'] & x
\end{tikzcd}$$

iff, forall $i \in I$:

$$\begin{tikzcd}
(F\boldsymbol{\mu}F) \otimes p \arrow[rr, "F(\mathsf{curry} g) \otimes \mathsf{id}"] \arrow[dd, "\mathsf{roll} \otimes \mathsf{id}"'] & & (F(x \mathbin{\leftarrow\!\otimes} p)) \otimes p \arrow[dd, "{[f_i]_i}"] \\
& (F_i(\boldsymbol{\mu}F)) \otimes p \arrow[ul, "\underline{C_i} \otimes \mathsf{id}"'] \arrow[r, phantom, "="] & {} \\
(\boldsymbol{\mu}F) \otimes p \arrow[rr, "g"'] & & x
\end{tikzcd}$$

Since we have:

$$\begin{tikzcd}
(F\boldsymbol{\mu}F)\otimes p \arrow[rrrrr, "F(\mathsf{curry}g)\otimes\mathsf{id}", "C_i\otimes\mathsf{id}\text{-naturality}"'] \arrow[dd, "\mathsf{roll}\otimes\mathsf{id}"'] & & & & & (F(x \leftarrow\!\otimes p))\otimes p \arrow[dd, "{[f_i]_i}"] \\
& \overset{\text{coproducts}}{=} & (F_i(\boldsymbol{\mu}F))\otimes p \arrow[ull, "C_i\otimes\mathsf{id}"'] \arrow[dll, "\underline{C_i}\otimes\mathsf{id}"] \arrow[r, "F_i(\mathsf{curry}\overline{g})\otimes\mathsf{id}", "\text{indexed parameterised induction}"'] & (F_i(x \leftarrow\!\otimes p))\otimes p \arrow[urr, "C_i\otimes\mathsf{id}"] \arrow[drr, "f_i"'] & \overset{\text{coproducts}}{=} & \\
(\boldsymbol{\mu}F)\otimes p \arrow[rrrrr, "g"'] & & & & & x
\end{tikzcd}$$

And so the $\mathsf{fold}f$ is the unique morphism satisfying the indexed universal property. □

Turning to prove the General Representation Thm B.2, let $\mathcal{C}$ be an assocciative, right-closed skew monoidal category, and $\mathbf{O} : \mathcal{C}_\bullet \to \mathcal{C}_\bullet$ be pointed-strong functor. Consider any $x \in \mathcal{C}$ and an initial algebra over $\$^{\mathbf{O}} \coloneqq \boldsymbol{\mu}z.(\underline{\mathbf{O}}z) \amalg \mathbb{I} \amalg (x \otimes z)$ given by $\left(\$^{\mathbf{O}}x, [\![-]\!], \mathsf{var}, \mathsf{meta}\right)$.

We will proceed in stages, establishing the desiderata for initiality over x:

- Construct the simultaneous substitution morphism $-[-] : \$^{\mathbf{O}}x \otimes \$^{\mathbf{O}}x \to \$^{\mathbf{O}}x$.
- Show its structural induction properties (Lemma E.2).
- Show it equips $\$^{\mathbf{O}}x$ with a substitution monoid structure (Lemma E.3 and Lemma E.5).
- The $\mathbf{O}$-algebra structure is compatible with this substitution, by the ‘op case’ equation of Lemma E.2.

Then, letting $\left(\mathbf{M}, \mathsf{menv} : x \to \underline{\mathbf{M}}\right)$ be any $\mathbf{O}$-monoid over x:

- Construct the mediating morphism $\mathbf{M}\,[\![-]\!] : \$^{\mathbf{O}}x \to \underline{\mathbf{M}}$.
- Show it is a monoid homomorphism (Lemma E.10).
- Show it is an $\mathbf{O}$-homomorphism (‘op preservation’ equation in (E.1) below).
- Show it extends env along meta (Lemma E.9).
- Show it is the unique such morphism (Lemma E.8).

The remainder of this section realises these steps.

First, define variable substitution $-[-] : \$^{\mathbf{O}}x \otimes \$^{\mathbf{O}}x \to \$^{\mathbf{O}}x$ by indexed parameterised induction, as the unique arrow satisfying:

$$\begin{tikzcd}
& (\mathbf{O}(\$x \leftarrow\!\otimes \$x))\otimes\$x \arrow[dr, "\mathsf{str}^{\mathbf{O}}_{\$x\leftarrow\otimes\$x,\mathsf{var}}"] & \\
(\mathbf{O}\$x)\otimes\$x \arrow[ur, "\mathbf{O}(\mathsf{curry}(-[-]))\otimes\mathsf{id}"] \arrow[dd, "{[\![-]\!]\otimes\mathsf{id}}"'] & & \mathbf{O}((\$x \leftarrow\!\otimes \$x)\otimes\$x) \arrow[d, "\mathbf{O}\mathsf{eval}"] \\
& \overset{\substack{\text{def op}\\\text{case}}}{=} & \mathbf{O}(\$x) \arrow[d, "{[\![-]\!]}"] \\
(\mathbf{O}\$x)\otimes\$x \arrow[rr, "{-[-]}"'] & & \$x
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathbb{I}\otimes\$x \arrow[dl, "\mathsf{var}\otimes\mathsf{id}"'] \arrow[dr, "\ell"] & \\
\$x\otimes\$x \arrow[rr, "{-[-]}"', "\overset{\substack{\text{var}\\\text{case}}}{=}"] & & \$x
\end{tikzcd}$$

$$\begin{tikzcd}
& (x\otimes(\$x \leftarrow\!\otimes \$x))\otimes\$x \arrow[dr, "\mathbf{a}"] & \\
(x\otimes\$x)\otimes\$x \arrow[ur, "(\mathsf{id}\otimes\mathsf{curry}(-[-]))\otimes\mathsf{id}"] \arrow[dd, "\mathsf{meta}\otimes\mathsf{id}"'] & & x\otimes((\$x \leftarrow\!\otimes \$x)\otimes\$x) \arrow[d, "\mathsf{id}\otimes\mathsf{eval}"] \\
& \overset{\substack{\text{def meta}\\\text{case}}}{=} & x\otimes\$x \arrow[d, "\mathsf{meta}"] \\
\$x\otimes\$x \arrow[rr, "{-[-]}"'] & & \$x
\end{tikzcd}$$

Lemma E.2 *Simultaneous substitution is the unique morphism satisfying:*

$$\begin{tikzcd}
& \underline{\mathbf{O}}(\$x \otimes \$x) \arrow[drr, "\underline{\mathbf{O}}(-[-])"] & & \\
(\underline{\mathbf{O}}\$x)\otimes_{\bullet}\ \mathsf{var} \arrow[ur, "\mathsf{str}"] \arrow[dr, "{[\![-]\!] \otimes \mathsf{id}}"'] & \overset{\substack{\text{op}\\ \text{case}}}{=} & & \underline{\mathbf{O}}\$x \arrow[dl, "{[\![-]\!]}"] \\
& \$x \otimes \$x \arrow[r, "-[-]"'] & \$x &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathbb{I} \otimes \$x \arrow[dl, "\mathsf{var} \otimes \mathsf{id}"'] \arrow[dr, "\ell"] & \\
\$x \otimes \$x \arrow[rr, "-[-]"'] & \overset{\substack{\text{var}\\ \text{case}}}{=} & \$x
\end{tikzcd}
\qquad
\begin{tikzcd}
& x \otimes (\$x \otimes \$x) \arrow[rr, "\mathsf{id} \otimes (-[-])"] & & x \otimes \$x \arrow[dr, "\mathsf{meta}"] & \\
(x \otimes \$x) \otimes \$x \arrow[ur, "\mathbf{a}"] \arrow[drr, "\mathsf{meta} \otimes \mathsf{id}"'] & & \overset{\substack{\text{metavariable}\\ \text{case}}}{=} & & \$x \\
& & \$x \otimes \$x \arrow[urr, "-[-]"'] & &
\end{tikzcd}$$

