Multicategorical Semantics for Untyped Effects

Liron Cohen[a] Ariel Grunfeld[a]

[a] *Faculty of Computer and Information Science*
Ben-Gurion University
Be'er Sheva, Israel

Abstract

Completeness proofs in categorical semantics usually proceed by building a syntactic category whose composition is given by substitution. For untyped effectful Call-by-Value languages, this runs into a basic obstacle: there is no canonical notion of simultaneous substitution of computations, since evaluation order is semantically meaningful. We address this by taking single computation substitutions, that is, binding steps, as primitive, and representing computation substitution by finite *sequential* lists composed by concatenation. We formalize this idea in a one-object Freyd-multicategorical setting. We introduce *Freyd operads*, separating a cartesian operad of values from a symmetric Ren-cartesian preoperad of computations, connected by a Freyd functor, and from any Freyd operad we construct a corresponding *Freyd PROP of substitutions*. We prove that this construction is representable and, in the strict one-object setting, left adjoint to restriction to codomain 1. Using the induced term model, we interpret untyped computational λ-calculus with procedures and higher-order functions in weakly closed Freyd operads, and prove soundness, initiality, and completeness. This yields a categorical semantics tailored to untyped effectful computation and broad enough to encompass realizability-oriented models such as monadic combinatory algebras.

Keywords: Freyd multicategories, Freyd operads, sequential substitution, Call-by-Value, effects, categorical semantics, untyped computational lambda calculus, monadic combinatory algebras.

1 Introduction

Computational λ-calculus, introduced by Moggi [17], provides an abstract language for effectful computation and has long served as a central test case for categorical semantics. For the typed language, the semantic story is well developed: from Moggi's λ_c-models [17], through abstract Kleisli categories [9] and Freyd categories [15], completeness is typically established by constructing a *term model*, namely a category whose objects are types and whose morphisms are terms modulo the equational theory. Completeness then follows because the valid equations in the term model are exactly the equations of the theory.

This type-based framework, however, does not translate naturally to an untyped setting. To interpret terms with several free variables, one needs product objects to interpret contexts, and this in turn requires the language itself to contain suitable product types. In typed settings this is natural, but for an untyped language there is no non degenerate type structure in which to add such products. Thus, the standard types as objects construction is poorly suited to untyped completeness arguments.

A more flexible alternative is to use a term model in which contexts are the objects and simultaneous substitutions between contexts are the morphisms [21]. This subsumes the types and terms approach: a type is represented by a one variable context, a term of that type is represented by a substitution into that

⋆ This work was partially supported by Grant No. 2020145 from the United States-Israel Binational Science Foundation (BSF) and Grant No. 2876/25 from the Israeli Science Foundation (ISF).

context, and equations between terms become equations between the corresponding substitutions. In pure languages, simultaneous substitutions carry the structure of finite tensor products, given by concatenation. By using contexts as objects, one therefore avoids the need for product types altogether, making this perspective particularly natural for untyped languages.

However, this approach breaks down for effectful languages, because there is no canonical notion of simultaneous substitution for effectful computations. The reason is that effectful computation is inherently sequential, and any would be notion of simultaneous computation substitution must choose an evaluation order. For pure values, genuine *simultaneous* substitution is well behaved: if $s = (V_1/x_1, \cdots, V_n/x_n)$ then applying s to a term replaces each free occurrence of x_i by V_i in one step, and such substitutions compose in the usual way. For effectful computations, one might hope for an analogous notion of simultaneous substitution, replacing several *computation holes* at once. The obstruction is that computations are only used by *running* them, via the sequencing binder let. Concretely, consider the following computation contexts with two holes

$$K_1[(-),(-)] := \mathsf{let}\ x_1 \Leftarrow (-)\ \mathsf{in}\ \mathsf{let}\ x_2 \Leftarrow (-)\ \mathsf{in}\ \mathsf{return}\ x_1$$
$$K_2[(-),(-)] := \mathsf{let}\ x_1 \Leftarrow (-)\ \mathsf{in}\ \mathsf{let}\ x_2 \Leftarrow (-)\ \mathsf{in}\ \mathsf{return}\ x_2$$

If we wish to substitute computations M_1 and M_2 into the two holes, we must choose an evaluation order, producing either $K_1[M_1, M_2] = \mathsf{let}\ x_1 \Leftarrow M_1\ \mathsf{in}\ \mathsf{let}\ x_2 \Leftarrow M_2\ \mathsf{in}\ \mathsf{return}\ x_1$ or $K_2[M_2, M_1] = \mathsf{let}\ x_1 \Leftarrow M_2\ \mathsf{in}\ \mathsf{let}\ x_2 \Leftarrow M_1\ \mathsf{in}\ \mathsf{return}\ x_2$. In the presence of effects, these two programs need not be equivalent. Sequencing is part of the meaning, and there is therefore no canonical simultaneous computation substitution operation that forgets this order.

Thus, for untyped effectful languages, one encounters a genuine conundrum. On the one hand, the usual types as objects approach is unavailable, because there are no product types to interpret contexts. On the other hand, the contexts and substitutions approach also fails, because computations do not admit a canonical simultaneous substitution operation. The right replacement is suggested by the analysis above: single computation substitutions, that is, individual binding steps, should be taken as primitive, and general computation substitutions should be represented by finite lists of such single substitutions. Composition is then given by concatenation, namely by sequencing the binding steps.

A natural setting in which this phenomenon already appears is that of Freyd multicategories [26]. There, values and computations are separated from the outset, and computations carry an order sensitive notion of substitution. This works just as well for untyped languages as for typed ones, and it leads to a straightforward completeness argument. At the same time, Freyd multicategories remain relatively unfamiliar, and using them directly as semantic targets obscures the connection with more standard categorical semantics [25]. Previous suggestions for categorical semantics of untyped computational λ-calculus appear in [18,8], but without the completeness result obtained here.

The point of this paper is that one need not choose between the two viewpoints. We take the multicategorical semantics as the correct structural starting point, and from it derive a *categorical* substitution construction tailored to the untyped effectful setting. This construction is semantic rather than syntax dependent, and can therefore be applied uniformly to arbitrary Freyd operadic data. It also extends a familiar pattern from ordinary multicategory theory, namely the free monoidal category generated by a multicategory. In the present sequential setting, the resulting category is no longer one of simultaneous substitutions, but one of ordered lists of single substitutions. In particular, in the strict one object setting relevant here, our construction yields the left adjoint to restriction to codomain 1, providing the categorical counterpart of Freyd multicategorical substitution for untyped effectful programs.

Beyond the categorical issue itself, our motivation comes from effectful realizability, in particular from recent work on syntactic effectful realizability and evidence frame semantics [6,7,28]. For realizability with effects, one wants an untyped computational core language whose terms can serve as realizers, together with a completeness theorem strong enough to interpret those realizers in algebraic or combinatory models such as monadic combinatory algebras [5]. The semantics developed here are designed with exactly this goal in mind: they provide a flexible untyped call by value core together with a class of categorical models broad enough to encompass effectful combinatory structures.

Main Contributions:

- We introduce *Freyd operads* as the one object structures that separate pure values from effectful computations in the untyped setting, together with *weakly closed* structure for abstraction and application.
- From a Freyd operad, we construct a category of substitutions whose morphisms are finite lists of single substitutions and renamings, modulo the appropriate structural equations, yielding a Freyd PROP of substitutions.
- We prove that this construction is representable and left adjoint to restriction to codomain 1, thereby recovering a categorical semantics from Freyd operadic data in the strict untyped setting.
- We interpret the untyped computational language $\lambda_{\mathrm{ml}*}$ in weakly closed Freyd operads and prove soundness, initiality of the term model, and completeness.
- We relate the resulting semantics to examples arising from monadic combinatory algebras, restriction categorical applicative systems, and reflexive object style models.

Outline: The paper is organized as follows. Section 2 introduces $\lambda_{\mathrm{ml}*}$ and its equational theory and recalls the categorical background. Section 3 develops Freyd operads and the free substitution construction, culminating in representability and the adjunction with Freyd PROPs. Section 4 gives the interpretation of $\lambda_{\mathrm{ml}*}$ in weakly closed Freyd operads and proves soundness, initiality, and completeness. Section 5 discusses examples and connections to realizability motivated and categorical models. Detailed proofs and constructions can be found in Appendix **??**.

2 Background: Untyped Computational Lambda Calculus

This section recalls the category of renamings, the syntax and equational theory of $\lambda_{\mathrm{ml}*}$, and the categorical vocabulary used in our semantics, in a concrete $\mathbb{N}$-indexed form that treats a number as a context size.

2.1 Renamings

Throughout, we use n, m, k for natural numbers and write $[n] = \{1, \ldots, n\}$ (with $[0] = \emptyset$).

Definition 2.1 (Renaming) *A* renaming $r : m \to n$ *is a function* $[m] \to [n]$. *We write* $\mathbf{Ren}\,(m \Rightarrow n)$ *for the set of such renamings. A bijective renaming is a* permutation *and we write* $\mathbf{Perm}\,(n) \subseteq \mathbf{Ren}\,(n \Rightarrow n)$.

We identify a renaming $r : [m] \to [n]$ with the tuple $(r(1), \ldots, r(m))$. The identity is $\mathsf{id}_n = (1, \ldots, n)$. Composition is ordinary composition of functions, and the tensor (coproduct) of $r \in \mathbf{Ren}\,(m_1 \Rightarrow n_1)$ and $r' \in \mathbf{Ren}\,(m_2 \Rightarrow n_2)$ is the block sum

$$r + r' \;:=\; \big(r(1), \ldots, r(m_1),\; n_1 + r'(1), \ldots, n_1 + r'(m_2)\big) \in \mathbf{Ren}\,(m_1 + m_2 \Rightarrow n_1 + n_2)\,.$$

If $r \in \mathbf{Ren}\,(m \Rightarrow n)$ and $r' \in \mathbf{Ren}\,(k \Rightarrow n)$ we write $[r, r'] \in \mathbf{Ren}\,(m + k \Rightarrow n)$ for their concatenation.

We single out the following basic renamings for swap, discard and copy:

$$\sigma_{m,n} := (n+1, \ldots, n+m,\, 1, \ldots, n) \in \mathbf{Ren}\,(m + n \Rightarrow n + m)$$
$$!_n := () \in \mathbf{Ren}\,(0 \Rightarrow n) \quad , \quad \Delta_n := (1, \ldots, n,\, 1, \ldots, n) \in \mathbf{Ren}\,(2n \Rightarrow n)\,.$$

Lemma 2.2 ([2,12]) *Every renaming* $r \in \mathbf{Ren}\,(m \Rightarrow n)$ *is generated by composition and tensor product from* id_1, $\sigma_{1,1}$, Δ_1, *and* $!_1$, *and every permutation* $r \in \mathbf{Perm}\,(n)$ *is generated by composition and tensor product from* id_1 *and* $\sigma_{1,1}$.

Lemma 2.3 ([2,12]) $\mathbf{Ren}$ *forms a cocartesian category with initial object* 0 *and coproduct on objects given by* $+$. *Moreover,* $\mathbf{Perm}$ *is a subgroupoid of* $\mathbf{Ren}$.

