Clones, closed categories, and combinatory logic*

Philip Saville

Department of Computer Science, University of Oxford, UK
philip.saville@cs.ox.ac.uk
www.philipsaville.co.uk

Abstract. We explain how to recast the semantics of the simply-typed λ-calculus, and its linear and ordered variants, using multi-ary structures. We define universal properties for multicategories, and use these to derive familiar rules for products, tensors, and exponentials. Finally we outline how to recover both the category-theoretic syntactic model and its semantic interpretation from the multi-ary framework. We then use these ideas to study the semantic interpretation of combinatory logic and the simply-typed λ-calculus without products. We introduce *extensional SK-clones* and show these are sound and complete for both combinatory logic with extensional weak equality and the simply-typed λ-calculus without products. We then show such SK-clones are equivalent to a variant of closed categories called *SK-categories*, so the simply-typed λ-calculus without products is the internal language of SK-categories.

Keywords: categorical semantics · abstract clones · lambda calculus · combinatory logic · closed categories · cartesian closed categories

1 Introduction

Lambek's correspondence between cartesian closed categories and the simply-typed λ-calculus is one of the central pillars of categorical semantics. One way of stating it categorically is to say that the syntax of typed λ-terms over a *signature* of base types and constants forms the free cartesian closed category (for a readable overview, see [27,9]). The existence of this *syntactic model* gives *completeness*: if an equation holds in every model, it holds in the free one, and hence in the syntax. The free property then gives *soundness*: for any interpretation of basic types and constants in a cartesian closed category $(\mathcal{C}, \Pi, \Rightarrow)$ one has a functor $[\![-]\!]$ from the syntactic model to $\mathcal{C}$, which is exactly the semantic interpretation of λ-terms. The fact this functor is required to preserve cartesian closed structure amounts to showing that the semantic interpretation is sound with respect to the usual $\beta\eta$-laws. All this justifies calling the simply-typed λ-calculus the *internal language* of cartesian closed categories.

This framework is powerful, but hides a fundamental mismatch: morphisms $A \to B$ in a category are *unary*—they have just one input—but terms-in-context

* Supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0038.

such as $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ can have many inputs. The standard solution (*e.g.* [9,23]) is to use categorical products to model contexts, so a term t as above corresponds to a map $\prod_{i=1}^{n} A_i \to B$ out of the product.

Despite its evident success, this solution remains somewhat unsatisfactory, in two ways (see also [21]). First, it forces us to conflate two different syntactic classes, namely contexts and product types. As a result, some encoding is required to construct the syntactic model: the interpretation of $x : A, y : B \vdash t : C$ is a term in context $p : A \times B$. This adds complexity to the construction, and results in the somewhat unintuitive fact that the semantic interpretation of a term t in the syntactic model may not be just t itself. In turn, this complicates the proof of completeness.

Second, we are forced to include products in our type theory if we want a category-theoretic internal language—even though the calculus without products likely has a stronger claim to being called 'the' simply-typed λ-calculus (*e.g.* see Church's original definition [8]). This raises the question: what categorical structure has the simply-typed λ-calculus without products as its internal language?

This paper. This paper has three main aims. First, to explain how removing the mismatch between terms-in-context and morphisms outlined above clarifies the semantic interpretation of simply-typed λ-calculi. To achieve this, one needs to move from the unary setting of categories to a *multi-ary* setting, in which we have *multimaps* $A_1, \ldots, A_n \to B$. These ideas are not new, but are under-appreciated, and I hope this will provide self-contained introduction for a wider audience. Second, to initiate a multi-ary investigation of the semantics of (cartesian) combinatory logic, in the style of Hyland's investigation of similar ideas for the untyped λ-calculus ([18,19]). Finally, to use these results to define a categorical semantics for the simply-typed λ-calculus without products.

Outline. In Sections 2 to 6, we explain how the multi-ary perspective yields a slick way to derive the unary semantic interpretation and syntactic model, together with soundness and completeness results (Section 4.2). We also show how important type-theoretic constructions such as products and exponentials can be derived from the semantics. This framework accommodates different choices of structural rules, such as whether the language is ordered, linear, or cartesian.

The idea of using multi-ary constructions goes back to Lambek ([25,26]), and has recently been exploited to great effect in a very general setting by Shulman [40]. Particular cases can also be found in the works of Hyland ([18,19]), Hyland & de Paiva [20] and Blanco & Zeilberger [7]. A reader familiar with these approaches will likely be unsurprised by the technical development below. However, we believe these ideas deserve to be more widely known, so spend time making them explicit in a concrete setting.

In Section 7 we introduce a multi-ary model of (cartesian) combinatory logic, called *SK-clones*, and prove that the sub-category of *extensional* SK-clones is equivalent to the category of *closed clones* modelling simply-typed λ-calculus without products. This provides a categorical statement of the classical correspondence between λ-calculus and extensional combinatory logic (*e.g.* [5,15]).