Proof. The equation ‘var case’ is already of the required form ‘var case’. For the other two equations, calculate:

$$\begin{tikzcd}
& (\mathbf{O}(\$x \mathrel{\leftarrow\!\otimes} \$x)) \otimes \$x \arrow[drr, "\mathsf{str}^{\mathbf{O}}_{\$x \leftarrow\otimes \$x, \mathsf{var}}"] & & \\
(\mathbf{O}\$x) \otimes \$x \arrow[ur, "\mathbf{O}(\mathsf{curry}(-[-])) \otimes \mathsf{id}"] \arrow[dr, "\mathsf{str}^{\mathbf{O}}_{\$x, \mathsf{var}}"'] \arrow[ddd, "{[\![-]\!] \otimes \mathsf{id}}"'] & \overset{\text{str-nat}}{=} & & \mathbf{O}\,((\$x \mathrel{\leftarrow\!\otimes} \$x) \otimes \$x) \arrow[dd, bend left, "\mathbf{O}\mathsf{eval}"] \\
& \mathbf{O}(\$x \otimes \$x) \arrow[urr, "\mathbf{O}(\mathsf{curry}(-[-]) \otimes \mathsf{id})"'] \arrow[drr, bend right, "\mathbf{O}(-[-])"'] & \overset{\substack{\text{right}\\ \text{exponential}}}{=} & \\
& & & \mathbf{O}(\$x) \arrow[d, "{[\![-]\!]}"] \\
(\mathbf{O}\$x) \otimes \$x \arrow[rrr, "-[-]"'] & & & \$x
\end{tikzcd}$$

$$\begin{tikzcd}
& (x \otimes (\$x \mathrel{\leftarrow\!\otimes} \$x)) \otimes \$x \arrow[drr, "\mathbf{a}"] & & \\
(x \otimes \$x) \otimes \$x \arrow[ur, "(\mathsf{id} \otimes \mathsf{curry}(-[-])) \otimes \mathsf{id}"] \arrow[dr, "\mathbf{a}"'] \arrow[ddd, "\mathsf{meta} \otimes \mathsf{id}"'] & \overset{\mathbf{a}\text{-nat}}{=} & & x \otimes ((\$x \mathrel{\leftarrow\!\otimes} \$x) \otimes \$x) \arrow[dd, bend left, "\mathsf{id} \otimes \mathsf{eval}"] \\
& x \otimes (\$x \otimes \$x) \arrow[urr, "\mathsf{id} \otimes (\mathsf{curry}(-[-]) \otimes \mathsf{id})"'] \arrow[drr, bend right, "\mathsf{id} \otimes -[-]"'] & \overset{\substack{\text{right}\\ \text{exponential}}}{=} & \\
& & & x \otimes \$x \arrow[d, "\mathsf{meta}"] \\
\$x \otimes \$x \arrow[rrr, "-[-]"'] & & & \$x
\end{tikzcd}$$

and so the inner face commutes iff the outer face commutes. □

Let $\mathbf{M}_{\mathbf{O}}x \coloneqq \big(\$^{\mathbf{O}}x, -[-], \mathsf{var}\big)$. The next step in the proof is to show that $\mathbf{M}x$ is a substitution monoid. The equation ‘var case’ is the left neutrality equation. Each lemma proves the each remaining equations.

Lemma E.3 *The monoid structure* $\mathbf{M}_{\mathbf{O}}x$ *satisfies the right neutrality equation:*

$$\begin{tikzcd}
\underline{\mathbf{M}} \otimes \mathbb{I} \arrow[r, "\mathsf{id} \otimes \mathsf{var}"] & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[d, "-[-]"] \\
\underline{\mathbf{M}} \arrow[u, "\mathbf{r}'"] \arrow[r, equal] \arrow[ur, phantom, "\overset{\substack{\text{right}\\ \text{unit}}}{=}"] & \underline{\mathbf{M}}
\end{tikzcd}$$

Proof. Appeal to the indexed, but otherwise ordinary, induction principle. Calculate:

$$\begin{tikzcd}
\mathbf{O}\$x \arrow[r, "\mathbf{Or}'"] \arrow[dd, "{[\![-]\!]}"'] \arrow[dr, "\mathbf{r}'"'] & \mathbf{O}(\$x \otimes \mathbb{I}) \arrow[rr, "\mathbf{O}(\mathsf{id} \otimes \mathsf{var})"] & & \mathbf{O}(\$x \otimes \$x) \arrow[rr, "\mathbf{O}(-[-])"] & & \mathbf{O}\$x \arrow[dd, "{[\![-]\!]}"] \\
\overset{\mathbf{r}'\text{-nat.}}{=} & \mathbf{O}(\$x) \otimes \mathbb{I} \arrow[u, "\mathsf{str}^{\mathbf{O}}_{\$x,\mathbb{I}}"'] \arrow[rr, "\mathsf{id} \otimes \mathsf{id}"'] \arrow[d, "{[\![-]\!] \otimes \mathsf{id}}"'] & \overset{\text{strength nat.}}{=} & \mathbf{O}(\$x) \otimes \$x \arrow[u, "\mathsf{str}^{\mathbf{O}}_{\$x,\mathsf{var}}"'] \arrow[d, "{[\![-]\!] \otimes \mathsf{id}}"] & \overset{\text{op case}}{=} & \\
\$x \arrow[r, "\mathbf{r}'"'] & \$x \otimes \mathbb{I} \arrow[rr, "\mathsf{id} \otimes \mathsf{var}"'] & \overset{\text{functoriality}}{=} & \$x \otimes \$x \arrow[rr, "-[-]"'] & & \$x
\end{tikzcd}$$

strength triangle =

$$\begin{tikzcd}
\mathbb{I} \arrow[rrrr, equal] \arrow[dddd, "\mathsf{var}"'] \arrow[drr, "\mathbf{r}'"'] & & \overset{\text{skew triangle}}{=} & & \mathbb{I} \arrow[dddd, "\mathsf{var}"] \\
& & \mathbb{I} \otimes \mathbb{I} \arrow[urr, "\ell"'] \arrow[ddl, "\mathsf{var} \otimes \mathsf{id}"'] \arrow[dr, "\mathsf{id} \otimes \mathsf{var}"] & & \overset{\ell\text{-nat}}{=} \\
\overset{\mathbf{r}'\text{-nat}}{=} & & & \mathbb{I} \otimes \$x \arrow[d, "\mathsf{var} \otimes \mathsf{id}"'] \arrow[dr, "\ell"] & \\
& & \overset{\text{functoriality}}{=} & \overset{\text{var case}}{=} & \\
\$x \arrow[r, "\mathbf{r}'"'] & \$x \otimes \mathbb{I} \arrow[rr, "\mathsf{id} \otimes \mathsf{var}"'] & & \$x \otimes \$x \arrow[r, "-[-]"'] & \$x
\end{tikzcd}$$