2.2 Untyped Computational Lambda Calculus

We work with an untyped, Call-by-Value variant of Moggi's computational λ calculus, similar to λ_{ml^*} in [24]. The syntax separates *values*, which do not perform effects, from *computations*, which may perform effects. The term former return injects values into computations, and $\mathsf{let}\ x \Leftarrow M_1\ \mathsf{in}\ M_2$ sequences computations by running M_1 and binding its returned value to x in M_2. As is standard, we identify terms

$$
\begin{array}{lll}
\textbf{Terms} & E & ::= V \mid M \\
\textbf{Values} & V & ::= x \mid f(V, \ldots, V) \mid \lambda x \,.\, M \\
\textbf{Computations} & M & ::= \mathsf{return}\ V \mid \mathsf{let}\ x \Leftarrow M\ \mathsf{in}\ M \mid p(V, \ldots, V) \mid VV
\end{array}
$$

$$
\frac{}{x \vdash_{\mathbf{v}} x} \qquad \frac{f \in \mathsf{Func}\,(n) \qquad \forall i \in \{1, \ldots, n\}.\, \Gamma_i \vdash_{\mathbf{v}} V_i}{\Gamma_1, \ldots, \Gamma_n \vdash_{\mathbf{v}} f(V_1, \ldots, V_n)} \qquad \frac{\Gamma, x \vdash_{\mathbf{c}} M}{\Gamma \vdash_{\mathbf{v}} \lambda x \,.\, M}
$$

$$
\frac{\Gamma \vdash_{\mathbf{v}} V}{\Gamma \vdash_{\mathbf{c}} \mathsf{return}\ V} \qquad \frac{p \in \mathsf{Proc}\,(n) \qquad \forall i \in \{1, \ldots, n\}.\, \Gamma_i \vdash_{\mathbf{v}} V_i}{\Gamma_1, \ldots, \Gamma_n \vdash_{\mathbf{c}} p(V_1, \ldots, V_n)} \qquad \frac{\Gamma_1, x \vdash_{\mathbf{c}} M_1 \qquad \Gamma_2 \vdash_{\mathbf{c}} M_2}{\Gamma_1, \Gamma_2 \vdash_{\mathbf{c}} \mathsf{let}\ x \Leftarrow M_2\ \mathsf{in}\ M_1}
$$

$$
\frac{\Gamma_1 \vdash_{\mathbf{v}} V_1 \qquad \Gamma_2 \vdash_{\mathbf{v}} V_2}{\Gamma_1, \Gamma_2 \vdash_{\mathbf{c}} V_1\, V_2} \qquad \frac{\Gamma \vdash E \quad x \notin \Gamma}{\Gamma, x \vdash E} \qquad \frac{x_1, \ldots, x_n \vdash E \quad r \in \mathbf{Perm}\,(n)}{x_{r(1)}, \ldots, x_{r(n)} \vdash E} \qquad \frac{\Gamma, x_1, x_2 \vdash E}{\Gamma, y \vdash E\,\{y/x_1, y/x_2\}}
$$

Fig. 1. Syntax and well-formedness rules for $\lambda_{\mathrm{ml}*}$.

up to α equivalence. All substitutions are capture avoiding, and we freely rename bound variables to avoid capture.

The syntax of $\lambda_{\mathrm{ml}*}$ is given in the top part of Fig. 1. The language is parameterized by a signature, which is a pair $\Sigma = (\mathsf{Func}, \mathsf{Proc})$, with two families of sets, parametrized by $\mathbb{N}$, where for each $n \in \mathbb{N}$, $\mathsf{Func}\,(n)$ is a set of function symbols of arity n, and $\mathsf{Proc}\,(n)$ is a set of procedure symbols of arity n. We assume a countably infinite set Var of variables. Values are either variables, the application of a function symbol of arity n on n values, or lambda abstractions, binding a variable in a computation. Computations are either a lifted value, a binding of a computation to a variable within another computation, the application of a procedure symbol of arity n on n values, or application of a value to another value. Note that application could be treated as a procedure symbol, but we give it a dedicated rule, so we make it a separate constructor. We track free variables using *contexts*, which are finite lists of distinct variables $\Gamma = x_1, \ldots, x_n$. We write Γ, x, Γ' for context concatenation with x in the middle. From now on, terms are always considered relative to a context listing all of their free variables.

Because $\lambda_{\mathrm{ml}*}$ is untyped, there is no typing judgment. Instead, there are two well-formedness judgments, whose rules are given in the bottom part of Fig. 1: $\Gamma \vdash_{\mathbf{v}} V$ for values and $\Gamma \vdash_{\mathbf{c}} M$ for computations, each requiring that all free variables occur in Γ. We write $\Gamma \vdash E$ for the appropriate judgment depending on whether E is a value or a computation.

To define the theory and semantics of $\lambda_{\mathrm{ml}*}$, we use substitutions that replace free variables with terms. Since $\lambda_{\mathrm{ml}*}$ is Call-by-Value, variables range over values, so only values may be substituted for variables. Computations can enter only via sequencing with $\mathsf{let} - \Leftarrow - \mathsf{in} -$. Although the equational theory uses only single-variable substitution, it is convenient to use simultaneous substitutions, which we therefore define for values. This restriction is also conceptually important: there is no well behaved notion of simultaneous substitution by computations, since computations interact through sequencing and evaluation order.

Definition 2.4 (Simultaneous Substitution) *Let $\Gamma = x_1, \ldots, x_n$ be a context. A* (simultaneous) substitution into Γ *is a list $s = V_1/x_1, \ldots, V_n/x_n$ such that each V_i is a well-formed value in some context Γ_i (i.e. $\Gamma_i \vdash_{\boldsymbol{v}} V_i$). We view s as a substitution* from *the list of source contexts $\Gamma_1, \ldots, \Gamma_n$* to *$\Gamma$, and write $s \in \mathsf{Sub}\,(\Gamma_1, \ldots, \Gamma_n \Rightarrow \Gamma)$. Given $s = V_1/x_1, \ldots, V_n/x_n \in \mathsf{Sub}\,(\Gamma_1, \ldots, \Gamma_n \Rightarrow \Gamma)$ and a well formed term $\Gamma \vdash t$, we write $t\,\{s\}$ for the result of applying s to t. It is defined by structural recursion on t, replacing each free occurrence of x_i by V_i (and avoiding capture in the λ-case).*

Lemma 2.5 *If $s \in \mathsf{Sub}\,(\Gamma_1 \Rightarrow \Gamma_2)$ and $\Gamma_2 \vdash E$, then $\Gamma_1 \vdash E\,\{s\}$.*

The theory of $\lambda_{\mathrm{ml}*}$ is presented in Fig. 2. The equational theory is the least congruence closed under the term formers, generated by the monad laws for $\mathsf{return} -$ and $\mathsf{let} - \Leftarrow - \mathsf{in} -$. Intuitively, (lunit) and (runit) say that $\mathsf{return} -$ is the unit for sequencing, and (assoc) states associativity of sequencing.

2.3 Freyd Categories

We use Freyd-style categorical semantics to model Call-by-Value effectful computation [15]. Intuitively, the pure fragment (values) forms a cartesian theory of renamings, copying, and discarding, while computations form a premonoidal category where sequencing is explicit. We package both components in a Freyd

$$\frac{}{\Gamma \vdash E \equiv E}\ (\text{id}) \qquad \frac{\Gamma \vdash E_1 \equiv E_2}{\Gamma \vdash E_2 \equiv E_1}\ (\text{sym}) \qquad \frac{\Gamma \vdash E_1 \equiv E_2 \quad \Gamma \vdash E_2 \equiv E_3}{\Gamma \vdash E_1 \equiv E_3}\ (\text{trans})$$

$$\frac{\Gamma \vdash_{\mathbf{v}} V \equiv V'}{\Gamma \vdash_{\mathbf{c}} \mathsf{return}\ V \equiv \mathsf{return}\ V'}\ (\text{ret}) \qquad \frac{\forall i \in \{1,\dots,k\}.\ \Gamma_i \vdash_{\mathbf{v}} V_i \equiv V_i'}{\Gamma_1,\dots,\Gamma_n \vdash_{\mathbf{v}} f\,(V_1,\dots,V_k) \equiv f\,(V_1',\dots,V_k')}\ (\text{f-app})$$

$$\frac{\Gamma_1, x \vdash_{\mathbf{c}} M_1 \equiv M_1' \quad \Gamma_2 \vdash_{\mathbf{c}} M_2 \equiv M_2'}{\Gamma_1,\Gamma_2 \vdash_{\mathbf{c}} \mathsf{let}\ x \Leftarrow M_2\ \mathsf{in}\ M_1 \equiv \mathsf{let}\ x \Leftarrow M_2'\ \mathsf{in}\ M_1'}\ (\text{let}) \qquad \frac{\forall i \in \{1,\dots,k\}.\ \Gamma_i \vdash_{\mathbf{v}} V_i \equiv V_i'}{\Gamma_1,\dots,\Gamma_n \vdash_{\mathbf{c}} p\,(V_1,\dots,V_k) \equiv p\,(V_1',\dots,V_k')}\ (\text{p-app})$$

$$\frac{\Gamma_1, x \vdash_{\mathbf{c}} M \quad \Gamma_2 \vdash_{\mathbf{v}} V}{\Gamma_1,\Gamma_2 \vdash_{\mathbf{c}} \mathsf{let}\ x \Leftarrow \mathsf{return}\ V\ \mathsf{in}\ M \equiv M\{V/x\}}\ (\text{lunit}) \qquad \frac{\Gamma \vdash_{\mathbf{c}} M}{\Gamma \vdash_{\mathbf{c}} \mathsf{let}\ x \Leftarrow M\ \mathsf{in}\ \mathsf{return}\ x \equiv M}\ (\text{runit})$$

$$\frac{\Gamma_1, x_2 \vdash_{\mathbf{c}} M \quad \Gamma_2, x_1 \vdash_{\mathbf{c}} M_2 \quad \Gamma_3 \vdash_{\mathbf{c}} M_1}{\Gamma_1,\Gamma_2,\Gamma_3 \vdash_{\mathbf{c}} \mathsf{let}\ x_2 \Leftarrow (\mathsf{let}\ x_1 \Leftarrow M_1\ \mathsf{in}\ M_2)\ \mathsf{in}\ M \equiv \mathsf{let}\ x_1 \Leftarrow M_1\ \mathsf{in}\ \mathsf{let}\ x_2 \Leftarrow M_2\ \mathsf{in}\ M}\ (\text{assoc})$$

$$\frac{\Gamma_1, x \vdash_{\mathbf{c}} M \quad \Gamma_2 \vdash_{\mathbf{v}} V}{\Gamma_1,\Gamma_2 \vdash_{\mathbf{c}} (\lambda x\,.\,M)\ V \equiv M\{V/x\}}\ (\text{beta}) \qquad \frac{\Gamma, x \vdash_{\mathbf{c}} M \equiv M'}{\Gamma \vdash_{\mathbf{v}} \lambda x\,.\,M \equiv \lambda x\,.\,M'}\ (\text{abs}) \qquad \frac{\Gamma_1 \vdash_{\mathbf{v}} V_1 \equiv V_1' \quad \Gamma_2 \vdash_{\mathbf{v}} V_2 \equiv V_2'}{\Gamma_1,\Gamma_2 \vdash_{\mathbf{c}} V_1\,V_2 \equiv V_1'\,V_2'}\ (\text{app})$$

Fig. 2. Equational theory of $\lambda_{\text{ml}*}$.

structure via a functor $J : \mathbb{V} \to \mathbb{C}$ that embeds pure maps as central morphisms. Throughout, a natural number n is read as a context of n variables. A morphism $m \to n$ should be read as an n-tuple of effectful bindings that may use m variables as input. With this convention, addition $m + n$ represents context concatenation, and whiskering by k on the left or right extends a morphism with k unused variables. The definitions below are standard [22,19,1]. We spell them out in the strict $\mathbb{N}$-indexed form used later in the paper.