Finally, in Section 8 we introduce a version of Eilenberg & Kelly's closed categories ([11,10]), called *SK-categories*, and show that the category of SK-categories is equivalent to the category of extensional SK-clones, and so to the category of closed clones. Hence, SK-categories are a categorical model for the simply-typed λ-calculus without products. SK-categories are a cartesian version of the prounital-closed categories of Uustalu, Veltri & Zeilberger ([43,44]), which in turn are closely related to an (incomplete) suggestion of Shulman's [39].

Jacobs has also isolated a structure that is sound and complete for simply-typed λ-calculus without products [21]. His approach, which fits into his elegant general framework [22], is also predicated on a careful distinction between contexts and products. His models are certain indexed categories, with the contexts encoded by the indexing: this makes them feel closer to multi-ary structures. In SK-categories, by contrast, contexts are modelled within the category itself by using the closed structure (*cf.* [35, §4.4]). Moreover, unlike other work relating closed categories to multi-ary structures, SK-categories do not force us to include a unit object in the corresponding type theory (*cf.* [31]).

Technical preliminaries. For a set S we write $S^\star$ for the set of finite sequences over S, and use Greek letters $\Gamma, \Delta, \ldots$ to denote elements of $S^\star$. The empty string is denoted $\diamond$, and the length of Γ by $|\Gamma|$. Where the length of a sequence is clear, we write simply $A_\bullet$ for $A_1, \ldots, A_n$. Contexts are assumed to be ordered lists.

We call multimaps of the form $A \to B$ *unary* and a multimap $\diamond \to B$ *nullary*.

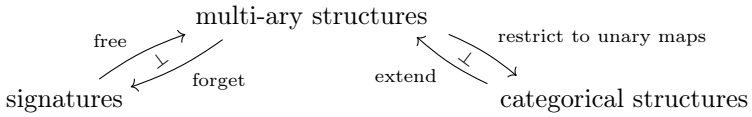
We define a *signature* $\mathcal{S}$ to be a set $|\mathcal{S}|$ of *basic sorts* with sets $\mathcal{S}(\Gamma; B)$ of *constants* $c : \Gamma \to B$ for each $\big(\Gamma, B\big) \in |\mathcal{S}|^\star \times |\mathcal{S}|$. A *homomorphism of signatures* $f : \mathcal{S} \to \mathcal{S}'$ is a map $|f| : |\mathcal{S}| \to |\mathcal{S}'|$ with maps $\mathcal{S}(A_1, \ldots, A_n; B) \to \mathcal{S}'(fA_1, \ldots, fA_n; fB)$ for each $\big((A_1, \ldots, A_n), B\big) \in |\mathcal{S}|^\star \times |\mathcal{S}|$. We write **Sig** for the category of signatures and their homomorphisms. One could also consider versions of higher-order constants, which may use the language's constructs. This extension does not change the theory significantly, and would require introducing multiple categories of signatures, so we do not seek this extra generality here (for an outline of this more general approach, see *e.g.* [38, §5.3.1]).

We assume familiarity with the simply-typed λ-calculus, as in *e.g.* [9]. We denote the simply-typed λ-calculus with constants and base types given by a signature $\mathcal{S}$, and both product and exponential types modulo $\alpha\beta\eta$-equality, by $\Lambda_{\mathcal{S}}^{\times,\to}$. We write $\Lambda_{\mathcal{S}}^{\times}$ and $\Lambda_{\mathcal{S}}^{\to}$ for the fragments with just products and just exponentials, respectively. Here we focus on the *typed* cases: the untyped versions—both in the syntax and the multi-ary models—are recovered by fixing a single base type $\star$ such that $\Theta(\star, \ldots, \star) = \star$ for each type constructor Θ.

We also assume familiarity with the basics of cartesian categories, cartesian closed categories, and monoidal categories, as in *e.g.* [30,27]. To avoid having to treat the unit type as a special case, cartesian categories are assumed to have n-ary products $\prod_n$ for all $n \in \mathbb{N}$. We also work with functors preserving structure *strictly*: this simplifies the exposition without any great loss of generality. Thus, **MonCat**, **SMonCat** and **CartCat** denote the categories of monoidal categories, symmetric monoidal categories, and cartesian categories, respectively, with functors preserving all the data on the nose.

2 Multicategories and clones

We begin with an intuitive overview of the place of multi-ary structures in semantics. A multi-ary structure has *multimaps* $A_1, \ldots, A_n \to B$ with multiple inputs and one output; unlike the morphisms in a category, multimaps correspond directly to terms-in-context. As a result, it is often easier to construct a multi-ary free model than it is to construct a unary one, and the interpretation of a term-in-context t in the free model is given by t itself. Moreover, every multi-ary structure gives rise to a unary one by restricting to multimaps with one input. The multi-ary semantics therefore factors the unary one, as shown:



(1)

One can then 'read off' the syntactic category, together with a guarantee that it has the right structure, by restricting the free multi-ary structure to unary maps. Similarly, the usual semantic interpretation in (say) a cartesian closed category $\mathcal{C}$ is exactly the interpretation that arises by extending $\mathcal{C}$ to a multi-ary structure. This gives an algebraic justification for encoding contexts as products: this is how one extends a cartesian closed category to a multi-ary structure. (For the details of these points, see Section 4.2.)