$$\begin{tikzcd}
x \otimes \$x \arrow[r, "\mathsf{id} \otimes \mathbf{r}'"] \arrow[dd, "\mathsf{meta}"'] \arrow[dr, "\mathbf{r}'"'] & x \otimes (\$x \otimes \mathbb{I}) \arrow[rr, "\mathsf{id} \otimes (\mathsf{id} \otimes \mathsf{var})"] & & x \otimes (\$x \otimes \$x) \arrow[rr, "\mathsf{id} \otimes (-[-])"] & & x \otimes \$x \arrow[dd, "\mathsf{meta}"] \\
\overset{\mathbf{r}'\text{ nat.}}{=} & (x \otimes \$x) \otimes \mathbb{I} \arrow[u, "\mathbf{a}"'] \arrow[rr, "(\mathsf{id} \otimes \mathsf{id}) \otimes \mathsf{var}"'] \arrow[d, "\mathsf{meta} \otimes \mathsf{id}"] & \overset{\mathbf{a}\text{ nat}}{=} & (x \otimes \$x) \otimes \$x \arrow[u, "\mathbf{a}"'] \arrow[d, "\mathsf{meta} \otimes \mathsf{id}"] & \overset{\text{metavariable case}}{=} & \\
\$x \arrow[r, "\mathbf{r}'"'] & \$x \otimes \mathbb{I} \arrow[rr, "\mathsf{id} \otimes \mathsf{var}"'] & \overset{\text{functoriality}}{=} & \$x \otimes \$x \arrow[rr, "-[-]"'] & & \$x
\end{tikzcd}$$

skew right =

Therefore, by induction, the right unit equation holds. □

We use the following fact to prove associativity:

Lemma E.4 *Simultaneous substitution preserves points, i.e.,* $-[-] : \mathsf{var} \to \mathsf{var} \otimes^{\bullet} \mathsf{var}$ *in* $\mathcal{C}^{\bullet}$.

Proof. Calculate:

$$\begin{tikzcd}
\mathbb{I} \arrow[rrrr, equal] \arrow[dd, "\mathsf{env}^{\mathsf{var} \otimes^{\bullet} \mathsf{var}}"'] \arrow[dr, "\mathbf{r}'"] & & \overset{\mathbf{r}'\text{ nat}}{=} & & \mathbb{I} \arrow[dl, "\mathsf{var}"'] \arrow[dd, "\mathsf{var}"] \\
{:=} & \mathbb{I} \otimes \mathbb{I} \arrow[r, "\mathsf{var} \otimes \mathsf{id}"] \arrow[dl, "\mathsf{var} \otimes \mathsf{var}"] & \$x \otimes \mathbb{I} \arrow[d, "\mathsf{id} \otimes \mathsf{var}"] & \$x \arrow[l, "\mathbf{r}'"'] \arrow[dr, equal] & \overset{\mathsf{id}}{=} \\
\$x \otimes \$x \arrow[rr, equal] & \overset{\text{functoriality}}{=} & \$x \otimes \$x \arrow[rr, "-[-]"'] & \overset{\text{monoid right unit}}{=} & \$x
\end{tikzcd}$$

□

Lemma E.5 *The monoid structure* $\mathbf{M_O}x$ *satisfies the associativity equation:*

$$\begin{tikzcd}
 & & \underline{\mathbf{M}}\otimes\underline{\mathbf{M}} \arrow[drr, bend left, "-[-]"] & & \\
(\underline{\mathbf{M}}\otimes\underline{\mathbf{M}})\otimes\underline{\mathbf{M}} \arrow[urr, bend left, "(-[-])\otimes\mathsf{id}"] \arrow[dr, "\mathbf{a}"'] & & \overset{\text{associativity}}{=} & & \underline{\mathbf{M}} \\
 & \underline{\mathbf{M}}\otimes(\underline{\mathbf{M}}\otimes\underline{\mathbf{M}}) \arrow[rr, bend right, "\mathsf{id}\otimes(-[-])"'] & & \underline{\mathbf{M}}\otimes\underline{\mathbf{M}} \arrow[ur, "-[-]"'] &
\end{tikzcd}$$

Proof. We use the associativity assumption and invert **a**, and so we need to show:

$$\begin{tikzcd}
 & & \mathbb{S}x\otimes\mathbb{S}x \arrow[drr, bend left, "-[-]"] & & \\
(\mathbb{S}x\otimes\mathbb{S}x)\otimes\mathbb{S}x \arrow[urr, bend left, "(-[-])\otimes\mathsf{id}"] & & \overset{\text{associativity}}{=} & & \mathbb{S}x \\
 & \mathbb{S}x\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[ul, "\mathbf{a}^{-1}"] \arrow[rr, bend right, "\mathsf{id}\otimes(-[-])"'] & & \mathbb{S}x\otimes\mathbb{S}x \arrow[ur, "-[-]"'] &
\end{tikzcd}$$

We employ the indexed parameterised induction principle from Lemma E.1. To do so, let the two sides of the pentagon be: $\mathbb{S}x\otimes(\mathbb{S}x\otimes\mathbb{S}x) \overset{g_2}{\underset{g_1}{\rightrightarrows}} \mathbb{S}x$. We will define morphisms:

$$\mathbf{O}(\mathbb{S}x \leftarrow_{\otimes} (\mathbb{S}x\otimes\mathbb{S}x))\otimes(\mathbb{S}x\otimes\mathbb{S}x) \xrightarrow{f_{\mathrm{op}}} \mathbb{S}x \qquad \mathbb{I}\otimes(\mathbb{S}x\otimes\mathbb{S}x) \xrightarrow{f_{\mathsf{var}}} \mathbb{S}x \qquad (x\otimes(\mathbb{S}x \leftarrow_{\otimes} (\mathbb{S}x\otimes\mathbb{S}x)))\otimes(\mathbb{S}x\otimes\mathbb{S}x) \xrightarrow{f_{\mathrm{meta}}} \mathbb{S}x$$

and show that both g_i satisfy the indexed parameterised induction cases, and therefore $g_1 = \mathsf{fold} f = g_2$.

Op case:

A straightforward, though somewhat length, calculation, in Fig E.1.