Definition 2.6 (Pre-PROP) *A pre-PROP $\mathbb{D}$ is a strict symmetric premonoidal category with the natural numbers as objects and addition as tensor product. We write $k \dashv (-)$ and $(-) \vdash k$ for the left and right whiskering endofunctors induced by tensoring with k, and $\sigma_{m,n} : m + n \to n + m$ for the symmetry isomorphisms.*

Definition 2.7 (Centrality in a Pre-PROP) *Given a pre-PROP $\mathbb{D}$, a morphism $f \in \mathbb{D}\,(m \Rightarrow n)$ is* central, *if for every $g \in \mathbb{D}\,(m' \Rightarrow n')$, $f \vdash n' \circ m \dashv g = n \dashv g \circ f \vdash m'$ and $g \vdash n \circ m' \dashv f = n' \dashv f \circ g \vdash m$.*

Definition 2.8 (PROP) *A PROP is a pre-PROP where all morphisms are central.*

Definition 2.9 (Cartesian PROP) *A PROP $\mathbb{D}$ is* Cartesian *if it is equipped with natural transformations: copy, $\Delta_n : n \to 2n$ and discard, $!_n : n \to 0$, forming a commutative comonoid for each n, with the following coherence condition: $\Delta_{n_1+n_2} \equiv \big(n_1 + \sigma_{n_1,n_2} + n_2\big) \circ \big(\Delta_{n_1} + \Delta_{n_2}\big)$ and $!_{n_1+n_2} \equiv !_{n_1} + !_{n_2}$.*

Our arrow convention reads $m \to n$ as producing n outputs from m inputs (substitution from an input context to an output context). This is the standard Lawvere theory presentation [13].

Definition 2.10 (Freyd PROP) *A Freyd PROP is a triple $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ where $\mathbb{V}$ is a cartesian PROP, $\mathbb{C}$ is a pre-PROP, and $\mathcal{J} : \mathbb{V} \longrightarrow \mathbb{C}$ is a pre-PROP functor such that for every $m, n \in \mathbb{N}$ and every $v \in \mathbb{V}\,(m \Rightarrow n)$, the morphism $\mathcal{J}(v)$ is central in $\mathbb{C}$.*

Definition 2.11 (Structure preserving functors)

- *A* pre-PROP functor $F : \mathbb{D}_1 \to \mathbb{D}_2$ *is a strict symmetric premonoidal functor that is the identity on objects (hence preserves addition on objects), and preserves whiskering and symmetries.*
- *A* cartesian PROP functor *is a pre-PROP functor that also preserves copy and discard: $F(\Delta_n) = \Delta_n$ and $F(!_n) = !_n$ for all n.*
- *A* Freyd PROP functor *between Freyd PROPs $(\mathbb{V}_1, \mathbb{C}_1, \mathcal{J}_1) \to (\mathbb{V}_2, \mathbb{C}_2, \mathcal{J}_2)$ is a pair $(F^{\mathbb{V}}, F^{\mathbb{C}})$ where $F^{\mathbb{V}}$ is a cartesian PROP functor, $F^{\mathbb{C}}$ is a pre-PROP functor, and $F^{\mathbb{C}}\big(\mathcal{J}_1(v)\big) = \mathcal{J}_2\big(F^{\mathbb{V}}(v)\big)$ for all v.*

3 Freyd operads and free Freyd PROPs

This section introduces the Freyd-operadic structures used in our semantics and the associated substitution construction. A Freyd operad consists of a cartesian operad of values, a symmetric Ren-cartesian preoperad of computations, and a cartesian functor embedding values into computations. From any Freyd operad we construct a corresponding Freyd PROP of substitutions, whose morphisms are finite lists of elementary substitution and renaming steps, quotiented by the equations forced by the operadic structure.

Sections 3.1 and 3.2 provide the required one-object, $\mathbb{N}$-indexed background. The main novel material starts in Section 3.3, where we define the substitution construction, and continues in Section 3.4 (representability) and Section 3.5 (the adjunction).

3.1 Preoperads and reindexing

Operads and multicategories provide a standard account of algebraic structure in terms of operations with many inputs and one output, with composition corresponding to *parallel* substitution. For effectful Call-by-Value computation, however, the basic structural operation is *sequencing*: substituting a computation into a hole is inherently ordered, and evaluation order is semantically meaningful. Following Staton–Levy's premulticategorical treatment of impure languages [26], and specializing to the untyped setting, we work with the strict $\mathbb{N}$-indexed one-object form of premulticategories. Although the notions in this subsection are standard there, we spell them out to fix notation for the free substitution construction of Sec. 3.3. Since we work in an untyped setting, a morphism in arity n is simply an element of a family $\mathbb{C}(n)$, read as an n-ary operation. Composition is given by substituting an operation into a designated input position. To model renaming of free variables, we equip preoperads with a coherent reindexing action by renamings.

Definition 3.1 (Preoperad) *A preoperad is given by the following data:*

- *For each $n \in \mathbb{N}$, a collection $\mathbb{C}(n)$ of* morphisms
- Identity morphism*: A morphism $\mathsf{id} \in \mathbb{C}(1)$*
- Composition*: Given a morphism $g \in \mathbb{C}(m)$ and a morphism $f \in \mathbb{C}(n_1 + 1 + n_2)$, there is a morphism $f\{n_1 \dashv g \vdash n_2\} \in \mathbb{C}(n_1 + m + n_2)$*

satisfying the following laws:

(i) Left Unit. *For all $f \in \mathbb{C}(m)$: $\mathsf{id}\{f\} \equiv f$*

(ii) Right Unit. *For all $f \in \mathbb{C}(n_1 + 1 + n_2)$: $f\{n_1 \dashv \mathsf{id} \vdash n_2\} \equiv f$*

(iii) Associativity. *For all $f \in \mathbb{C}(m_1 + 1 + m_2)$, $g \in \mathbb{C}(n_1 + 1 + n_2)$, and $h \in \mathbb{C}(n)$:*
$f\{m_1 \dashv g \vdash m_2\}\{m_1 + n_1 \dashv h \vdash n_2 + m_2\} \equiv f\{m_1 \dashv g\{n_1 \dashv h \vdash n_2\} \vdash m_2\}$

Concretely, if $f \in \mathbb{C}(m_1 + 1 + m_2)$ and $g \in \mathbb{C}(m)$, then $f\{m_1 \dashv g \vdash m_2\} \in \mathbb{C}(m_1 + m + m_2)$ is obtained by inserting g into the distinguished input of f (the (m_1+1)st input), leaving the other inputs untouched. In terms of multicategories, one may think of this as composing f with a tuple of arguments in which all positions are filled by identities except the distinguished one, which is filled by g:

$$f\{m_1 \dashv g \vdash m_2\} \;\approx\; f\big(\underbrace{\mathsf{id}, \ldots, \mathsf{id}}_{m_1},\, g,\, \underbrace{\mathsf{id}, \ldots, \mathsf{id}}_{m_2}\big)$$

In the strict $\mathbb{N}$-indexed presentation this plays the role of *whiskering with identities* to make arities line up for composition. For example, take $f \in \mathbb{C}(2+1+0) = \mathbb{C}(3)$ and $g \in \mathbb{C}(4)$. Then $f\{2 \dashv g \vdash 0\} \in \mathbb{C}(2+4+0) = \mathbb{C}(6)$. If $h \in \mathbb{C}(1)$, associativity specializes to the arity check $\Big(f\{2 \dashv g \vdash 0\}\Big)\{2+1 \dashv h \vdash 2\} \;\equiv\; f\{2 \dashv (g\{1 \dashv h \vdash 2\}) \vdash 0\}$, where both sides have arity 6.

The terminology for cartesian premulticategories in [26] is not compositional: a cartesian multicategory is not simply a cartesian premulticategory that is also a multicategory, but requires additional naturality with respect to renamings. To keep the one-object presentation compositional, we explicitly separate the presence of a coherent renaming action ($\mathcal{S}$-cartesian) from the additional naturality conditions that recover fully cartesian (multi)categorical structure.

Definition 3.2 ($\mathcal{S}$-Cartesian Preoperad) *Given a subcategory $\mathcal{S}$ of* **Ren**, *closed to $+$ as a strict tensor product, a preoperad $\mathbb{V}$ is $\mathcal{S}$-cartesian when for every $r \in \mathcal{S}(m \Rightarrow n)$ and $v \in \mathbb{V}(m)$, there is a morphism $v[r] \in \mathbb{V}(n)$, such that:*

(i) $v[\mathsf{id}_n] \equiv v$

(ii) $v[r_2 \circ r_1] \equiv v[r_1][r_2]$

(iii) *For all $u \in \mathbb{V}(n_1 + 1 + n_2)$, $v \in \mathbb{V}(n)$, $r_i \in \mathcal{S}(m_i \Rightarrow n_i)$, $s \in \mathcal{S}(m \Rightarrow n)$:*
$u\{n_1 \dashv v \vdash n_2\}[r_1 + s + r_2] \equiv u[r_1 + \mathsf{id}_1 + r_2]\{m_1 \dashv v[s] \vdash m_2\}$

Intuitively, symmetric preoperads express that reindexing by permutations (exchange) interacts naturally with single substitution.

Definition 3.3 (Symmetric Preoperad) *A preoperad $\mathbb{C}$ is called a* symmetric preoperad *when it is* **Perm**-*cartesian, and $\sigma_{1,n}$ and $\sigma_{n,1}$ are natural in n, i.e., for every $f \in \mathbb{C}(m_1 + 2 + m_2)$ and $g \in \mathbb{C}(n)$:*

$$f[\mathsf{id}_{m_1} + \sigma_{1,1} + \mathsf{id}_{m_2}]\{m_1 \dashv g \vdash 1 + m_2\} \equiv f\{m_1 + 1 \dashv g \vdash m_2\}[\mathsf{id}_{m_1} + \sigma_{1,n} + \mathsf{id}_{m_2}]$$
$$f[\mathsf{id}_{m_1} + \sigma_{1,1} + \mathsf{id}_{m_2}]\{m_1 + 1 \dashv g \vdash m_2\} \equiv f\{m_1 \dashv g \vdash 1 + m_2\}[\mathsf{id}_{m_1} + \sigma_{n,1} + \mathsf{id}_{m_2}]$$

In the pure fragment, weakening and contraction are harmless: weakening adds an unused variable, and contraction merely identifies two variables, both of which commute with substitution. In contrast, in a Call-by-Value effectful language, weakening or contraction at the level of computations would implicitly allow discarding or duplicating effects. For example, even if x does not occur free in a computation M_1, the term $\mathsf{let}\ x \Leftarrow M_2\ \mathsf{in}\ M_1$ is not equivalent to M_1 in general, since the effect of M_2 is still performed. Exchange is different: it only reorders variables in the surrounding context and does not change what is evaluated or when. Accordingly, our effectful structure requires naturality for permutations (exchange), while copying and discarding are confined to the pure cartesian part.

Preoperad functors are the structure-preserving morphisms between preoperads, namely the one-object specializations of premulticategory morphisms from [26]. In the strict $\mathbb{N}$-indexed presentation there is no separate object component, since arities already encode lists of the unique object. In the cartesian case, such functors must moreover preserve reindexing as well as substitution.

Definition 3.4 (Preoperad Functor) *Given preoperads $\mathbb{C}_1$ and $\mathbb{C}_2$, a preoperad functor $\mathcal{G}$ from $\mathbb{C}_1$ to $\mathbb{C}_2$, $\mathcal{G} : \mathbb{C}_1 \longrightarrow \mathbb{C}_2$, is an assignment of a morphism $\mathcal{G}f \in \mathbb{C}_2(n)$ for each $f \in \mathbb{C}_1(n)$, such that $\mathcal{G}\mathsf{id} \equiv \mathsf{id}$ and $\mathcal{G}(f\{m_1 \dashv g \vdash m_2\}) \equiv (\mathcal{G}f)\{m_1 \dashv \mathcal{G}g \vdash m_2\}$. Together, these two properties are called* functoriality.