The multi-ary perspective also provides a unifying framework for type theories with different structural rules. The simply-typed λ-calculus is *cartesian*: it admits the structural rules of weakening, contraction, and permutation (as in *e.g.* [9, Fig. 3.2]). The corresponding multi-ary structures are certain *abstract clones*. *Ordered* type theories (*e.g.* [24,36]), also known as *planar* type theories (*e.g.* [2,46]), do not admit weakening, contraction, or permutation, and correspond to certain *multicategories*. *Linear* type theories (*e.g.* [16]), which admit only permutation, correspond to certain *symmetric multicategories* (see also the alternative 'tangled' option in [33]). Since abstract clones and symmetric multicategories may be seen as special cases of multicategories, we can develop a theory of how to add structure to cartesian, linear, and ordered type theories by analysing how to add structure to multicategories.

2.1 Multicategories, clones, and their internal languages

We now introduce multicategories and abstract clones and show how they correspond to certain type theories. An even more general framework for syntax, allowing multi-ary domains and codomains as well as both cartesian and linear contexts, is provided by Shulman's recent work with polycategories [40]. Clones, and their correspondence with syntax, also play a key role in the 'algebraic syntax' programme of Fiore and collaborators initiated in [13] (see *e.g.* [12,3,4]).

Definition 1 ([25]). *A* multicategory $\mathbb{M}$ *consists of a set* $|\mathbb{M}|$ *of* objects *and sets* $\mathbb{M}(\Gamma; B)$ *of* multimaps *for every* $\Gamma \in |\mathbb{M}|^\star$ *and* $B \in |\mathbb{M}|$*, together with*

1. *An* identity *multimap* $\mathrm{Id}_A \in \mathbb{M}(A;A)$ *for every* $A \in |\mathbb{M}|$*;*
2. *For any* $A_1,\dots,A_n,B \in |\mathbb{M}|$ *and* $(\Delta_i \in |\mathbb{M}|^\star)_{i=1,\dots,n}$*, a* composition *map*

$$\begin{aligned}\mathbb{M}(A_1,\dots,A_n;B) \times \textstyle\prod_{i=1}^n \mathbb{M}(\Delta_i;A_i) &\to \mathbb{M}(\Delta_1,\dots,\Delta_n;B)\\ \big(t,(u_1,\dots,u_n)\big) &\mapsto t \circ \langle u_1,\dots,u_n\rangle\end{aligned}$$

subject to an associativity law and two unit laws (see e.g. [28, p. 35]). A multicategory functor $f : \mathbb{M} \to \mathbb{N}$ *consists of a map* $|f| : |\mathbb{M}| \to |\mathbb{N}|$ *with maps* $f_{A_\bullet,B} : \mathbb{M}(A_1,\dots,A_n;B) \to \mathbb{N}(fA_1,\dots fA_n;fB)$ *for every* $A_1,\dots,A_n,B \in |\mathbb{M}|$*, such that substitution and the identity are preserved (see e.g. [28, p. 39]).*

Definition 2 ([32,20]). *A* symmetric multicategory *consists of a multicategory* $\mathbb{M}$ *together with a symmetric group action: for each* $A_1,\dots,A_n \in |\mathbb{M}|$ *and* $\sigma \in S_n$ *one has* $(-)\bullet\sigma : \mathbb{M}(A_1,\dots,A_n;B) \to \mathbb{M}(A_{\sigma 1},\dots,A_{\sigma n};B)$ *compatible with substitution and satisfying unit and associativity laws (e.g. [28, p. 54]). A* symmetric multicategory functor *is a multicategory functor which preserves the action.*

We write **Multicat** (resp. **SMulticat**) for the category of (symmetric) multicategories and their functors, and write $t : \Gamma \to B$ for $t \in \mathbb{M}(\Gamma;B)$.

Example 1. Every monoidal category $(\mathcal{C},\otimes,I)$ induces a multicategory $\mathcal{TC}$. The objects are those of $\mathcal{C}$, with multimaps $(\mathcal{TC})(A_1,\dots,A_n;B) := \mathcal{C}(\bigotimes_{i=1}^n A_i, B)$ for a chosen n-ary bracketing of the tensor product. This determines functors **MonCat** $\to$ **Multicat**, and **SMonCat** $\to$ **SMulticat** (see *e.g.* [28, p. 39]); we denote both of these by $\mathcal{T}$.

Lambek [25] essentially observed that every multicategory has an internal language, as follows. One identifies multimaps $t : A_1,\dots,A_n \to B$ with terms $x_1 : A_1,\dots,x_n : A_n \vdash t : B$, for a fixed ordering of an infinite set of variables $\{x_1,x_2,\dots\}$. The identity Id_A is identified with the variable $x : A$, and the composition operation becomes a formal substitution operation on the language. Stated in this way, the three axioms become well-known properties of substitution: the unit laws say $x[u] = u$ and $t[x_1,\dots,x_n] = t$, and the associativity law is a linear version of the so-called Substitution Lemma (*e.g.* [5, Lemma 2.1.16]).