Var case:

$$\begin{tikzcd}
\mathbb{I}\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[rr, equal] \arrow[ddd, "\mathsf{var}\otimes\mathsf{id}"'] & & \mathbb{I}\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[d, "\mathsf{var}\otimes\mathsf{id}"] \\
 & & \mathbb{S}x\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[d, "\mathsf{id}\otimes(-[-])"] \\
 & \overset{\mathsf{id}}{=} & \mathbb{S}x\otimes\mathbb{S}x \arrow[d, "-[-]"] \\
\mathbb{S}x\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[r, "\mathsf{id}\otimes(-[-])"'] & \mathbb{S}x\otimes\mathbb{S}x \arrow[r, "-[-]"'] & \mathbb{S}x
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbb{I}\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[rrrrr, equal] \arrow[ddd, "\mathsf{var}\otimes\mathsf{id}"'] \arrow[ddr, "\mathbf{a}^{-1}"] \arrow[dddrr, bend left, "\ell"] & & & & & \mathbb{I}\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[dll, "\mathsf{id}\otimes(-[-])"'] \arrow[d, "\mathsf{var}\otimes\mathsf{id}"] \\
 & \overset{\text{skew left}}{=} & & \mathbb{I}\otimes\mathbb{S}x \arrow[drr, bend right, "\mathsf{var}\otimes\mathsf{id}"'] \arrow[ddrr, bend right, "\ell"'] & \overset{\text{functoriality}}{=} & \mathbb{S}x\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[d, "\mathsf{id}\otimes(-[-])"] \\
\overset{\mathbf{a}^{-1}\text{-nat}}{=} & (\mathbb{I}\otimes\mathbb{S}x)\otimes\mathbb{S}x \arrow[d, "(\mathsf{var}\otimes\mathsf{id})\otimes\mathsf{id}"'] \arrow[dr, bend left, "\ell\otimes\mathsf{id}"] & \overset{\ell\text{-nat}}{=} & \overset{\ell\text{-nat}}{=} & \overset{\text{var case}}{=} & \mathbb{S}x\otimes\mathbb{S}x \arrow[d, "-[-]"] \\
\mathbb{S}x\otimes(\mathbb{S}x\otimes\mathbb{S}x) \arrow[r, "\mathbf{a}^{-1}"'] & (\mathbb{S}x\otimes\mathbb{S}x)\otimes\mathbb{S}x \arrow[r, "(-[-])\otimes\mathsf{id}"'] & \mathbb{S}x\otimes\mathbb{S}x \arrow[rrr, "-[-]"'] & & & \mathbb{S}x
\end{tikzcd}$$

$$\begin{tikzcd}
(\mathbf{O}(\$x)) \otimes (\$x \otimes \$x) \arrow[rrrr, "\mathbf{O}(\mathsf{curry} g_1) \otimes \mathsf{id}"] \arrow[drr, "\mathsf{str}\ \$x,\, \mathsf{var} \otimes^{\bullet} \mathsf{var}"] \arrow[ddr, "\mathsf{id} \otimes (-[-])"'] \arrow[ddd, bend right, "{[\![-]\!]} \otimes \mathsf{id}"'] & & & & (\mathbf{O}(\$x \leftarrow\!\!\otimes (\$x \otimes \$x))) \otimes (\$x \otimes \$x) \arrow[d, "\mathsf{str}^{\mathbf{O}}_{\$x,\mathsf{var} \otimes^{\bullet} \mathsf{var}}"] \arrow[dll, phantom, "\overset{\text{str-nat}}{="] \\
 & & \mathbf{O}(\$x \otimes (\$x \otimes \$x)) \arrow[rr, "\mathbf{O}(\mathsf{curry} g_1 \otimes \mathsf{id})"'] \arrow[dr, "\mathbf{O}(\mathsf{id} \otimes (-[-]))"'] \arrow[dl, phantom, "\overset{\substack{\text{str-nat} \\ -[-]\,:\,\mathsf{var} \to \mathsf{var} \otimes^{\bullet} \mathsf{var}}}{="] & & \mathbf{O}(\$x \leftarrow\!\!\otimes (\$X \otimes \$x)) \arrow[d, "\mathbf{O}\mathsf{eval}"] \arrow[dl, phantom, "\overset{\text{exponential}}{="] \\
 & (\mathbf{O}\$x) \otimes \$x \arrow[rr, "\mathsf{str}_{\$x,\mathsf{var}}"'] \arrow[dr, "{[\![-]\!]} \otimes \mathsf{id}"'] \arrow[dl, phantom, "\overset{\text{functoriality}}{="] & & \mathbf{O}(\$x \otimes \$x) \arrow[r, "\mathbf{O}(-[-])"'] \arrow[dr, phantom, "\overset{\text{op case}}{="] & \mathbf{O}(\$x) \arrow[d, "{[\![-]\!]}"] \\
\$x \otimes (\$x \otimes \$x) \arrow[rr, "\mathsf{id} \otimes (-[-])"'] & & \$x \otimes \$x \arrow[rr, "-[-]"'] & & \$x
\end{tikzcd}$$

$$\begin{tikzcd}
(F_i(\$x)) \otimes (\$x \otimes \$x) \arrow[rrrr, "\mathbf{O}(\mathsf{curry} g_2) \otimes \mathsf{id}"] \arrow[dr, "\mathsf{str}_{\$x,\mathsf{var} \otimes^{\bullet} \mathsf{var}}"] \arrow[ddddr, bend right, "\mathbf{a}^{-1}"] \arrow[ddddd, "{[\![-]\!]} \otimes \mathsf{id}"'] \arrow[dr, phantom, shift right=6, "\overset{\substack{\text{strength} \\ \text{pentagon}}}{="'] & & & & (F_i(\$x \leftarrow\!\!\otimes (\$x \otimes \$x))) \otimes (\$x \otimes \$x) \arrow[d, "\mathsf{str}^{\mathbf{O}}_{\$x,\mathsf{var} \otimes^{\bullet} \mathsf{var}}"] \arrow[dlll, phantom, "\overset{\text{str-nat.}}{="] \\
 & \mathbf{O}(\$x \otimes (\$x \otimes \$x)) \arrow[rrr, "\mathbf{O}(\mathsf{curry} g_1 \otimes \mathsf{id})"'] \arrow[d, "\mathbf{O}\mathbf{a}^{-1}"'] \arrow[drr, phantom, "\overset{\text{exponential}}{="] & & & \mathbf{O}(\$x \leftarrow\!\!\otimes (\$X \otimes \$x)) \arrow[d, "\mathbf{O}\mathsf{eval}"] \\
 & \mathbf{O}((\$x \otimes \$x) \otimes \$x) \arrow[r, "\mathbf{O}(-[-] \otimes \mathsf{id})"] & \mathbf{O}(\$x \otimes \$x) \arrow[rr, "\mathbf{O}(-[-])"] \arrow[d, phantom, "\overset{\text{str-nat}}{="] & & \mathbf{O}(\$x) \arrow[ddd, "{[\![-]\!]}"] \\
 & & (\mathbf{O}(\$x \otimes \$x)) \otimes \$x \arrow[ul, "\mathsf{str}_{\$x \otimes \$x,\mathsf{var}} \otimes \mathsf{id}"'] \arrow[r, "(\mathbf{O}\,{[\![-]\!]}) \otimes \mathsf{id}"'] \arrow[dd, phantom, "\overset{\text{op case}}{="] & (\mathbf{O}\$x) \otimes \$x \arrow[ul, "\mathsf{str}_{\$x,\mathsf{var}}"'] \arrow[dd, "{[\![-]\!]} \otimes \mathsf{id}"] \arrow[ddr, phantom, "\overset{\text{op case}}{="] & \\
 & ((\mathbf{O}\$x) \otimes \$x) \otimes \$x \arrow[ur, "\mathsf{str}_{\$x,\mathsf{var}} \otimes \mathsf{id}"'] \arrow[d, "({[\![-]\!]} \otimes \mathsf{id}) \otimes \mathsf{id}"] \arrow[dl, phantom, "\overset{\mathbf{a}^{-1}\text{-nat}}{="] & & & \\
\$x \otimes (\$x \otimes \$x) \arrow[r, "\mathbf{a}^{-1}"'] & (\$x \otimes \$x) \otimes \$x \arrow[rr, "(-[-]) \otimes \mathsf{id}"'] & & \$x \otimes \$x \arrow[r, "-[-]"'] & \$x
\end{tikzcd}$$

Fig. E.1. Inductive case for operations in the proof of Lemma E.5.