Definition 3.5 (Cartesian Preoperad Functor) *A preoperad functor $\mathcal{G}$ between* **Ren**-*cartesian preoperads is called* cartesian *if it preserves all renamings: $\mathcal{G}(f[r]) \equiv (\mathcal{G}f)[r]$*

3.2 Centrality and Freyd operads

Composition in a premulticategory is inherently sequential, since one substitutes into a chosen input position. Centrality identifies those morphisms for which the order of substitution does not matter, up to the evident arity adjustment. In general, multicategories are recovered as the central part of premulticategories [26]. In our one-object, $\mathbb{N}$-indexed setting, this recovers operads from preoperads and isolates the fragment relevant to our semantics, where values are cartesian and computations admit only exchange.

Definition 3.6 *We say $g_1 \in \mathbb{C}(n_1)$* commutes *with $g_2 \in \mathbb{C}(n_2)$ when the following holds for every $f \in \mathbb{C}(m_1 + 1 + m + 1 + m_2)$:*

$$f\{m_1 \dashv g_1 \vdash m + 1 + m_2\}\{m_1 + n_1 + m \dashv g_2 \vdash m_2\} \equiv f\{m_1 + 1 + m \dashv g_2 \vdash m_2\}\{m_1 \dashv g_1 \vdash m + n_2 + m_2\}$$

$$f\{m_1 \dashv g_2 \vdash m + 1 + m_2\}\{m_1 + n_2 + m \dashv g_1 \vdash m_2\} \equiv f\{m_1 + 1 + m \dashv g_1 \vdash m_2\}\{m_1 \dashv g_2 \vdash m + n_1 + m_2\}$$

A morphism f is central *if it commutes with every morphism in $\mathbb{C}$. A preoperad $\mathbb{C}$ is an* operad *if every morphism in $\mathbb{C}$ is central.*

If $f \in \mathbb{C}(k)$ and $g_1, \ldots, g_k$ are central, iterated single substitution is independent of insertion order, so we write $f\{g_1 + \cdots + g_k\}$ for the resulting parallel composite.

Cartesian operads are operads equipped with renamings and the usual structural maps for exchange, discarding, and copying, satisfying the standard naturality axioms.

Definition 3.7 (Cartesian Operad) *An operad $\mathbb{V}$ is said to be* cartesian *when it is* **Ren***-cartesian, symmetric, and discarding ($!$) and copying (Δ) are natural in the following sense:*

(i) *For $f \in \mathbb{V}(m_1 + m_2)$ and $g \in \mathbb{V}(n)$:* $f[\mathsf{id}_{m_1} + !_1 + \mathsf{id}_{m_2}]\{m_1 \dashv g \vdash m_2\} \equiv f[\mathsf{id}_{m_1} + !_n + \mathsf{id}_{m_2}]$

(ii) *For $f \in \mathbb{V}(m_1 + 2 + m_2)$ and $g \in \mathbb{V}(n)$:*
$f[\mathsf{id}_{m_1} + \Delta_1 + \mathsf{id}_{m_2}]\{m_1 \dashv g \vdash m_2\} \equiv f\{m_1 \dashv g + g \vdash m_2\}[\mathsf{id}_{m_1} + \Delta_n + \mathsf{id}_{m_2}]$

For the semantics of $\lambda_{\mathrm{ml}*}$, we need two preoperads, one to interpret values and simultanous substitutions, and the other to interpret computations and their binding structure given by the $\mathsf{let} - \Leftarrow - \mathsf{in} -$ form. The preoperad designated for values is a cartesian operad, to enable the parallel, non-linear structure of simultaneous substitutions. The preoperad designated for computations is symmetric **Ren**-cartesian, to enable the appropriate interaction of reindexing with the $\mathsf{let} - \Leftarrow - \mathsf{in} -$ form. The return form relates the two, and essentially embeds the structure of values within the syntax of computations. This embedding is enabled through a preoperad functor between the two preoperads, preserving centrality and the cartesian structure. Crucially, the image of this functor is required to be central, expressing that pure computations commute with sequencing. These three components together are called *Freyd operad*.

Definition 3.8 (Freyd Operad) *A Freyd operad is a triple $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ where $\mathbb{V}$ is a cartesian operad, $\mathbb{C}$ is a symmetric* **Ren***-cartesian preoperad, $\mathcal{J} : \mathbb{V} \longrightarrow \mathbb{C}$ is a cartesian preoperad functor, and the image of $\mathcal{J}$ is a cartesian operad.*

In the term model of $\lambda_{\mathrm{ml}*}$, the Freyd functor $\mathcal{J} : \mathbb{V} \to \mathbb{C}$ is interpreted by the constructor $\mathsf{return} -$. Centrality of its image expresses that pure computations commute with sequencing. Thus inserting $\mathsf{return}\ V$ into a surrounding let-context does not constrain evaluation order. Concretely, for any computations M and computation context $K[-]$, the two programs obtained by evaluating $\mathsf{return}\ V$ before or after M are identified by the equational theory:

$$K\big[\mathsf{let}\ x \Leftarrow M\ \mathsf{in}\ \mathsf{let}\ y \Leftarrow \mathsf{return}\ V\ \mathsf{in}\ M'\big] \ \equiv\ K\big[\mathsf{let}\ y \Leftarrow \mathsf{return}\ V\ \mathsf{in}\ \mathsf{let}\ x \Leftarrow M\ \mathsf{in}\ M'\big],$$

Definition 3.9 (Freyd Operad Functor) *Given Freyd operads $(\mathbb{V}_1, \mathbb{C}_1, \mathcal{J}_1)$ and $(\mathbb{V}_2, \mathbb{C}_2, \mathcal{J}_2)$, a Freyd operads functor $\mathcal{G} : \mathcal{J}_1 \Longrightarrow \mathcal{J}_2$ is a pair $\mathcal{G} = (\mathcal{G}^{\mathbb{V}}, \mathcal{G}^{\mathbb{C}})$ such that $\mathcal{G}^{\mathbb{V}} : \mathbb{V}_1 \longrightarrow \mathbb{V}_2$ and $\mathcal{G}^{\mathbb{C}} : \mathbb{C}_1 \longrightarrow \mathbb{C}_2$ are cartesian preoperad functors and $\mathcal{G}^{\mathbb{C}}\mathcal{J}_1 \equiv \mathcal{J}_2\mathcal{G}^{\mathbb{V}}$.*

3.3 Substitutions and the free construction

We now give the main construction of the paper: given a preoperad $\mathbb{C}$, we build a *substitution category* $\mathbf{Sub}_{\mathbb{C}}$ whose morphisms are finite sequences of single substitutions and renamings, modulo the equations of the preoperad axioms. Composition is given by concatenation. This yields a pre-PROP $\mathbf{Sub}_{\mathbb{C}}$. When V is a cartesian operad, the same construction yields a cartesian PROP $\mathbf{Sub}^{\times}_{\mathbb{V}}$. Moreover, any Freyd operad $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ induces a functor $\mathcal{J}_{\mathbf{Sub}} : \mathbf{Sub}^{\times}_{\mathbb{V}} \to \mathbf{Sub}_{\mathbb{C}}$, and therefore a Freyd PROP $\big(\mathbf{Sub}^{\times}_{\mathbb{V}}, \mathbf{Sub}_{\mathbb{C}}, \mathcal{J}_{\mathbf{Sub}}\big)$.

In the theory of multicategories, there is a standard adjunction between multicategories and monoidal categories [10], and between operads and PROPs [14]. The construction of the left adjoint freely adds a tensor product to a multicategory by taking lists of objects for objects, and lists of multicategory morphisms for category morphisms. The resulting monoidal category can be considered as a category of simultaneous substitutions over the multicategory. As pointed out in [23], this technique cannot work as-is in the case of premulticategories, since lists of morphisms do not encode the sequential order between them. While [23] specifies the appropriate forgetful functor, relating premonoidal categories with premulticategories, the explicit construction of its left adjoint is left as an open problem. While there is probably a way to resolve it meta-categorically, in a similar vein to [10], we use a more direct approach, based on the following core principle: Multicategories are representable presheaves over their category of substitutions. With that in mind, we construct the category of substitutions by taking the substitutions and renamings used to modify morphisms in a preoperad, and rather than immediately applying them on a morphism in the preoperad, we put them in a list, while omitting the morphism.

Example 3.10 Let $\mathbb{C}$ be a preoperad, $f \in \mathbb{C}(m_1 + 1 + m_2)$, $g \in \mathbb{C}(n_1 + 1 + n_2)$, and $h \in \mathbb{C}(n')$. Consider the iterated action on f obtained by first substituting g into the distinguished input of f, then substituting h into the distinguished input of g, and finally reindexing by a renaming r:

Substitution congruence generators. Rules are applied *in context*: $\Gamma\, X\, \Gamma' \cong \Gamma\, Y\, \Gamma'$ whenever well typed.

Symmetric rules:

(U$_1$)	$\{m_1 \dashv \mathsf{id} \vdash m_2\} \cong \emptyset$	$(\mathsf{id} \in \mathbb{C}(1))$
(U$_2$)	$[\mathsf{id}_m] \cong \emptyset$	$(\mathsf{id}_m \in \mathbf{Ren}(m,m))$
(A)	$\{m_1 \dashv g \vdash m_2\}\ \{m_1 + n_1 \dashv h \vdash n_2 + m_2\} \cong \{m_1 \dashv (g\{n_1 \dashv h \vdash n_2\}) \vdash m_2\}$	$(g \in \mathbb{C}(n_1{+}1{+}n_2),\ h \in \mathbb{C}(n'))$
(R)	$[r_1]\ [r_2] \cong [r_2 \circ r_1]$	$(r_1 \in \mathbf{Ren}(n,m),\ r_2 \in \mathbf{Ren}(k,n))$
(N)	$\{n_1 \dashv v \vdash n_2\}\ [r_1 + r + r_2] \cong [r_1 + \mathsf{id}_1 + r_2]\ \{m_1 \dashv v[r] \vdash m_2\}$	$(v \in \mathbb{C}(m),\ r \in \mathbf{Ren}(n,m),\ r_i \in \mathbf{Ren}(m_i,n_i))$
(S$_\ell$)	$[\mathsf{id}_{m_1} + \sigma_{1,1} + \mathsf{id}_{m_2}]\ \{m_1 \dashv g \vdash 1 + m_2\} \cong \{m_1 + 1 \dashv g \vdash m_2\}\ [\mathsf{id}_{m_1} + \sigma_{1,n} + \mathsf{id}_{m_2}]$	$(g \in \mathbb{C}(n))$
(S$_r$)	$[\mathsf{id}_{m_1} + \sigma_{1,1} + \mathsf{id}_{m_2}]\ \{m_1 + 1 \dashv g \vdash m_2\} \cong \{m_1 \dashv g \vdash 1 + m_2\}\ [\mathsf{id}_{m_1} + \sigma_{n,1} + \mathsf{id}_{m_2}]$	$(g \in \mathbb{C}(n))$

Additional cartesian rules:

(CEN$_1$)	$\{m_1 \dashv g_1 \vdash m + 1 + m_2\}\ \{m_1 + n_1 + m \dashv g_2 \vdash m_2\} \cong \{m_1 + 1 + m \dashv g_2 \vdash m_2\}\ \{m_1 \dashv g_1 \vdash m + n_2 + m_2\}$	$(g_1 \in \mathbb{V}(n_1),\ g_2 \in \mathbb{V}(n_2))$
(CEN$_2$)	$\{m_1 \dashv g_2 \vdash m + 1 + m_2\}\ \{m_1 + n_2 + m \dashv g_1 \vdash m_2\} \cong \{m_1 + 1 + m \dashv g_1 \vdash m_2\}\ \{m_1 \dashv g_2 \vdash m + n_1 + m_2\}$	$(g_1 \in \mathbb{V}(n_1),\ g_2 \in \mathbb{V}(n_2))$
(D)	$[\mathsf{id}_{m_1} + !_1 + \mathsf{id}_{m_2}]\ \{m_1 \dashv g \vdash m_2\} \cong [\mathsf{id}_{m_1} + !_n + \mathsf{id}_{m_2}]$	$(g \in \mathbb{V}(n))$
(C)	$[\mathsf{id}_{m_1} + \Delta_1 + \mathsf{id}_{m_2}]\ \{m_1 \dashv g \vdash m_2\} \cong \{m_1 \dashv (g + g) \vdash m_2\}\ [\mathsf{id}_{m_1} + \Delta_n + \mathsf{id}_{m_2}]$	$(g \in \mathbb{V}(n))$

Fig. 3. Rules for the equivalence relation on pre-substitutions.