The next result shows this terminology does not differ too much from the notion of internal language in the introduction. For a signature $\mathcal{S}$ and $\Gamma := (x_i : A_i)_{i=1,\dots,n}$, write $\mathcal{O}_\mathcal{S}$ for the ordered language generated by the two rules on the left below, and $\mathfrak{L}_\mathcal{S}$ for the linear language generated by all three rules:

$$\frac{}{x : A \vdash x : A} \qquad \frac{c \in \mathcal{S}(\Gamma;B) \qquad (\Delta_i \vdash u_i : A_i)_{i=1,\dots,n}}{\Delta_1,\dots,\Delta_n \vdash c^{\S}(u_1,\dots,u_n) : B} \qquad \frac{\Theta, x : A, y : B, \Delta \vdash t : C}{\Theta, y : B, x : A, \Delta \vdash t : C}$$

Substitution is defined as usual, so that the following rule is admissible:

$$\frac{x_1 : A_1,\dots,x_n : A_n \vdash t : B \qquad (\Delta_i \vdash u_i : A_i)_{i=1,\dots,n}}{\Delta_1,\dots,\Delta_n \vdash t[u_1/x_1,\dots,u_n/x_n] : B} \tag{2}$$

With this rule as composition, $\mathcal{O}_\mathcal{S}$ and $\mathfrak{L}_\mathcal{S}$ define a syntactic multicategory $\mathrm{Syn}(\mathcal{O}_\mathcal{S})$ and a syntactic symmetric multicategory $\mathrm{Syn}(\mathfrak{L}_\mathcal{S})$, respectively. These

define left adjoints to the functors **Multicat** $\to$ **Sig** and **SMulticat** $\to$ **Sig** sending a (symmetric) multicategory $\mathbb{M}$ to the signature with objects $|\mathbb{M}|$ and constants $\{\mathbb{M}(\Gamma;B)\}_{\Gamma\in|\mathbb{M}|^\star,B\in|\mathbb{M}|}$; we denote both these functors by U.

Lemma 1. $\mathrm{Syn}(\mathcal{O}_\mathcal{S})$ *(resp.* $\mathrm{Syn}(\mathfrak{L}_\mathcal{S})$*) is the free multicategory (resp. symmetric multicategory) on* $\mathcal{S}$*.*

Thus, the internal language of a symmetric multicategory is the core of Abramsky's linear λ-calculus [1]. To recover a cartesian language, we use *(multi-sorted) abstract clones*. These differ from multicategories in that the result of substituting $(u_i : \Delta \to A_i)_{i=1,2}$ into $t : A_1, A_2 \to B$ yields a multimap of type $\Delta \to B$, not $\Delta, \Delta \to B$. Abstract clones are equivalently *cartesian multicategories* (see *e.g.* [18]), but this formulation is less natural syntactically: it amounts to adding explicit duplication and deletion operations to the language.

Definition 3. *An* abstract clone $\mathbb{C}$ *consists of a set* $|\mathbb{C}|$ *of* sorts *and sets* $\mathbb{C}(\Gamma;B)$ *of* multimaps *for every* $\Gamma \in |\mathbb{C}|^\star$ *and* $B \in |\mathbb{C}|$*, together with*

1. Projection *multimaps* $\mathsf{p}_i^{A\bullet} \in \mathbb{C}(A_1,\dots,A_n;A_i)$ *for every* $A_1,\dots,A_n \in |\mathbb{C}|$*;*
2. *For every* $A_1,\dots,A_n,B \in |\mathbb{C}|$ *and* $\Delta \in |\mathbb{C}|^\star$*, a* substitution *operation*

$$\begin{aligned}\mathbb{C}(A_1,\dots,A_n;B) \times \textstyle\prod_{i=1}^n \mathbb{C}(\Delta;A_i) &\to \mathbb{C}(\Delta;B)\\ \big(t,(u_1,\dots,u_n)\big) &\mapsto t[u_1,\dots,u_n]\end{aligned}$$

subject to an associativity law and two unit laws for any $t \in \mathbb{C}(A_1,\dots,A_n;B)$*,* $\big(u_i \in \mathbb{C}(B_1,\dots,B_m;A_i)\big)_{i=1,\dots,n}$ *and* $\big(v_j \in \mathbb{C}(\Theta;B_j)\big)_{j=1,\dots,m}$*:*

$$\big(t[u_\bullet]\big)[v_\bullet] = t\big[\dots,u_i[v_\bullet],\dots\big] \;\;,\;\; \mathsf{p}_i^{A\bullet}[u_1,\dots,u_n] = u_i \;\;,\;\; t[\mathsf{p}_1^{A\bullet},\dots,\mathsf{p}_n^{A\bullet}] = t$$