Metavariable case:

$$\begin{tikzcd}
(x \otimes \$x) \otimes (\$x \otimes \$x) \arrow[rrrr, "(\mathsf{id} \otimes \mathsf{curry} g_1) \otimes \mathsf{id}"] \arrow[drrr, "\mathbf{a}"] \arrow[dddr, bend right, "\mathsf{id} \otimes (-[-])"] \arrow[dddd, "\mathsf{meta} \otimes \mathsf{id}"'] & & & & (x \otimes (\$x \leftarrow\!\!\otimes (\$x \otimes \$x))) \otimes (\$x \otimes \$x) \arrow[d, "\mathbf{a}"] \arrow[dl, phantom, "\overset{\mathbf{a}\text{-nat}}{="] \\
 & & & x \otimes (\$x \otimes (\$x \otimes \$x)) \arrow[r, "\mathsf{id} \otimes (\mathsf{curry} g_1 \otimes \mathsf{id})"'] \arrow[dl, "\mathsf{id} \otimes (\mathsf{id} \otimes (-[-]))"'] \arrow[d, phantom, "\overset{\text{exponential}}{="] & x \otimes ((\$x \leftarrow\!\!\otimes (\$x \otimes \$x)) \otimes (\$x \otimes \$x)) \arrow[d, "\mathsf{id} \otimes \mathsf{eval}"] \\
 & \overset{\mathbf{a}\text{-nat}}{=} & x \otimes (\$x \otimes \$x \arrow[rr, "\mathsf{id} \otimes (-[-])"'] & & x \otimes \$x \arrow[dd, "\mathsf{meta}"] \\
 & (x \otimes \$x) \otimes \$x \arrow[ur, "\mathbf{a}"'] \arrow[drr, "\mathsf{meta} \otimes \mathsf{id}"] \arrow[dl, phantom, "\overset{\text{functoriality}}{="] & & \overset{\substack{\text{metavariable} \\ \text{case}}}{=} & \\
\$x \otimes (\$x \otimes \$x) \arrow[rrr, "\mathsf{id} \otimes (-[-])"'] & & & \$x \otimes \$x \arrow[r, "-[-]"'] & \$x
\end{tikzcd}$$

$$\begin{tikzcd}
(x\otimes\$x)\otimes(\$x\otimes\$x) \arrow[rrrrr, "(\mathsf{id}\otimes\mathsf{curry}g_2)\otimes\mathsf{id}"] \arrow[drr, "\mathbf{a}"] \arrow[ddddr, "\mathbf{a}^{-1}"] \arrow[ddddd, "\mathsf{meta}\otimes\mathsf{id}"'] \arrow[from=1-1, to=2-6, phantom, "\overset{\mathbf{a}\text{-nat}}{=}"] \arrow[from=1-1, to=6-2, phantom, "\overset{\mathbf{a}^{-1}\text{-nat}}{=}"] & & & & & (x\otimes(\$x \mathrel{\leftarrow\!\otimes} (\$x\otimes\$x)))\otimes(\$x\otimes\$x) \arrow[d, "\mathbf{a}"] \\
 & & x\otimes(\$x\otimes(\$x\otimes\$x)) \arrow[rrr, "\mathsf{id}\otimes(\mathsf{curry}g_2\otimes\mathsf{id})"] \arrow[d, "\mathsf{id}\otimes\mathbf{a}^{-1}"'] \arrow[from=2-3, to=5-2, phantom, "\overset{\text{skew pentagon}}{=}"] & & & x\otimes((\$x \mathrel{\leftarrow\!\otimes} (\$x\otimes\$x))\otimes(\$x\otimes\$x)) \arrow[d, "\mathsf{id}\otimes\mathsf{eval}"] \arrow[from=2-6, to=3-4, phantom, "\overset{\text{exponential}}{=}"] \\
 & & x\otimes((\$x\otimes\$x)\otimes\$x) \arrow[r, "\mathsf{id}\otimes((-[-])\otimes\mathsf{id})"] \arrow[from=3-3, to=5-5, phantom, "\overset{\mathbf{a}\text{-nat}}{=}"] & x\otimes(\$x\otimes\$x) \arrow[rr, "\mathsf{id}\otimes(-[-])"] & & x\otimes\$x \arrow[ddd, "\mathsf{meta}"] \arrow[from=3-6, to=6-5, phantom, "\overset{\text{metavriable case}}{=}"] \\
 & & (x\otimes(\$x\otimes\$x))\otimes\$x \arrow[u, "\mathbf{a}"] \arrow[drr, "(\mathsf{id}\otimes(-[-]))\otimes\mathsf{id}"] & & & \\
 & ((x\otimes\$x)\otimes\$x)\otimes\$x \arrow[ur, "\mathbf{a}\otimes\mathsf{id}"] \arrow[dr, "(\mathsf{meta}\otimes\mathsf{id})\otimes\mathsf{id}"] \arrow[from=5-2, to=6-5, phantom, "\overset{\text{metavariable case}}{=}"] & & & (x\otimes\$x)\otimes\$x \arrow[uul, "\mathbf{a}"] \arrow[d, "-[-]"] & \\
\$x\otimes(\$x\otimes\$x) \arrow[rr, "\mathbf{a}^{-1}"'] & & (\$x\otimes\$x)\otimes\$x \arrow[rr, "(-[-])\otimes\mathsf{id}"'] & & \$x\otimes\$x \arrow[r, "-[-]"'] & \$x
\end{tikzcd}$$

completing the proof. □

To show $\mathbf{F_O}x$ is initial over x, define the unit for this adjunction by:

$$\mathsf{menv} : x \xrightarrow{\mathbf{r}'} x\otimes\mathbb{I} \xrightarrow{\mathsf{id}\otimes\mathsf{var}} x\otimes\$^{\mathbf{O}}x \xrightarrow{\mathsf{meta}} \$^{\mathbf{O}}x$$

Lemma E.6 *We can recover* meta *from* menv *and substitution:*

$$\begin{tikzcd}
x\otimes\$x \arrow[rrd, "\mathsf{menv}\otimes\mathsf{id}"] \arrow[dd, "\mathsf{meta}"'] & & \\
 & \overset{\mathsf{meta\ by\ menv}}{=} & \$x\otimes\$x \arrow[lld, "-[-]"] \\
\$x & &
\end{tikzcd}$$