$f\,\{m_1 \dashv g \vdash m_2\}\,\{m_1 + n_1 \dashv h \vdash n_2 + m_2\}\,[r]$. Instead of applying these operations to f, we record them as the word $s := (\,\{m_1 \dashv g \vdash m_2\}\,,\ \{m_1 + n_1 \dashv h \vdash n_2 + m_2\}\,,\ r\,)$, which remembers the *sequential order* of the two substitutions and the renaming. In the quotient defining substitutions, s is identified with $(\,\{m_1 \dashv g\,\{n_1 \dashv h \vdash n_2\} \vdash m_2\}\,,\ r\,)$ by associativity of substitutions and the unit law for renamings.

Starting from a preoperad $\mathbb{C}$, we first form words consisting of single substitutions and renamings.

Definition 3.11 (Substitutions) *Let $\mathbb{C}$ be a preoperad.*

(i) *A* single substitution *from $m_1 + n + m_2$ to $m_1 + 1 + m_2$ is a tuple (m_1, f, m_2), where $f \in \mathbb{C}(n)$, written $\{m_1 \dashv f \vdash m_2\}$. We write $\mathbf{Sub}^{\mathbf{1}}_{\mathbb{C}}(m \Rightarrow n)$ for the set of single substitutions $m \to n$.*

(ii) *A* pre-substitution *$s : m \to n$ is a finite list $s = (s_1, \ldots, s_k)$ equipped with arities $m = m_0, m_1, \ldots, m_k = n$ such that for each $i = 1, \ldots, k$ the element s_i is either a single substitution $s_i \in \mathbf{Sub}^{\mathbf{1}}_{\mathbb{C}}(m_{i-1} \Rightarrow m_i)$, or a renaming $s_i \in \mathbf{Ren}^{\mathbf{op}}(m_{i-1} \Rightarrow m_i)$. We write $\mathbf{PreSub}_C(m \Rightarrow n)$ for the set of pre-substitutions $m \to n$, and $\emptyset \in \mathbf{PreSub}_{\mathbb{C}}(n \Rightarrow n)$ for the empty list.*

Next, we define two notions of substitutions by defining equivalence relations over pre-substitutions, each of which will correspond to one of our particular preoperads of interest, and the quotient by them.

Definition 3.12 (Symmetric and Cartesian substitutions) *If $\mathbb{C}$ is a symmetric* **Ren***-cartesian preoperad, $\mathbf{Sub}_{\mathbb{C}}(m \Rightarrow n)$ is the quotient of $\mathbf{PreSub}_{\mathbb{C}}(m \Rightarrow n)$ by the symmetric rules of Fig. 3. If $\mathbb{V}$ is a cartesian operad, $\mathbf{Sub}^{\times}_{\mathbb{V}}(m \Rightarrow n)$ is the quotient of $\mathbf{PreSub}_{\mathbb{V}}(m \Rightarrow n)$ by all rules of Fig. 3.*

The congruence of Fig. 3 is designed so that the resulting quotients inherit the standard PROP structure by acting componentwise on words: composition is concatenation, and whiskering extends each step by unused variables on the left or right.

Definition 3.13 (Freyd PROP of Substitutions) *Given a Freyd operad $(\mathbb{V}, \mathbb{C}, \mathcal{J})$, define an identity-*

on-objects map $\mathcal{J}^{\mathbf{Sub}} : \mathbf{Sub}_{\mathbb{V}}^{\times} \longrightarrow \mathbf{Sub}_{\mathbb{C}}$ *by recursion on substitution words:*

$$\begin{aligned}
\mathcal{J}^{\mathbf{Sub}}\emptyset &:= \emptyset \\
\mathcal{J}^{\mathbf{Sub}}\left([r], s\right) &:= \left([r], \mathcal{J}^{\mathbf{Sub}} s\right) \\
\mathcal{J}^{\mathbf{Sub}}\left(\{m_1 \dashv v \vdash m_2\}, s\right) &:= \left(\{m_1 \dashv \mathcal{J}v \vdash m_2\}, \mathcal{J}^{\mathbf{Sub}} s\right)
\end{aligned}$$

The functoriality of $\mathcal{J}$*, together with the defining equations of* $\mathbb{V}$ *and* $\mathbb{C}$*, ensures that* $\mathcal{J}^{\mathbf{Sub}}$ *preserves the equivalence relation.*

Proposition 3.14 *The followings hold.*

(i) *If* $\mathbb{C}$ *is a symmetric* **Ren***-cartesian preoperad, then* $\mathbf{Sub}_{\mathbb{C}}$ *is a pre-PROP.*

(ii) *If* $\mathbb{V}$ *is a cartesian operad, then* $\mathbf{Sub}_{\mathbb{V}}^{\times}$ *is a cartesian PROP.*

(iii) *If* $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ *is a Freyd operad, then* $\left(\mathbf{Sub}_{\mathbb{V}}^{\times}, \mathbf{Sub}_{\mathbb{C}}, \mathcal{J}^{\mathbf{Sub}}\right)$ *is a Freyd PROP.*

(iv) *Given a Freyd operad functor* $(\mathcal{G}^{\mathbb{V}}, \mathcal{G}^{\mathbb{C}}) : (\mathbb{V}_1, \mathbb{C}_1, \mathcal{J}_1) \longrightarrow (\mathbb{V}_2, \mathbb{C}_2, \mathcal{J}_2)$*, there is an induced Freyd PROP functor* $(\mathcal{G}^{\mathbf{Sub}^{\times}}, \mathcal{G}^{\mathbf{Sub}}) : \left(\mathbf{Sub}_{\mathbb{V}_1}^{\times}, \mathbf{Sub}_{\mathbb{C}_1}, \mathcal{J}_1^{\mathbf{Sub}}\right) \longrightarrow \left(\mathbf{Sub}_{\mathbb{V}_2}^{\times}, \mathbf{Sub}_{\mathbb{C}_2}, \mathcal{J}_2^{\mathbf{Sub}}\right)$*, obtained by applying* $\mathcal{G}^{\mathbb{V}}$ *and* $\mathcal{G}^{\mathbb{C}}$ *to each substitution step and leaving renamings unchanged.*

Proof.

(i) The rules of Fig. 3 define a congruence on pre-substitutions, so concatenation descends to a well-defined associative composition with unit $\emptyset$. Whiskering is defined componentwise on steps, and each generating rule is stable under whiskering, so tensor is well-defined on the quotient. The symmetry is represented by the singleton renaming word $[\sigma_{m,n}]$, and its naturality follows from the renaming/substitution interaction and the symmetry rules.

(ii) The additional cartesian rules force centrality of substitution steps and the naturality of copying and discarding. The copy and discard maps are represented by the singleton renaming words $[\Delta_n]$ and $[!_n]$, and the comonoid laws are inherited from $\mathbf{Ren}^{\mathbf{op}}$.

(iii) The map $\mathcal{J}^{\mathbf{Sub}}$ is well defined because $\mathcal{J}$ preserves renamings and the generating equations. Its image is central by the Freyd operad axioms, hence it defines a Freyd functor.

(iv) The induced maps lists respect the quotients since $\mathcal{G}^{\mathbb{V}}$ and $\mathcal{G}^{\mathbb{C}}$ preserve the relevant operadic structure and renamings. Compatibility with $\mathcal{J}^{\mathbf{Sub}}$ follows from the Freyd-operad functor axioms. □

3.4 Representability

The substitution categories of Sec. 3.3 are designed so that arity families of a preoperad can be recovered as morphisms with codomain 1. Concretely, a word $s : m \to n$ in $\mathbf{Sub}_{\mathbb{C}}$ should act on any n-ary operation $f \in \mathbb{C}(n)$ by producing an m-ary operation obtained by performing the indicated substitutions and renamings in sequence. This yields a contravariant action $\mathbb{C}(n) \to \mathbb{C}(m)$, hence a presheaf $\mathbb{C} : \mathbf{Sub}_{\mathbb{C}}{}^{\mathrm{op}} \to \mathbf{Set}$, and we show that this presheaf is representable by the implicit generating object 1.

Definition 3.15 (Pre-Substitution Application) *Let* $\mathbb{C}$ *be a symmetric* **Ren***-cartesian preoperad,* $s \in \mathbf{PreSub}_{\mathbb{C}}(m \Rightarrow n)$*, and* $f \in \mathbb{C}(n)$*. Define the* application $f \star s \in \mathbb{C}(m)$*, by recursion on* s*:*

$$f \star \emptyset := f \quad , \quad f \star \left(\{m_1 \dashv g \vdash m_2\}, s\right) := f\{m_1 \dashv g \vdash m_2\} \star s \quad , \quad f \star (r, s) := f[r] \star s$$

Theorem 3.16 *The followings hold.*

(i) *If* $\mathbb{C}$ *is a symmetric* **Ren***-cartesian preoperad, and* $s_1 \cong s_2 \in \mathbf{Sub}_{\mathbb{C}}(m \Rightarrow n)$ *then for all* $f \in \mathbb{C}(n)$*:* $f \star s_1 \equiv f \star s_2 \in \mathbb{C}(m)$*.*

(ii) *If* $\mathbb{V}$ *is a cartesian operad, and* $s_1 \cong s_2 \in \mathbf{Sub}_{\mathbb{V}}^{\times}(m \Rightarrow n)$ *then for all* $v \in \mathbb{V}(n)$*:* $v \star s_1 \equiv v \star s_2 \in \mathbb{V}(m)$*.*

Proof. For (i), it suffices to check each generating rule of Fig. 3. Each rule is exactly one of the axioms of a symmetric **Ren**-cartesian preoperad (units, associativity, renaming functoriality, renaming-substitution

interchange, and symmetry naturality). For (ii), the additional cartesian rules correspond to centrality and to the naturality axioms for copying and discarding in a cartesian operad. □

The implication in Thm. 3.16 is one-directional and it states that our congruence is *sound*: equivalent words act identically on every operation of $\mathbb{C}$, so that application descends to the quotient. The converse need not hold for a fixed preoperad $\mathbb{C}$, since $\mathbb{C}$ may satisfy additional equations beyond those enforced by the defining axioms, which may identify more words extensionally. Thus the congruence should be understood as the least one generated by the structural axioms in Fig. 3.

Corollary 3.17 *For symmetric $\mathbb{C}$, the assignment $n \mapsto \mathbb{C}(n)$ extends to a functor $\mathbb{C} : \mathbf{Sub}_{\mathbb{C}}^{\mathrm{op}} \to \mathbf{Set}$ by $\mathbb{C}(s)(f) = f \star s$. Similarly, for cartesian $\mathbb{V}$ we obtain $V : \mathbf{Sub}_{\mathbb{V}}^{\times\,\mathrm{op}} \to \mathbf{Set}$.*

Theorem 3.18 *Let $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ be a Freyd operad.*

(i) *The presheaf $\mathbb{V} : \mathbf{Sub}_{\mathbb{V}}^{\times\,\mathrm{op}} \to \mathbf{Set}$ is representable by 1, with $\mathbb{V}(n) \cong \mathbf{Sub}_{\mathbb{V}}^{\times}(n \Rightarrow 1)$.*

(ii) *The presheaf $\mathbb{C} : \mathbf{Sub}_{\mathbb{C}}^{\mathrm{op}} \to \mathbf{Set}$ is representable by 1, with $\mathbb{C}(n) \cong \mathbf{Sub}_{\mathbb{C}}(n \Rightarrow 1)$.*

(iii) *The map $\mathcal{J} : \mathbb{V} \to \mathbb{C}$ is a natural transformation $\mathbb{V} \Rightarrow \mathbb{C}\mathcal{J}^{\mathbf{Sub}}$.*

Proof.