A homomorphism of clones $f : \mathbb{C} \to \mathbb{D}$ *consists of a map* $|f| : |\mathbb{C}| \to |\mathbb{D}|$ *and maps* $f_{A_\bullet,B} : \mathbb{C}(A_1,\dots,A_n;B) \to \mathbb{D}(fA_1,\dots fA_n;fB)$ *for every* $A_1,\dots,A_n,B \in |\mathbb{C}|$*, such that* $f(\mathsf{p}_i^{A\bullet}) = \mathsf{p}_i^{(fA)\bullet}$ *and* $f\big(t[u_1,\dots,u_n]\big) = (ft)[fu_1,\dots,fu_n]$*. We write* **Clone** *for the category of clones and clone homomorphisms.*

Example 2 (cf. Example 1). Any cartesian category $(\mathcal{C},\Pi)$ determines a clone $\mathcal{PC}$ with sorts the objects of $\mathcal{C}$ and $(\mathcal{PC})\big(A_1,\dots,A_n;B\big) := \mathcal{C}(\prod_{i=1}^n A_i;B)$.

We distinguish between clones and multicategories by using $[\dots]$ for a clone's substitution operation and $\langle\dots\rangle$ for a multicategory's composition operation. Every multicategory, and hence every clone, has an underlying category.

Definition 4. *The* nucleus $\overline{\mathbb{M}}$ *of a multicategory or clone* $\mathbb{M}$ *is the category with the same objects and* $\overline{\mathbb{M}}(A,B) := \mathbb{M}(A;B)$*. This defines functors* $\overline{(-)}$: **Multicat** $\to$ **Cat** *and* $\overline{(-)}$: **Clone** $\to$ **Cat** *to the category of small categories.*

The internal language of a clone is a cartesian version of that for multicategories. Write $\Lambda_\mathcal{S}$ for the language below; substitution is defined as usual.

$$\frac{(i=1,\dots,n)}{x_1:A_1,\dots,x_n:A_n \vdash x_i : A_i} \qquad \frac{c \in \mathcal{S}(\Gamma;B) \qquad (\Delta \vdash u_i : A_i)_{i=1,\dots,n}}{\Delta \vdash c^\S(u_1,\dots,u_n) : B}$$

Identifying variables with projections, we get a syntactic clone $\mathrm{Syn}(\Lambda_\mathcal{S})$.

Lemma 2. *The canonical forgetful functor* $\mathrm{U} : \mathbf{Clone} \to \mathbf{Sig}$ *has a left adjoint, and the free clone on* $\mathcal{S}$ *is* $\mathrm{Syn}(\Lambda_{\mathcal{S}})$.

Example 3. The languages $\Lambda^{\times}_{\mathcal{S}}, \Lambda^{\rightarrow}_{\mathcal{S}}$ and $\Lambda^{\times,\rightarrow}_{\mathcal{S}}$ each induce syntactic clones we denote by $\mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}}), \mathrm{Syn}(\Lambda^{\rightarrow}_{\mathcal{S}})$ and $\mathrm{Syn}(\Lambda^{\times,\rightarrow}_{\mathcal{S}})$, respectively.

3 Universal properties for multicategories

In this section we generalise the categorical notion of *universal arrows* (as in *e.g.* [30, §3]) to give a notion of universal property for multicategories. This will provide a uniform way to introduce new connectives to a type theory. One could also define the required conditions directly (see [7,40]), but here we wish to emphasise that they arise from category-theoretic ideas.

Definition 5 (*cf.* **[17]**). *Let* $f : \mathbb{M} \to \mathbb{N}$ *be a multicategory functor.*

1. *A* universal arrow from f to $Y \in |\mathbb{N}|$ *is a pair* $(R \in |\mathbb{M}|, \rho : fR \to Y)$ *such that for every* $t : fA_1, \ldots, fA_n \to Y$ *there exists a unique multimap* $t^{\#} : A_1, \ldots, A_n \to R$ *such that* $\rho \circ \langle f(t^{\#}) \rangle = t$.
2. *A* universal arrow from $X_1, \ldots, X_n \in |\mathbb{N}|$ to f *is a pair* $(R \in |\mathbb{M}|, \rho : X_1, \ldots, X_n \to fR)$ *such that for every* $t : X_1, \ldots, X_n \to fB$ *there exists a unique multimap* $t^{\#} : R \to B$ *such that* $f(t^{\#}) \circ \langle \rho \rangle = t$.

We extend this definition—and hence our notion of universal property—to clones by using the next observation (*cf.* the fact a cartesian category is monoidal).

Lemma 3. *There is a faithful functor* $\mathrm{M} : \mathbf{Clone} \to \mathbf{Multicat}$ *sending a clone* $\mathbb{C}$ *to the multicategory with the same objects and hom-sets, and composition given using substitution in* $\mathbb{C}$ *and the projections.*

Definition 5 does not involve 'global' conditions like naturality, so is particularly amenable to a type-theoretic interpretation. As in the categorical setting, however, it can be rephrased using natural isomorphisms (*cf.* [30, §3.2]).