Proof. Calculate:

$$\begin{tikzcd}
x\otimes\$x \arrow[rrr, "\mathsf{menv}\otimes\mathsf{id}"] \arrow[dr, "\mathbf{r}'\otimes\mathsf{id}"] \arrow[ddd, equal, bend right] \arrow[from=1-1, to=3-2, phantom, "\overset{\text{skew rectangle}}{=}"] \arrow[from=1-2, to=2-4, phantom, ":="] & & & \$x\otimes\$x \arrow[ddd, "-[-]"] \\
 & (x\otimes\mathbb{I})\otimes\$x \arrow[r, "(\mathsf{id}\otimes\mathsf{var})\otimes\mathsf{id}"'] \arrow[d, "\mathbf{a}"] \arrow[dr, phantom, "\overset{\mathbf{a}\text{-nat}}{=}"] & (x\otimes\$x)\otimes\$x \arrow[ur, "\mathsf{meta}\otimes\mathsf{id}"] \arrow[d, "\mathbf{a}"'] & \\
 & x\otimes(\mathbb{I}\otimes\$x) \arrow[r, "\mathsf{id}\otimes(\mathsf{var}\otimes\mathsf{id})"] \arrow[dl, "\mathsf{id}\otimes\ell"'] \arrow[from=3-2, to=4-2, phantom, "\overset{\text{left unit}}{=}"] & x\otimes(\$x\otimes\$x) \arrow[dll, bend left, "\mathsf{id}\otimes(-[-])"] \arrow[ur, phantom, "\overset{\text{metavariable case}}{=}"] & \\
x\otimes\$x \arrow[rrr, bend right, "\mathsf{meta}"'] & & & \$x
\end{tikzcd}$$

as we wanted. □

Let $(\mathbf{M}, \mathsf{menv} : x \to \underline{\mathbf{M}})$ be any $\mathbf{O}$-monoid over x. Define:

$$\mathsf{meta}^{\mathbf{M}} : x\otimes\underline{\mathbf{M}} \xrightarrow{\mathsf{menv}\otimes\mathsf{id}} \underline{\mathbf{M}}\otimes\underline{\mathbf{M}} \xrightarrow{-[-]_{\mathbf{M}}} \underline{\mathbf{M}}$$

Lemma E.7 *We can recover* $\mathsf{menv}^{\mathbf{M}}$ *from* $\mathsf{meta}^{\mathbf{M}}$ *and* meta *satisfies an analogue to metavariable case:*

$$\begin{tikzcd}
x \arrow[r, "\mathbf{r}'"] \arrow[d, "\mathsf{menv}^{\mathbf{M}}"'] \arrow[dr, phantom, "\overset{\mathsf{menv}\text{ by }\mathsf{meta}}{=}"] & x \otimes \mathbb{I} \arrow[d, "\mathrm{id} \otimes \mathsf{env}"] \\
\underline{\mathbf{M}} & x \otimes \underline{\mathbf{M}} \arrow[l, "\mathsf{meta}^{\mathbf{M}}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
(x \otimes \underline{\mathbf{M}}) \otimes \underline{\mathbf{M}} \arrow[r, "\mathbf{a}"] \arrow[dd, "\mathsf{meta} \otimes \mathrm{id}"'] \arrow[ddr, phantom, "\overset{\text{metavariable case}}{=}"] & x \otimes (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[d, "\mathrm{id} \otimes (-[-])"] \\
 & x \otimes \underline{\mathbf{M}} \arrow[d, "\mathsf{meta}"] \\
\underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[r, "-[-]"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

Proof. Straightforward calculations:

$$\begin{tikzcd}
x \arrow[rrr, "\mathbf{r}'"] \arrow[dr, "\mathsf{menv}"] \arrow[dddd, "\mathsf{menv}"'] & & & x \otimes \mathbb{I} \arrow[dl, "\mathsf{menv} \otimes \mathrm{id}"'] \arrow[dddd, "\mathrm{id} \otimes \mathsf{env}"] \\
 & \arrow[r, phantom, "\overset{\mathbf{r}'\text{-nat}}{=}"] & \underline{\mathbf{M}} \otimes \mathbb{I} \arrow[dd, "\mathrm{id} \otimes \mathsf{env}"'] & \\
\arrow[r, phantom, "="] & \underline{\mathbf{M}} \arrow[ur, "\mathbf{r}'"] \arrow[ddl, equal] & \overset{\text{right unit}}{=} & \overset{\text{functoriality}}{=} \\
 & & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[dll, "-[-]_{\mathbf{M}}"'] & \\
\underline{\mathbf{M}} & & := & x \otimes \underline{\mathbf{M}} \arrow[ul, "\mathsf{menv} \otimes \mathrm{id}"'] \arrow[lll, bend left, "\mathsf{meta}"]
\end{tikzcd}$$

$$\begin{tikzcd}
(x \otimes \underline{\mathbf{M}}) \otimes \underline{\mathbf{M}} \arrow[rrr, "\mathbf{a}"] \arrow[dr, "(\mathsf{menv} \otimes \mathrm{id}) \otimes \mathrm{id}"] \arrow[dddd, "\mathsf{meta} \otimes \mathrm{id}"'] & & & x \otimes (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[dl, "\mathsf{menv} \otimes (\mathrm{id} \otimes \mathrm{id})"'] \arrow[dd, "\mathrm{id} \otimes (-[-])"] \\
 & (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \otimes \underline{\mathbf{M}} \arrow[r, "\mathbf{a}"] \arrow[dddl, "(-[-]) \otimes \mathrm{id}"'] & \underline{\mathbf{M}} \otimes (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[d, "\mathrm{id} \otimes (-[-])"'] & \overset{\text{functoriality}}{=} \\
:= & \overset{\mathbf{a}\text{-nat}}{=} & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[ddr, "-[-]"'] & x \otimes \underline{\mathbf{M}} \arrow[l, "\mathsf{menv} \otimes \mathrm{id}"'] \arrow[dd, "\mathsf{meta}"] \\
 & & \overset{\text{associativity}}{=} & =: \\
\underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[rrr, "-[-]"'] & & & \underline{\mathbf{M}}
\end{tikzcd}$$

as we wanted. □

Let $h : \$x \to \underline{\mathbf{M}}$ be the unique arrow satisfying:

$$\begin{tikzcd}
\mathbf{O}\$x \arrow[r, "\mathbf{O}h"] \arrow[d, "\$x\,\llbracket - \rrbracket"'] \arrow[dr, phantom, "\overset{\text{op preservation}}{=}"] & \mathbf{O}\underline{\mathbf{M}} \arrow[d, "\mathbf{M}\,\llbracket - \rrbracket"] \\
\$x \arrow[r, "h"'] & \underline{\mathbf{M}}
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathbb{I} \arrow[dl, "\mathsf{var}"'] \arrow[dr, "\mathsf{env}"] \arrow[d, phantom, "\overset{\text{var preservation}}{=}"] & \\
\$x \arrow[rr, "h"'] & {} & \underline{\mathbf{M}}
\end{tikzcd}
\qquad
\begin{tikzcd}
x \otimes \$x \arrow[r, "\mathrm{id} \otimes h"] \arrow[d, "\mathsf{meta}^{\mathbf{M}}"'] \arrow[dr, phantom, "\overset{\text{metavariable preservation}}{=}"] & x \otimes \underline{\mathbf{M}} \arrow[d, "\mathsf{meta}^{\$x}"] \\
\$x \arrow[r, "h"'] & \underline{\mathbf{M}}
\end{tikzcd}
\qquad \text{(E.1)}$$