(i) The action of both categories is given by pre-substitution application. Thm. 3.16 ensures that it is well-defined, as it preserves the equivalence relations defining the appropriate categories. Functoriality is given by construction, since $f \star \emptyset = f$ and induction on s_1 verifies that $f \star (s_1, s_2) \equiv (f \star s_1) \star s_2$. To prove representability for $\mathbb{V}$, we go from $\mathbb{V}(n)$ to $\mathbf{Sub}_{\mathbb{V}}^{\times}(n \Rightarrow 1)$ by taking each $v \in \mathbb{V}(n)$ to $(\{0 \dashv v \vdash 0\}) \in \mathbf{Sub}_{\mathbb{V}}^{\times}(n \Rightarrow 1)$, and in the other direction by taking $s \in \mathbf{Sub}_{\mathbb{V}}^{\times}(n \Rightarrow 1)$ to $\mathsf{id} \star s \in \mathbb{V}(n)$. From $\mathsf{id} \star (\{0 \dashv v \vdash 0\}) \equiv v$ we get one part of the isomorphism, and by induction on s, we get $(\{0 \dashv \mathsf{id} \star s \vdash 0\}) \cong s$, which gives us the other part of the isomorphism.

(ii) The same method is used to prove representability for $\mathbb{C}$.

(iii) For naturality of $\mathcal{J}$, we need to prove that for every $m, n \in \mathbb{N}$, $s \in \mathbf{Sub}_{\mathbb{V}}^{\times}(m \Rightarrow n)$, and $v \in \mathbb{V}(n)$, we get $\mathcal{J}v \star \mathcal{J}^{\mathbf{Sub}} s \equiv \mathcal{J}(v \star s)$. This is shown by induction on s, using the functoriality of $\mathcal{J}$. □

Representability yields natural bijections $\mathbb{C}(n) \cong \mathbf{Sub}_{\mathbb{C}}(n \Rightarrow 1)$ and $\mathbb{V}(n) \cong \mathbf{Sub}_{\mathbb{V}}^{\times}(n \Rightarrow 1)$. Thus a substitution PROP recovers its underlying (Freyd) operad by restriction to codomain 1.

3.5 The Freyd adjunction

We now prove the universal property of the substitution construction. The representability result (Thm. 3.18), packages restriction to codomain 1 as a functor $U : \mathbf{FrPROP} \to \mathbf{FrOp}$, and we show that the free substitution construction $F : \mathbf{FrOp} \to \mathbf{FrPROP}$ is left adjoint to U. We use **FrPROP** to denote the category of (small) Freyd PROP categories, with Freyd PROP functors between them. We use **FrOp** to denote the category of (small) Freyd operads, with Freyd operad functors between them. We get the following result:

Definition 3.19 (The forgetful functor $\mathcal{U}$) *Define the functor $\mathcal{U} : \mathbf{FrPROP} \to \mathbf{FrOp}$ as follows.*

On objects, $\mathcal{U}$ sends a Freyd PROP $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ to the Freyd operad $(\mathbb{V}^{\mathcal{U}}, \mathbb{C}^{\mathcal{U}}, \mathcal{J}^{\mathcal{U}})$ where

$$\mathbb{V}^{\mathcal{U}}(n) := \mathbb{V}(n \Rightarrow 1) \qquad \mathbb{C}^{\mathcal{U}}(n) := \mathbb{C}(n \Rightarrow 1)$$

*and $\mathcal{J}^{\mathcal{U}}$ is the restriction of $\mathcal{J}$ to morphisms with codomain 1. The operadic identity is $\mathsf{id}_1 \in \mathbb{V}(1 \Rightarrow 1)$ (and similarly in $\mathbb{C}$), and the operadic substitution is induced by whiskering and composition in the PROP: for $f \in \mathbb{V}(n_1 + 1 + n_2 \Rightarrow 1)$ and $g \in \mathbb{V}(k \Rightarrow 1)$, $f\{n_1 \dashv g \vdash n_2\} := f \circ (n_1 \dashv g \vdash n_2)$, and similarly in $\mathbb{C}$. Renamings in $\mathbb{V}^{\mathcal{U}}$ are interpreted in $\mathbb{V}$ via Lemma 2.2 and the **Ren**-cartesian structure is given by: $v[r] := v \circ r$ In $\mathbb{C}^{\mathcal{U}}$, renamings are the same renamings as in $\mathbb{V}^{\mathcal{U}}$, but with: $f[r] := f \circ \mathcal{J}r$.*

On morphisms, $\mathcal{U}$ sends a Freyd PROP functor $(\mathcal{G}^{\mathbb{V}}, \mathcal{G}^{\mathbb{C}})$ to the Freyd operad functor $(\mathcal{U}\mathcal{G}^{\mathbb{V}}, \mathcal{U}\mathcal{G}^{\mathbb{C}})$ obtained by restricting $\mathcal{G}^{\mathbb{V}}$ and $\mathcal{G}^{\mathbb{C}}$ to morphisms with codomain 1.

Definition 3.20 (The free functor $\mathcal{F}$) *Define the functor* $\mathcal{F} : \mathbf{FrOp} \to \mathbf{FrPROP}$ *as follows.*

On objects, $\mathcal{F}$ *sends a Freyd operad* $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ *to the Freyd PROP* $\mathcal{F}(\mathbb{V}, \mathbb{C}, \mathcal{J}) := (\mathbf{Sub}_{\mathbb{V}}^{\times}, \mathbf{Sub}_{\mathbb{C}}, \mathcal{J}^{\mathbf{Sub}})$*, as in Def. 3.13. On morphisms,* $\mathcal{F}$ *sends a Freyd operad functor* $(\mathcal{G}^{\mathbb{V}}, \mathcal{G}^{\mathbb{C}})$ *to the induced Freyd PROP functor* $(\mathcal{G}^{\mathbf{Sub}_{\mathbb{V}}^{\times}}, \mathcal{G}^{\mathbf{Sub}_{\mathbb{C}}})$ *from Thm. 3.14.*

Theorem 3.21 (The Freyd Adjunction) *There is an adjunction* $\mathcal{F} \dashv \mathcal{U}$ *where*

$$\mathcal{U} : \mathbf{FrPROP} \longrightarrow \mathbf{FrOp} \qquad \textit{and} \qquad \mathcal{F} : \mathbf{FrOp} \longrightarrow \mathbf{FrPROP}.$$

Proof. Given a Freyd operad $\mathbb{M}$ and Freyd PROP $\mathbb{D}$, we define the following isomorphism between $\mathbf{FrPROP}(\mathcal{F}(\mathbb{M}) \Rightarrow \mathbb{D})$ and $\mathbf{FrOp}(\mathbb{M} \Rightarrow \mathcal{U}(\mathbb{D}))$. One direction is obtained by restricting a Freyd PROP functor to single substitutions, and the other by interpreting substitution lists in the target Freyd PROP. These constructions are mutually inverse and natural in both variables, hence determine a natural bijection of hom-sets. Therefore $\mathcal{F} \dashv \mathcal{U}$. □

4 Interpretation in Freyd operads

In this section we interpret $\lambda_{\mathrm{ml}*}$ in Freyd operads, prove soundness, and construct the canonical term model. A context of length n is interpreted as the arity n, values as morphisms in $\mathbb{V}(n)$, and computations as morphisms in $\mathbb{C}(n)$. The first-order fragment is interpreted in any Freyd operad, but to interpret abstraction and application we additionally require a weak closure structure. This is the untyped, intensional analogue of the usual closed structure used for higher-order categorical semantics, presented through a representation of an appropriate left module following [26]. Our weak closure validates beta reduction but does not impose eta laws. To model eta, it suffices to add the axiom $\triangleleft^n \circledast \{\mathcal{J}v \vdash 1\} \triangleright = v$, recovering the standard notion of function space from [26]. Since we work in an untyped setting, there is no explicit representing object, but the construction is otherwise the same.

4.1 Models

We first define the semantic notion of model for $\lambda_{\mathrm{ml}*}$.

Definition 4.1 (Weakly Closed Freyd Operad) *A Freyd operad* $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ *is* weakly closed *if it is equipped with a transformation* $\triangleleft^n - \triangleright : \mathbb{C}(n+1) \to \mathbb{V}(n)$ *of* abstraction *and a morphism* $\circledast \in \mathbb{C}(2)$ *of* application*, such that for every* f*, the following holds (when it is defined):*

$$\circledast \{\mathcal{J} \triangleleft^n f \triangleright \vdash 1\} = f \qquad \triangleleft^{m_1+n+m_2} f \{m_1 \dashv \mathcal{J}v \vdash m_2 + 1\} \triangleright = \triangleleft^{m_1+1+m_2} f \triangleright \{m_1 \dashv v \vdash m_2\}$$

Thus abstraction is a weak representation of the left $\mathbb{V}$-module $\mathbb{C}(n+1)$, with $\circledast$ providing the corresponding counit.

Definition 4.2 (Closed Freyd Operad Functor) *A Freyd operad functor between two weakly closed Freyd operads* $\mathcal{G} : (\mathbb{V}_1, \mathbb{C}_1, \mathcal{J}_1) \longrightarrow (\mathbb{V}_2, \mathbb{C}_2, \mathcal{J}_2)$ *is* closed *when it preserves abstraction and application:* $\mathcal{G}\circledast = \circledast$*, and* $\mathcal{G} \triangleleft^n f \triangleright = \triangleleft^n \mathcal{G} f \triangleright$*.*

Definition 4.3 ($\lambda_{\mathrm{ml}*}$-Structure) *Given a signature* $\Sigma = (\mathsf{Func}, \mathsf{Proc})$*, a* λ_{ml*}*-structure for* Σ *is a tuple* $\mathcal{S} = (\mathbb{V}, \mathbb{C}, \mathcal{J}, \sigma^F, \sigma^P)$ *where* $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ *is a weakly closed Freyd operad,* σ^F *assigns to each* $f \in \mathsf{Func}(n)$ *a morphism* $\sigma^F(f) \in \mathbb{V}(n)$*, and* σ^P *assigns to each* $p \in \mathsf{Proc}(n)$ *a morphism* $\sigma^P(p) \in \mathbb{C}(n)$*. A morphism* $\mathcal{G} : \mathcal{S}_1 \to \mathcal{S}_2$ *of* λ_{ml*}*-structures is a closed Freyd operad functor between the underlying weakly closed Freyd operads that also preserves the symbol interpretations:* $\mathcal{G}(\sigma^F(f)) = \sigma^F(f)$ *and* $\mathcal{G}(\sigma^P(p)) = \sigma^P(p)$*.*

We will later construct a canonical *term* λ_{ml*}*-structure* from $\lambda_{\mathrm{ml}*}$ terms modulo the equational theory and show that it is initial.

4.2 Interpretation and Soundness

Fix a signature $\Sigma = (\mathsf{Func}, \mathsf{Proc})$ and $\lambda_{\mathrm{ml}*}$-structure $\mathcal{S} = (\mathbb{V}, \mathbb{C}, \mathcal{J}, \sigma^F, \sigma^P)$ over Σ. We write $|\Gamma|$ for the length of a context, and we freely use concatenation notation $\Gamma = \Gamma_1, \ldots, \Gamma_k$ when a term is formed from

subterms in the corresponding subcontexts. The interpretation follows the value/computation split of the syntax: variables and function symbols are interpreted in the cartesian value operad, computations in the computation preoperad, the let-constructor by single substitution in $\mathbb{C}$, and abstraction and application by the weak closure structure.