Lemma 4. *Let* $f : \mathbb{M} \to \mathbb{N}$ *be a multicategory functor.*

1. *Giving a universal arrow from* f *to* $X \in |\mathbb{N}|$ *is equivalent to giving* $R \in \mathbb{M}$ *and an isomorphism* $\phi_{A_\bullet} : \mathbb{M}(A_1, \ldots, A_n; R) \xrightarrow{\cong} \mathbb{N}(fA_1, \ldots, fA_n; Y)$*, natural in the sense that the left diagram below commutes for any* $t : A_1, \ldots, A_n \to B$*;*
2. *Giving a universal arrow from* $X_1, \ldots, X_n \in |\mathbb{N}|$ *to* f *is equivalent to giving* $R \in |\mathbb{M}|$ *and an isomorphism* $\psi_B : \mathbb{M}(R; B) \xrightarrow{\cong} \mathbb{N}(X_1, \ldots, X_n; fB)$*, natural in the sense that the right diagram below commutes for any* $u : B \to C$.

$$\begin{array}{ccc} \mathbb{M}(B;R) & \xrightarrow{\phi_B} & \mathbb{N}(fB;X) \\ {\scriptstyle (-)\circ\langle t\rangle}\downarrow & & \downarrow{\scriptstyle (-)\circ\langle ft\rangle} \\ \mathbb{M}(A_1,\ldots,A_n;R) & \xrightarrow[\phi_{A_\bullet}]{} & \mathbb{N}(fA_1,\ldots,fA_n;X) \end{array} \qquad \begin{array}{ccc} \mathbb{M}(R;B) & \xrightarrow{\psi_B} & \mathbb{N}(X_1,\ldots,X_n;fB) \\ {\scriptstyle u\circ\langle -\rangle}\downarrow & & \downarrow{\scriptstyle f(u)\circ\langle -\rangle} \\ \mathbb{M}(R;C) & \xrightarrow[\psi_C]{} & \mathbb{N}X_1,\ldots,X_n;fC) \end{array}$$

A corollary is that giving a right adjoint to a multicategory functor $f : \mathbb{N} \to \mathbb{M}$ in Hermida's 2-category of multicategories [17] is equivalent to giving a mapping $g_0 : |\mathbb{M}| \to |\mathbb{N}|$ and a universal arrow $fg(X) \to X$ from f to X for each $X \in |\mathbb{N}|$.

4 Product structure

We now have enough to define products for multicategories, and hence for clones. An n-ary product is exactly a limit over the discrete category with n objects. Rephrasing in terms of universal arrows (*e.g.* [30, §3]) we get that equipping a category $\mathcal{C}$ with n-ary products is exactly equipping it with a universal arrow from the diagonal functor $\Delta^{(n)} : \mathcal{C} \to \mathcal{C}^{\times n}$ to $(A_1, \ldots, A_n)$ for every $A_1, \ldots, A_n \in \mathcal{C}$.

Since **Multicat** has finite products defined in much the same way as the category of small categories **Cat**, we may make the following definition. The prefix 'cartesian' is already used for multicategories, so we use '**f**inite-**p**roducts'.

Definition 6. *An* fp-multicategory *is a multicategory* $\mathbb{M}$ *equipped with a universal arrow* $\big(\prod_{i=1}^n A_i, (\pi_1^{A\bullet}, \ldots, \pi_n^{A\bullet})\big)$ *from the diagonal functor* $\Delta^{(n)} : \mathbb{M} \to \mathbb{M}^{\times n}$ *to* $(A_1, \ldots, A_n)$ *for every* $n \in \mathbb{N}$ *and* $A_1, \ldots, A_n \in |\mathbb{M}|$.

Asking for $\mathbb{M}$ to have finite products is equivalent to asking for a product object $\prod_{i=1}^n A_i$ and unary multimaps $\big(\pi_i^{A\bullet} : \prod_{i=1}^n A_i \to A_i\big)_{i=1,\ldots,n}$ for each $A_1, \ldots, A_n \in |\mathbb{M}|$, such that composition induces isomorphisms $\mathbb{M}\big(\Gamma; \prod_{i=1}^n A_i\big) \cong \prod_{i=1}^n \mathbb{M}(\Gamma; A_i)$. In the internal language, this amounts to the following rules:

$$\frac{}{p : \prod_{i=1}^n A_i \vdash \pi_i^{A\bullet}(p) : A_i}\ {\scriptstyle (i\,=\,1,\,\ldots,\,n)} \quad , \quad \frac{(\Gamma \vdash t_i : A_i)_{i=1,\ldots,n}}{\Gamma \vdash \langle t, \ldots, t_n\rangle : \prod_{i=1}^n A_i} \tag{3}$$

$$\pi_i^{A\bullet}(p)\big[\langle t_1, \ldots, t_n\rangle\big] = t_i \quad , \quad \big\langle \pi_1^{A\bullet}(p)[u], \ldots, \pi_n^{A\bullet}(p)[u]\big\rangle = u$$