Lemma E.8 *If* $k : \mathbf{F_O}x \to \mathbf{M}$ *is an* $\mathbf{O}$*-monoid homomorphism extending* $\mathsf{menv}^{\mathbf{M}}$ *along* $\mathsf{menv}^{\$x}$*, then* $k = h$*.*

Proof. Since k is a $\mathbf{O}$-homomorphism, the equation 'op preservation' holds. Since k is a monoid-homomorphism, the equation 'var preservation' holds. For the equation 'metavariable preservation':

$$\begin{tikzcd}
x \otimes \$x \arrow[rrrrr, "\mathsf{id} \otimes k"] \arrow[ddd, "\mathsf{meta}"'] \arrow[dd, phantom, shift left=8ex, "\substack{\mathsf{meta}\text{ by}\\ \mathsf{menv}\\ =}"] \arrow[ddrr, "\mathsf{menv}^{\$x} \otimes \mathsf{id}"] & & & & & x \otimes \underline{\mathbf{M}} \arrow[ddd, "\mathsf{meta}^{\mathbf{M}}"] \arrow[ddll, "\mathsf{menv}^{\mathbf{M}} \otimes \mathsf{id}"'] \\
& & \arrow[r, phantom, "\substack{k\text{ preserves }\mathsf{menv}\\ =}"] & {} & & \\
& & \$x \otimes \$x \arrow[r, "k \otimes k"] \arrow[dll, "-[-]_{\mathbf{F}_{\mathbf{O}}x}"'] \arrow[d, phantom, shift left=6ex, "\substack{k\text{ monoid homo.}\\ =}"] & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[drr, "-[-]_{\mathbf{M}}"'] & =: & \\
\$x \arrow[rrrrr, "k"'] & & {} & & & \underline{\mathbf{M}}
\end{tikzcd}$$

Thus by definition $k = h$. □

Lemma E.9 *The arrow* h *extends* $\mathsf{menv}^{\mathbf{M}}$ *along* $\mathsf{menv}^{\$x}$*:*

$$\begin{tikzcd}
& x \arrow[dl, "\mathsf{menv}^{\$x}"'] \arrow[dr, "\mathsf{menv}^{\mathbf{M}}"] \arrow[d, phantom, "\substack{\mathsf{menv}\\ \text{preservation}\\ =}"] & \\
\$x \arrow[rr, "h"'] & {} & \underline{\mathbf{M}}
\end{tikzcd}$$

Proof. Calculate:

$$\begin{tikzcd}
& & x \arrow[d, "\mathbf{r}'"] \arrow[ddll, bend right, "\mathsf{menv}^{\$x}"'] \arrow[ddrr, bend left, "\mathsf{menv}^{\mathbf{M}}"] \arrow[d, phantom, shift right=6ex, ":="] \arrow[d, phantom, shift left=7ex, "\substack{\mathsf{menv}\\ \text{by }\mathsf{meta}\\ =}"] & & \\
& & x \otimes \mathbb{I} \arrow[dl, "\mathsf{id} \otimes \mathsf{var}"'] \arrow[dr, "\mathsf{id} \otimes \mathsf{env}"] \arrow[d, phantom, "\substack{\text{var}\\ \text{preservation}\\ =}"] & & \\
\$x \arrow[rrrr, bend right, "h"'] & x \otimes \$x \arrow[rr, "\mathsf{id} \otimes h"'] \arrow[l, "\mathsf{meta}"'] & \substack{\text{metavariable}\\ \text{preservation}\\ =} & x \otimes \underline{\mathbf{M}} \arrow[r, "\mathsf{meta}"] & \underline{\mathbf{M}}
\end{tikzcd}$$

as we wanted. □

Lemma E.10 *The arrow* h *is a substitution monoid homomorphism.*

Proof. By the 'var preservation' equation, h preserves the unit of the monoids. It remains to show it preserves substitution:

$$\begin{tikzcd}
\$x \otimes \$x \arrow[r, "h \otimes h"] \arrow[d, "-[-]_{\mathbf{F}_{\mathbf{O}}x}"'] \arrow[dr, phantom, "="] & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[d, "-[-]_{\mathbf{M}}"] \\
\$x \arrow[r, "h"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

We do so using indexed parameterised induction. Denote the two sides of the square as follows:

$$g_1 : \$x \otimes \$x \xrightarrow{-[-]_{\mathbf{F}_{\mathbf{O}}x}} \$x\mathbf{M} \xrightarrow{h} \underline{\mathbf{M}} \qquad g_2 : \$x \otimes \$x \xrightarrow{h \otimes h} \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \xrightarrow{-[-]_{\mathbf{M}}} \underline{\mathbf{M}}$$

We will show that $g_1 = \mathsf{fold} f = g_2$.

Op case:

$$\begin{tikzcd}
(\mathbf{O}\$x) \otimes \$x \arrow[rrr, "{(\mathbf{O}(\mathsf{curry} g_1)) \otimes \mathsf{id}}"] \arrow[dr, "{\mathsf{str}_{\mathsf{var}}}"] \arrow[ddd, "{\mathbf{F_O}x\,[\![-]\!] \otimes \mathsf{id}}"'] \arrow[drr, phantom, "{\substack{\text{str-nat}\\ =}}"] & & & (\mathbf{O}(\underline{\mathbf{M}} \leftarrow\!\!\otimes \$x)) \otimes \$x \arrow[d, "{\mathsf{str}_{\mathsf{var}}}"] \\
& \mathbf{O}(\$x \otimes \$x) \arrow[rr, "{\mathbf{O}\left((\mathsf{curry} g_1) \otimes \mathsf{id}\right)}"] \arrow[dr, "{\mathbf{O}(-[-])}"'] \arrow[drr, phantom, "{\substack{\text{exponential}\\ =}}"] \arrow[ddl, phantom, "{\substack{\text{compatible}\\ \text{monoid}\\ =}}"] & & \mathbf{O}((\underline{\mathbf{M}} \leftarrow\!\!\otimes \$x) \otimes \$x) \arrow[d, "{\mathbf{O}\mathsf{eval}}"] \\
& & \mathbf{O}\$x \arrow[r, "{\mathbf{O}h}"] \arrow[d, "{[\![-]\!]}"'] \arrow[dr, phantom, "{\substack{\text{op}\\ \text{preservation}\\ =}}"] & \mathbf{O}\underline{\mathbf{M}} \arrow[d, "{\mathbf{M}\,[\![-]\!]}"] \\
\$x \otimes \$x \arrow[rr, "{-[-]}"'] & & \$x \arrow[r, "{h}"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