Definition 4.4 (Interpretation) *Given a λ_{ml*}-structure $\mathcal{S}$, we define two families of interpretation functions, $[\![-]\!]^V_\Gamma : \mathbf{VAL}(\Gamma) \to \mathbb{V}(|\Gamma|)$ and $[\![-]\!]^C_\Gamma : \mathbf{CMP}(\Gamma) \to \mathbb{C}(|\Gamma|)$, by mutual recursion:*

$$\begin{array}{lcl}
[\![x_k]\!]^V_{x_1,\ldots,x_n} & := & \mathsf{id}\,[!_{k-1} + \mathsf{id}_1 + !_{n-k}] \\
[\![f(V_1,\ldots,V_k)]\!]^V_{\Gamma_1,\ldots,\Gamma_k} & := & \sigma^F(f)\left\{[\![V_1]\!]^V_{\Gamma_1} + \ldots + [\![V_k]\!]^V_{\Gamma_k}\right\} \\
[\![\lambda x\,.\,M]\!]^V_\Gamma & := & \triangleleft^{\Pi}\,[\![M]\!]^C_{\Gamma,x}\,\triangleright \\
[\![\mathsf{return}\ V]\!]^C_\Gamma & := & \mathcal{J}[\![V]\!]^V_\Gamma \\
[\![\mathsf{let}\ x \Leftarrow M_2\ \mathsf{in}\ M_1]\!]^C_{\Gamma_1,\Gamma_2} & := & [\![M_1]\!]^C_{\Gamma_1,x}\left\{|\Gamma_1| \dashv [\![M_2]\!]^C_{\Gamma_2}\right\} \\
[\![p(V_1,\ldots,V_k)]\!]^C_{\Gamma_1,\ldots,\Gamma_k} & := & \sigma^P(p)\left\{\mathcal{J}[\![V_1]\!]^V_{\Gamma_1} + \ldots + \mathcal{J}[\![V_k]\!]^V_{\Gamma_k}\right\} \\
[\![V_1\,V_2]\!]^C_{\Gamma_1,\Gamma_2} & := & \circledast\left\{\mathcal{J}[\![V_1]\!]^V_{\Gamma_1} + \mathcal{J}[\![V_2]\!]^V_{\Gamma_2}\right\}
\end{array}$$

We next prove the fundamental lemma, namely, that interpretation preserves substitution.

Lemma 4.5 *Given a strucutre $\mathcal{S}$, for any contexts $\Gamma_1,\ldots,\Gamma_n$ and values $V_1,\ldots,V_n$ s.t $\Gamma_i \vdash_{\boldsymbol{v}} V_i$, the followings hold for any V s.t $x_1,\ldots,x_n \vdash_{\boldsymbol{v}} V$ and any M s.t $x_1,\ldots,x_n \vdash_{\boldsymbol{c}} M$:*

$$\begin{array}{l}
[\![V\{V_1/x_1,\ldots,V_n/x_n\}]\!]^V_{\Gamma_1,\ldots,\Gamma_n} \equiv [\![V]\!]^V_{x_1,\ldots,x_n}\left\{[\![V_1]\!]^V_{\Gamma_1} + \ldots + [\![V_n]\!]^V_{\Gamma_n}\right\} \\
[\![M\{V_1/x_1,\ldots,V_n/x_n\}]\!]^C_{\Gamma_1,\ldots,\Gamma_n} \equiv [\![M]\!]^C_{x_1,\ldots,x_n}\left\{\mathcal{J}[\![V_1]\!]^V_{\Gamma_1} + \ldots + \mathcal{J}[\![V_n]\!]^V_{\Gamma_n}\right\}
\end{array}$$

Definition 4.6 (Satisfiability) *Let $\mathcal{S}$ be a λ_{ml*}-structure. For well formed terms E_1, E_2 of the same sort in a context Γ, we write $\mathcal{S} \Vdash E_1 \equiv_\Gamma E_2$ when their interpretations are equal in the corresponding semantic sort, i.e., in $\mathbb{V}(|\Gamma|)$ for values and in $\mathbb{C}(|\Gamma|)$ for computations. We write $\mathcal{S} \Vdash_{\boldsymbol{v}} V_1 \equiv_\Gamma V_2$ and $\mathcal{S} \Vdash_{\boldsymbol{c}} M_1 \equiv_\Gamma M_2$ for value and computation, respectively. We write $\Vdash E_1 \equiv_\Gamma E_2$ when $\mathcal{S} \Vdash E_1 \equiv_\Gamma E_2$ holds for every λ_{ml*}-structure $\mathcal{S}$, and $\Vdash_{\boldsymbol{v}} V_1 \equiv_\Gamma V_2$ and $\Vdash_{\boldsymbol{c}} M_1 \equiv_\Gamma M_2$ for the two sorts, respectively.*

We now show that the equational theory of $\lambda_{\mathrm{ml}*}$ is respected by the interpretation. Intuitively, each equation in Fig. 2 expresses one of the defining laws of a (weakly closed) Freyd operad: the sequencing rules correspond to the preoperad substitution laws in $\mathbb{C}$, and beta corresponds to weak closure. Well formedness assumptions play no semantic role beyond ensuring that the relevant interpretations are defined.

Theorem 4.7 (Soundness) *Given well-formed terms E_1, E_2 in Γ: $\Gamma \vdash E_1 \equiv E_2 \implies \Vdash E_1 \equiv_\Gamma E_2$.*

Proof. By induction on the derivation of $\Gamma \vdash E_1 \equiv E_2$. The structural cases are immediate, and β follows from weak closure. For *lunit*, the interpretation of $\mathsf{let}\ x \Leftarrow \mathsf{return}\ V\ \mathsf{in}\ M$ is $[\![M]\!]^C_{\Gamma_1,x}\left\{|\Gamma_1| \dashv \mathcal{J}[\![V]\!]^V_{\Gamma_2}\right\}$, which is the interpretation of the right hand side by Lemma 4.5. The *runit* case is exactly the right unit law of the preoperad, and *assoc* is exactly the associativity law of substitution in the preoperad. □

4.3 Term model and Initiality

We now construct the canonical *term model* of $\lambda_{\mathrm{ml}*}$. It packages the syntax modulo the equational theory of Fig. 2 into a weakly closed Freyd operad, and it enjoys the expected universal property. As usual, initiality yields completeness as an immediate corollary.

For each $n \in \mathbb{N}$, let $\mathbf{VAL}(n)$ be the set of equivalence classes $[V]$ of well formed values $x_1,\ldots,x_n \vdash_{\mathbf{v}} V$, modulo provable value equality in Fig. 2. Similarly, let $\mathbf{CMP}(n)$ be the set of equivalence classes $[M]$ of well formed computations $x_1,\ldots,x_n \vdash_{\mathbf{c}} M$, modulo provable computation equality. We write $\mathbf{VAL} := (\mathbf{VAL}(n))_{n\in\mathbb{N}}$ and $\mathbf{CMP} := (\mathbf{CMP}(n))_{n\in\mathbb{N}}$.

The operadic structures are induced on representatives: for values, $[V_1]\{n_1 \dashv [V_2] \vdash n_2\} := [V_1\{V_2/y\}]$, and for computations, $[M_1]\{n_1 \dashv [M_2] \vdash n_2\} := [\mathsf{let}\ y \Leftarrow M_2\ \mathsf{in}\ M_1]$, where y is the distinguished variable in the middle position. Renamings are induced by syntactic renaming on representatives. These operations are well defined by the congruence rules and the substitution laws of the syntax. The return map is induced by the constructor $\mathsf{return}\ -$, namely $\mathsf{return}([V]) := [\mathsf{return}\ V]$. The interpretations of primitive symbols are given by their canonical terms: $\sigma^F(f) := [f(x_1, \ldots, x_n)] \in \mathbf{VAL}(n)$ and $\sigma^P(p) := [p(x_1, \ldots, x_n)] \in \mathbf{CMP}(n)$. Finally, weak closure is induced by the term formers for application and abstraction: $\circledast := [x_1\, x_2] \in \mathbf{CMP}(2)$ and, for $x_1, \ldots, x_n, x \vdash_{\mathbf{c}} M$, $\triangleleft^n[M]\triangleright := [\lambda x\,.\,M] \in \mathbf{VAL}(n)$.

Theorem 4.8 $(\mathbf{VAL}, \mathbf{CMP}, \mathsf{return}, \mathsf{Func}, \mathsf{Proc})$ *is a* λ_{ml*}*-structure.*

Proof. The operad and preoperad laws for **VAL** and **CMP** follow from the corresponding substitution laws of the syntax, together with the exchange and renaming rules. That return is a Freyd operad functor is witnessed by the let $\Leftarrow$ in unit and associativity equations, and centrality and cartesian structure are witnessed by the structural rules in Fig. 2. Weak closure holds because the equations of Def. 4.3 are exactly the beta axiom and the compatibility of abstraction with substitution, both provable in the theory. Finally, the symbol assignments are well formed by construction and satisfy no further axioms. □

Let $\mathbf{FrOp}_{\mathcal{S}}$ be the category whose objects are $\lambda_{\mathrm{ml}*}$-structures over the fixed signature Σ and whose morphisms are Freyd operad functors preserving the interpretations of function and procedure symbols.

Theorem 4.9 (Initiality) *The term model* $(\mathbf{VAL}, \mathbf{CMP}, \mathsf{return}, \mathsf{Func}, \mathsf{Proc})$ *is initial in* $\mathbf{FrOp}_{\mathcal{S}}$.

Proof. Let $(\mathbb{V}, \mathbb{C}, \mathcal{J}, \sigma^F, \sigma^P)$ be a $\lambda_{\mathrm{ml}*}$-structure. First we show that the interpretation function given in Def. 4.4 forms a Freyd operad functor. Identity is preserved by sending $x \vdash_{\mathbf{v}} x$ to $\mathsf{id}\,[\mathsf{id}_1] \equiv \mathsf{id}$ in $\mathbb{V}(1)$ and $x \vdash_{\mathbf{c}} \mathcal{J}x$ to $\mathcal{J}(\mathsf{id}) \equiv \mathsf{id}$ in $\mathbb{C}(1)$. Composition of values is preserved by Lem 4.5. Composition of computations is preserved by the interaction of the $[\![\mathsf{let}\ x \Leftarrow M_2\ \mathsf{in}\ M_1]\!]^C_{\Gamma_1,\Gamma_2}$ case with the exchange rule. Renamings for both values and computations are preserved by Lem 4.5. The $\mathcal{J}$ functor is preserved by definition. For uniqueness, let $\mathcal{G} : (\mathbf{VAL}, \mathbf{CMP}, \mathsf{return}, \mathsf{Func}, \mathsf{Proc}) \longrightarrow (\mathbb{V}, \mathbb{C}, \mathcal{J}, \sigma^F, \sigma^P)$ be a Freyd operad functor. Given any $V \in \mathbf{VAL}(n)$ or $M \in \mathbf{CMP}(n)$, mutual induction on V and M shows that $\mathcal{G}^{\mathbb{V}}(V)$ and $\mathcal{G}^{\mathbb{C}}(M)$ are uniquely determined by V and M, since $\mathcal{G}$ preserves the Freyd operad structure. □

If an equation is valid, it is satisfied in particular by the term model. But equality in the term model is, by definition, provable equality in the equational theory, which obtains completeness.

Corollary 4.10 (Completeness) *Given well-formed terms* E_1, E_2 *in* Γ*:* $\Vdash E_1 \equiv_\Gamma E_2 \implies \Gamma \vdash E_1 \equiv E_2$.