We can now derive the rules for & in linear λ-calculus [1]. Indeed, given $\Gamma, x : A_i, \Theta \vdash t : B$, from (3) we get $\Gamma, p : \prod_{i=1}^n A_i, \Theta \vdash t[\pi_i^{A\bullet}(p)/x] : B$. This suggests the following. Let $\mathcal{O}_{\mathcal{S}}^{\&}$ (resp. $\mathfrak{L}_{\mathcal{S}}^{\&}$) be the extension of $\mathcal{O}_{\mathcal{S}}$ (resp. $\mathfrak{L}_{\mathcal{S}}$) with

$$\frac{\Gamma, x_i : A_i, \Theta \vdash t : C \qquad \Delta \vdash u : \&_{i=1}^n A_i}{\Gamma, \Delta, \Theta \vdash \mathtt{let}\ x_i\ \mathtt{be}\ \mathtt{p}_i\ \mathtt{of}\ u\ \mathtt{in}\ t : C} \quad , \quad \frac{(\Gamma \vdash t_i : A_i)_{i=1,\ldots,n}}{\Gamma \vdash \langle t_1, \ldots, t_n\rangle : \&_{i=1}^n A_i}$$

$$\mathtt{let}\ x_i\ \mathtt{be}\ \mathtt{p}_i\ \mathtt{of}\ \langle u_i\rangle_{i=1}^n\ \mathtt{in}\ t = t[u_i/x_i] \quad , \quad \langle \mathtt{let}\ x_i\ \mathtt{be}\ \mathtt{p}_i\ \mathtt{of}\ u\ \mathtt{in}\ x_i\rangle_{i=1}^n = u$$

where we write $\langle u_i\rangle_{i=1}^n$ for $\langle u_1, \ldots, u_n\rangle$. This syntax defines a free property. To see this, say a multicategory functor f *(strictly) preserves finite products* if it preserves all the data on the nose, so that $f(\prod_{i=1}^n A_i) = \prod_{i=1}^n fA_i$, $f(\pi_i^{A\bullet}) = \pi_i^{fA\bullet}$, and $f(\langle t_1, \ldots, t_n\rangle) = \langle ft_1, \ldots, ft_n\rangle$. Write **fpMulticat** for the category of fp-multicategories and product-preserving functors, and **fpSMulticat** for the subcategory of symmetric multicategories with finite products, with functors preserving both structures.

Lemma 5. *The composite forgetful functor* **fpMulticat** → **Multicat** → **Sig** *has a left adjoint, and the free fp-multicategory on* $\mathcal{S}$ *is* $\mathrm{Syn}(\mathcal{O}_{\mathcal{S}}^{\&})$. *This extends to symmetric structure: replace* **fpMulticat** *by* **fpSMulticat** *and* $\mathcal{O}^{\&}$ *by* $\mathfrak{L}^{\&}$.

Returning to the cartesian setting, we define products in a clone using the corresponding structure for multicategories and Lemma 3.

Definition 7. *A* cartesian clone $(\mathbb{C}, \Pi)$ *is a clone* $\mathbb{C}$ *equipped with a choice of finite products on* $\mathrm{M}\mathbb{C}$*. A* (strict) homomorphism of cartesian clones *is a clone homomorphism* f *that strictly preserves all the product structure. We write* **CartClone** *for the category of cartesian clones and strict homomorphisms.*

Writing $\pi_i(t)$ for the multimap $\pi_i^{A\bullet}[t]$, the rules (3) translate directly to the usual product rules of λ-calculus. So cartesian clones exactly capture $\Lambda^\times$.

Lemma 6. *The composite forgetful functor* **CartClone** → **Clone** → **Sig** *has a left adjoint, and* $\mathrm{Syn}(\Lambda^\times_{\mathcal{S}})$ *is the free cartesian clone on* $\mathcal{S}$*.*

Using the characterisation of universal arrows in terms of natural isomorphisms we get the following refinement of Example 2.

Example 4. For any cartesian category $(\mathcal{C}, \Pi)$ the induced clone $\mathcal{PC}$ is cartesian, essentially by definition; this extends to a functor $\mathcal{P} : \mathbf{CartCat} \to \mathbf{CartClone}$. Moroever, if $(\mathbb{C}, \Pi)$ is a cartesian clone, then so is its nucleus $\overline{\mathbb{C}}$. Hence $\overline{(-)}$ restricts to a functor $\mathbf{CartClone} \to \mathbf{CartCat}$.

The two functors in this example are actually adjoints, yielding our first version of the schema in (1). The unit is identity-on-objects and sends $t : A_1, \ldots, A_n \to B$ to $t[\pi_1^{A\bullet}, \ldots, \pi_n^{A\bullet}] : \prod_{i=1}^n A_i \to B$.

Proposition 1. *The functor* $\overline{(-)} : \mathbf{CartClone} \to \mathbf{CartCat}$ *fits into the following diagram of adjunctions:*

$$\mathbf{Sig} \underset{\mathrm{U}}{\overset{\mathrm{F}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathbf{CartClone} \underset{\mathcal{P}}{\overset{\overline{(-)}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathbf{CartCat}$$

Moreover, $\mathrm{U} \circ \mathcal{P}$ *is equal to the canonical forgetful functor* $\mathbf{CartCat} \to \mathbf{Sig}$*. Hence, the free cartesian category on* $\mathcal{S}$ *is canonically isomorphic to* $\overline{\mathrm{Syn}(\Lambda^\times_{\mathcal{S}})}$*.*

4.1 Cartesian structure from representability

In the preceding section we defined products using a multi-ary version of the familiar universal property. There is another way to define 'monoidal structure' in a multicategory: Hermida's *representability* [17]. From the perspective of linear logic, the finite product structure explored above corresponds to the additive conjunction &; Hermida's representability will correspond to the multiplicative conjunction ⊗. We shall also see that, for clones, the two are equivalent.