$$\begin{tikzcd}
(\mathbf{O}\$x) \otimes \$x \arrow[rrr, "{(\mathbf{O}(\mathsf{curry} g_2)) \otimes \mathsf{id}}"] \arrow[dr, bend left, "{\mathsf{str}_{\mathsf{var}}}"] \arrow[ddr, bend right, "{(\mathbf{O}h) \otimes h}"] \arrow[ddd, "{\mathbf{F_O}x\,[\![-]\!] \otimes \mathsf{id}}"'] \arrow[drrr, phantom, "{\substack{\text{str-nat}\\ =}}"] & & & (\mathbf{O}(\underline{\mathbf{M}} \leftarrow\!\!\otimes \$x)) \otimes \$x \arrow[d, "{\mathsf{str}_{\mathsf{var}}}"] \\
& \mathbf{O}(\$x \otimes \$x) \arrow[rr, "{\mathbf{O}\left((\mathsf{curry} g_1) \otimes \mathsf{id}\right)}"] \arrow[dr, "{\mathbf{O}(h \otimes h)}"] \arrow[d, phantom, "{\substack{\mathsf{str\ nat}\\ h : \mathbb{I}^{\bullet} \to \mathsf{var}\\ =}}"] \arrow[drr, phantom, "{\substack{\text{exponential}\\ =}}"] & & \mathbf{O}((\underline{\mathbf{M}} \leftarrow\!\!\otimes \$x) \otimes \$x) \arrow[d, "{\mathbf{O}\mathsf{eval}}"] \\
& (\mathbf{O}\underline{\mathbf{M}}) \otimes \underline{\mathbf{M}} \arrow[r, "{\mathsf{str}_h}"] \arrow[dr, "{[\![-]\!] \otimes \mathsf{id}}"] \arrow[d, phantom, "{\substack{\text{op preservation}\\ \text{functoriality}\\ =}}"] & \mathbf{O}(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[r, "{\mathbf{O}(-[-])}"'] \arrow[dr, phantom, "{\substack{\text{compatible monoid}\\ =}}"] & \mathbf{O}\underline{\mathbf{M}} \arrow[d, "{\mathbf{M}\,[\![-]\!]}"] \\
\$x \otimes \$x \arrow[rr, "{h \otimes h}"'] & & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[r, "{-[-]}"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

Var case:

$$\begin{tikzcd}
\mathbb{I} \otimes \$x \arrow[rr, equal] \arrow[dd, "{\mathsf{var} \otimes \mathsf{id}}"'] \arrow[ddr, "{\ell}"] \arrow[dd, phantom, shift left=4ex, "{\substack{\text{left}\\ \text{unit}\\ =}}"] & & \mathbb{I} \otimes \$x \arrow[d, "{\mathsf{id} \otimes h}"] \\
& \substack{\ell\text{-nat}\\ =} & \mathbb{I} \otimes \underline{\mathbf{M}} \arrow[d, "{\ell}"] \\
\$x \otimes \$x \arrow[r, "{-[-]}"'] & \$x \arrow[r, "{h}"'] & \underline{\mathbf{M}}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{I} \otimes \$x \arrow[rr, equal] \arrow[dd, "{\mathsf{var} \otimes \mathsf{id}}"'] & & \mathbb{I} \otimes \$x \arrow[d, "{\mathsf{id} \otimes h}"] \\
& \substack{\text{var preservation}\\ +\ \text{functoriality}\\ =} & \mathbb{I} \otimes \underline{\mathbf{M}} \arrow[d, "{\ell}"] \arrow[dl, "{\mathsf{env} \otimes \mathsf{id}}"'] \arrow[dl, phantom, shift left=3ex, "{\substack{\text{left}\\ \text{unit}\\ =}}"] \\
\$x \otimes \$x \arrow[r, "{h \otimes h}"'] & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[r, "{-[-]}"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

Metavariable case:

$$\begin{tikzcd}
(x \otimes \$x) \otimes \$x \arrow[rrr, "{(x \otimes (\mathsf{curry} g_1)) \otimes \mathsf{id}}"] \arrow[dr, "{\mathbf{a}}"] \arrow[ddd, "{\mathsf{meta} \otimes \mathsf{id}}"'] \arrow[drr, phantom, "{\substack{\mathbf{a}\text{-nat}\\ =}}"] & & & (x \otimes (\underline{\mathbf{M}} \leftarrow\!\!\otimes \$x)) \otimes \$x \arrow[d, "{\mathbf{a}}"] \\
& x \otimes (\$x \otimes \$x) \arrow[rr, "{\mathsf{id} \otimes ((\mathsf{curry} g_1) \otimes \mathsf{id})}"] \arrow[dr, "{\mathsf{id} \otimes (-[-])}"'] \arrow[drr, phantom, "{\substack{\text{exponential}\\ =}}"] \arrow[ddl, phantom, "{\substack{\text{metavariable}\\ \text{case}\\ =}}"] & & x \otimes ((\underline{\mathbf{M}} \leftarrow\!\!\otimes \$x) \otimes \$x) \arrow[d, "{\mathsf{id} \otimes \mathsf{eval}}"] \\
& & x \otimes \$x \arrow[r, "{\mathsf{id} \otimes h}"] \arrow[d, "{\mathsf{meta}}"'] \arrow[dr, phantom, "{\substack{\text{metavariable}\\ \text{preservation}\\ =}}"] & x \otimes \underline{\mathbf{M}} \arrow[d, "{\mathsf{meta}^{\mathbf{M}}}"] \\
\$x \otimes \$x \arrow[rr, "{-[-]}"'] & & \$x \arrow[r, "{h}"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

$$\begin{tikzcd}
(x \otimes \$x) \otimes \$x \arrow[rrrr, "(x \otimes (\mathsf{curry} g_2)) \otimes \mathsf{id}"] \arrow[drr, "\mathbf{a}"'] \arrow[ddr, bend right, "(\mathsf{id} \otimes h) \otimes h"'] \arrow[ddd, "\mathsf{meta} \otimes \mathsf{id}"'] \arrow[rr, phantom, shift right=3ex, "\overset{\mathbf{a}\text{-nat}}{=}"] & & & & (x \otimes (\underline{\mathbf{M}} \mathbin{\leftarrow\!\!\otimes} \$x)) \otimes \$x \arrow[d, "\mathbf{a}"] \\
 & & x \otimes (\$x \otimes \$x) \arrow[rr, "\mathsf{id} \otimes ((\mathsf{curry} g_2) \otimes \mathsf{id})"] \arrow[dr, "\mathsf{id} \otimes (h \otimes h)"] & & x \otimes ((\underline{\mathbf{M}} \mathbin{\leftarrow\!\!\otimes} \$x) \otimes \$x) \arrow[d, "\mathsf{id} \otimes \mathsf{eval}"] \arrow[dl, phantom, "\overset{\text{exponential}}{=}"] \\
 & (x \otimes \underline{\mathbf{M}}) \otimes \underline{\mathbf{M}} \arrow[rr, "\mathbf{a}"] \arrow[dr, "\mathsf{meta} \otimes \mathsf{id}"] \arrow[rr, phantom, shift left=4ex, "\overset{\mathbf{a}\text{-nat}}{=}"] & & x \otimes (\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[r, "\mathsf{id} \otimes (-[-])"] & x \otimes \underline{\mathbf{M}} \arrow[d, "\mathsf{meta}^{\mathbf{M}}"] \\
\$x \otimes \$x \arrow[rr, "h \otimes h"'] \arrow[rr, phantom, shift left=3ex, "\overset{\text{metavariable preservation}}{=}"] & & \underline{\mathbf{M}} \otimes \underline{\mathbf{M}} \arrow[rr, "-[-]"'] \arrow[rr, phantom, shift left=3ex, "\overset{\text{metavariable case}}{=}"] & & \underline{\mathbf{M}}
\end{tikzcd}$$

as we wanted. □