Now we finally get to the categorical semantics of $\lambda_{\mathrm{ml}*}$. But rather than developing them independently, they arise immediately from the adjunction, and can equivalently be given directly in a Freyd PROP. Every weakly closed Freyd PROP equipped with interpretations of the symbols of Σ induces a $\lambda_{\mathrm{ml}*}$-structure by restriction to morphisms with codomain 1. Therefore the interpretation, soundness, and completeness results above apply verbatim in the PROP setting. Combining this observation with the adjunction $F \dashv U$ and the initiality of the term model yields the corresponding PROP-level universal property.

5 Examples of weakly closed Freyd operads

In this section, we briefly illustrate how the semantic notion introduced above arises in several familiar settings. The first example supports our realizability motivation; the others demonstrate the structure in standard categorical models.

Monadic Combinatory Algebras. Monadic combinatory algebras (MCAs) were introduced in [5] as a generalization of partial combinatory algebras (PCAs), an abstraction of models for the untyped lambda calculus commonly used in realizability, to the effectful setting. They provide a direct source of weakly closed Freyd operads for $\lambda_{\mathrm{ml}*}$. Let (M, μ, η) be a monad on **Set**, and let $(\mathbb{A}, \circledast)$ be a monadic applicative structure, where $\circledast : \mathbb{A}^2 \to M(\mathbb{A})$ is a Kleisli application operation. This data induces a Freyd operad $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ by taking $\mathbb{V}(n)$ to be the set of functions $\mathbb{A}^n \to \mathbb{A}$, $\mathbb{C}(n)$ to be the set of Kleisli maps $\mathbb{A}^n \to M(\mathbb{A})$, and $\mathcal{J}(f) := \eta \circ f$. The image of the first-order language in this structure is given by the usual monadic interpretation of application terms. The resulting Kleisli maps $\mathbb{A}^n \to M(\mathbb{A})$ are the

$\mathbb{A}$-*monomials*. A computation $m \in M(\mathbb{A})$ is *computable* if it is of the form $\eta(c)$ for some $c \in \mathbb{A}$. An $\mathbb{A}$-monomial $f : \mathbb{A}^{n+1} \to M(\mathbb{A})$ is *computable* if there exists a function $\lhd^n f \rhd : \mathbb{A}^n \to \mathbb{A}$ such that $\eta \circ \lhd^n f \rhd$ is computable and, for every $c \in \mathbb{A}$, application of $\lhd^n f \rhd$ to c computes $f(-, c)$. Thus, abstraction and application provide exactly the weak closure data for the Freyd operad of $\mathbb{A}$-monomials. Then the induced Freyd operad is weakly closed precisely when all $\mathbb{A}$-monomials are computable (the MCA analogue of combinatory completeness). This bridges our semantics with realizability-style models.

Applicative Systems in Cartesian Restriction Categories. Another example comes from partial combinatory algebras internal to cartesian restriction categories [4,3]. As observed in [5], every cartesian restriction category carries a Freyd categorical structure determined by its restriction. Let $\mathbb{A}$ be an applicative system in a cartesian restriction category, with application morphism $\circledast : \mathbb{A}^2 \to \mathbb{A}$. A morphism $f : \mathbb{A}^n \to \mathbb{A}$ is *computable* when there is a total point $c_f : \mathbf{1} \to \mathbb{A}$ such that $\circledast \circ (c_f \times \mathsf{id}_{\mathbb{A}^n}) = f$ and all partial applications are total. One then obtains a Freyd operad $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ by taking $\mathbb{C}(n)$ to be the morphisms $\mathbb{A}^n \to \mathbb{A}$, $\mathbb{V}(n)$ to be the total morphisms $\mathbb{A}^n \to \mathbb{A}$, and $\mathcal{J}$ to be the inclusion of total maps into all maps. In this case $\mathbb{C}$ is in fact an operad, not merely a preoperad. In [3], combinatory completeness is formulated using a category of polynomials over $\mathbb{A}$. Restricting attention to monomials suggests a closer match with the present framework: one expects a Freyd operad $(\mathbb{V}_{\mathbb{A}}, \mathbb{C}_{\mathbb{A}}, \mathcal{J}_{\mathbb{A}})$ whose computation part consists of monomials over $\mathbb{A}$ and whose value part consists of the total ones, with weak closure corresponding to combinatory completeness. This comparison is especially suggestive because the Freyd category of substitutions generated by such an operad should play the role of the polynomial category. We do not pursue this equivalence here, but it indicates that the substitutional viewpoint of Section 3 is closely related to the restriction categorical account of computability.

Reflexive Objects in a Cartesian Closed Category. Our final example recovers the usual higher-order semantics of reflexive objects [18]. Let $\mathcal{C}$ be a cartesian closed category equipped with a strong monad M, and let $\mathbb{A}$ be an object equipped with maps $e : M(\mathbb{A})^{\mathbb{A}} \to \mathbb{A}$ and $d : \mathbb{A} \to M(\mathbb{A})^{\mathbb{A}}$. If e and d form an isomorphism, then $\mathbb{A}$ is a reflexive object in the usual sense, however, if one assumes only $e \circ d = \mathsf{id}$, one obtains the weaker intensional structure corresponding to β without η. From such data one obtains a weakly closed Freyd operad $(\mathbb{V}, \mathbb{C}, \mathcal{J})$ by taking $\mathbb{V}(n) := \mathcal{C}(\mathbb{A}^n, \mathbb{A})$, $\mathbb{C}(n)$ to be the Kleisli maps $\mathbb{A}^n \to M(\mathbb{A})$, and $\mathcal{J}(f) := \eta \circ f$. Application is induced by the uncurrying of d, namely $\circledast := \lfloor d \rfloor : \mathbb{A}^2 \to M(\mathbb{A})$, and abstraction is induced by e: for $f : \mathbb{A}^{n+1} \to M(\mathbb{A})$, define $\lhd^n f \rhd := e \circ \lceil f \rceil : \mathbb{A}^n \to \mathbb{A}$ (where $\lceil f \rceil$ is the currying of f). The weak closure axioms are then exactly the categorical forms of β and substitution compatibility.

6 Conclusion

We introduced Freyd operads as a substitutional structure for effectful Call by Value computation, where computation substitution is sequential. From any Freyd operad we constructed a Freyd PROP, proved representability, and established the expected adjunction. This yields an initial semantic model for untyped computational λ calculus with procedures and higher order functions, and hence a sound and complete categorical account of its equational theory. More broadly, this shows that substitution can serve as an organizing principle for the semantics of effectful higher-order computation, opening a route toward syntactically faithful categorical frameworks with further semantic and logical applications. Although this work focuses on the untyped setting, the extension to typed systems seems straightforward. The untyped case was prioritized primarily due to the specific difficulties it introduces for achieving completeness.

There are several natural directions for future work. First, we plan to relate the present framework more directly to realizability and to evaluation logic [20], where the sequential character of computation is also fundamental. Second, it would be interesting to understand how the present Call-by-Value account interacts with Call-by-Push-Value [16], and whether Freyd operads admit a corresponding reformulation in that setting. Third, we shall extend this work to a linear variant of computational lambda calculus using non-Cartesian categories of values, and relate it to linear realizability [11,27]. Finally, we plan to lift the construction to a more genuinely metacategorical level, for example by formulating it using internal categories in the sense of [10], which may lead to a more flexible and conceptual account of substitutional semantics beyond the one-object case.

References

[1] Bonchi, F., P. Sobocinski and F. Zanasi, *Lawvere Categories as Composed PROPs*, in: *International Workshop on Coalgebraic Methods in Computer Science*, pages 11–32, Springer (2016).

[2] Burroni, A., *Higher Dimensional Word Problem*, in: *International Conference on Category Theory and Computer Science*, pages 94–105, Springer (1991).

[3] Cockett, J., *Categories and Computability*, Lecture notes available at http://pages. cpsc. ucalgary. ca/robin (2010).

[4] Cockett, J. R. B. and P. J. Hofstra, *Introduction to Turing Categories*, Annals of pure and applied logic **156**, pages 183–209 (2008).

[5] Cohen, L., A. Grunfeld, D. Kirst and É. Miquey, *From Partial to Monadic: Combinatory Algebra with Effects*, in: *FSCD 2025-10th International Conference on Formal Structures for Computation and Deduction* (2025).

[6] Cohen, L., A. Grunfeld, D. Kirst and É. Miquey, *Syntactic Effectful Realizability in Higher-Order Logic*, 2025 40th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) pages 16–30 (2025).

[7] Cohen, L., É. Miquey and R. Tate, *Evidenced Frames: A Unifying Framework Broadening Realizability Models*, in: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13 (2021).

[8] De'Liguoro, U. and R. Treglia, *The Untyped Computational λ-Calculus and its Intersection Type Discipline*, Theoretical Computer Science **846**, pages 141–159 (2020).

[9] Führmann, C., *Direct Models of the Computational Lambda-Calculus*, Electronic Notes in Theoretical Computer Science **20**, pages 245–292 (1999).

[10] Hermida, C., *Representable Multicategories*, Advances in Mathematics **151**, pages 164–225 (2000).

[11] Hoshino, N., *Linear realizability*, in: *International Workshop on Computer Science Logic*, pages 420–434, Springer (2007).

[12] Lafont, Y., *Towards an Algebraic Theory of Boolean Circuits*, Journal of Pure and Applied Algebra **184**, pages 257–310 (2003).

[13] Lawvere, F. W., *Functorial Semantics of Algebraic Theories*, Proceedings of the National Academy of Sciences **50**, pages 869–872 (1963).

[14] Leinster, T., *Higher Operads, Higher Categories*, 298, Cambridge University Press (2004).

[15] Levy, P., J. Power and H. Thielecke, *Modelling Environments in Call-by-Value Programming Languages*, Information and computation **185**, pages 182–210 (2003).

[16] Levy, P. B., *Call-by-Push-Value: A Subsuming Paradigm*, in: *International Conference on Typed Lambda Calculi and Applications*, pages 228–243, Springer (1999).

[17] Moggi, E., *Notions of Computation and Monads*, Information and computation **93**, pages 55–92 (1991).

[18] Moggi, E. *et al.*, *Computational Lambda-Calculus and Monads*, in: *Fourth Annual Symposium on Logic in Computer Science*, pages 14–23 (1989).

[19] Pirashvili, T., *On the PROP Corresponding to Bialgebras*, Cahiers de topologie et géométrie différentielle catégoriques **43**, pages 221–239 (2002).

[20] Pitts, A. M., *Evaluation Logic*, in: *IV Higher Order Workshop, Banff 1990: Proceedings of the IV Higher Order Workshop, Banff, Alberta, Canada 10–14 September 1990*, pages 162–189, Springer (1991).

[21] Pitts, A. M., *Categorical logic, Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures (S. Abramsky, DM Gabbay, and TSE Maibaum, eds.)* (2001).

[22] Power, J. and E. Robinson, *Premonoidal Categories and Notions of Computation*, Mathematical structures in computer science **7**, pages 453–468 (1997).

[23] Rajesh, N., *Universal Algebra and Effectful Computation*, arXiv preprint arXiv:2504.10314 (2025).

[24] Sabry, A. and P. Wadler, *A Reflection on Call-by-Value*, ACM transactions on programming languages and systems (TOPLAS) **19**, pages 916–941 (1997).

[25] Staton, S., *Freyd categories are Enriched Lawvere Theories*, Electronic Notes in Theoretical Computer Science **303**, pages 197–206 (2014).

[26] Staton, S. and P. B. Levy, *Universal Properties of Impure Programming Languages*, ACM SIGPLAN Notices **48**, pages 179–192 (2013).

[27] Tomita, H., *Realizability Without Symmetry*, in: *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, pages 38–1, Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2021).

[28] Yamada, R., *Effectful Toposes and Their Lawvere-Tierney Topologies*, arXiv preprint arXiv:2602.23086 (2026).