Definition 8. *A* representable multicategory *is a multicategory* $\mathbb{M}$ *equipped with a universal arrow* $\Big(\mathrm{T}(X_1, \ldots, X_n), \rho_{X_\bullet} : X_1, \ldots, X_n \to \mathrm{T}(X_1, \ldots, X_n)\Big)$ *from* $X_1, \ldots, X_n$ *to the identity* $\mathrm{id}_{\mathbb{M}}$ *for each* $X_1, \ldots, X_n \in |\mathbb{M}|$*; we write* $\mathrm{T}_{i=1}^n X_i$ *for* $\mathrm{T}(X_1, \ldots, X_n)$*. These universal arrows must be closed under composition, so*

$$X_1, \ldots, X_n, Y_1, \ldots, Y_m \xrightarrow{\langle \rho_{X_\bullet}, \rho_{Y_\bullet} \rangle} \mathrm{T}_{i=1}^n X_i, \mathrm{T}_{j=1}^m Y_j \xrightarrow{\rho} \mathrm{T}\big(\mathrm{T}_{i=1}^n X_i, \mathrm{T}_{j=1}^m Y_j\big)$$

must also be universal. A representable multicategory functor f *is a multicategory functor that preserves all the universal arrows, so that* $f(\mathrm{T}^n_{i=1} A_i) = \mathrm{T}^n_{i=1} fA_i$, $f(\rho_{A_\bullet}) = \rho_{fA_\bullet}$ *and* $f(t^\#) = ft^\#$. *Write* **RepMulticat** *for the category of representable multicategories, and* **SRepMulticat** *for the category of representable multicategories whose underlying multicategories are also symmetric, with functors preserving both structures.*

Example 5 (cf. Example 1). The multicategory $\mathcal{TC}$ induced by a monoidal category $(\mathcal{C}, \otimes, I)$ is representable. We therefore obtain functors **MonCat** $\to$ **RepMulticat** and **SMonCat** $\to$ **SRepMulticat**; we denote them both $\mathcal{T}$.

A representable multicategory is a multicategory equipped with rules which are dual to those in (3) in the sense that the universal arrow goes the other direction. Indeed, writing $x_1 \otimes \ldots \otimes x_n$ for $\rho_{A_\bullet}$, and $\mathtt{let}\ (x_1, \ldots, x_n)\ \mathtt{be}\ p\ \mathtt{in}\ t$ for $t^\#$, and extending this to all terms by

$$u_1 \otimes \ldots \otimes u_n := (x_1 \otimes \ldots \otimes x_n)[u_1/x_1, \ldots, u_n/x_n]$$

$$\mathtt{let}\ (x_1, \ldots, x_n)\ \mathtt{be}\ u\ \mathtt{in}\ t := \big(\mathtt{let}\ (x_1, \ldots, x_n)\ \mathtt{be}\ p\ \mathtt{in}\ t\big)[u/p]$$

we obtain the following rules, where $\Gamma := (x_i : A_i)_{i=1,\ldots,n}$:

$$\frac{(\Delta_i \vdash u_i : A_i)_{i=1,\ldots,n}}{\Delta_1, \ldots, \Delta_n \vdash \otimes^n_{i=1} u_i : \bigotimes^n_{i=1} A_i} \quad , \quad \frac{\Lambda, \Gamma, \Theta \vdash t : B \qquad \Delta \vdash u : \bigotimes^n_{i=1} A_i}{\Lambda, \Delta, \Theta \vdash \mathtt{let}\ (x_1, \ldots, x_n)\ \mathtt{be}\ u\ \mathtt{in}\ t : B} \tag{4}$$

$$\mathtt{let}\ (x_1, \ldots, x_n)\ \mathtt{be}\ p\ \mathtt{in}\ t[\otimes^n_{i=1} x_i/p] = t\ ,\ \mathtt{let}\ (x_1, \ldots, x_n)\ \mathtt{be} \otimes^n_{i=1} x_i\ \mathtt{in}\ t = t$$

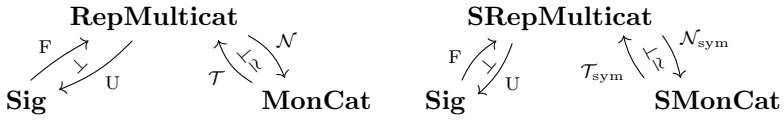
We write $\mathcal{O}^\otimes_\mathcal{S}$ (resp. $\mathfrak{L}^\otimes_\mathcal{S}$) for the extension of $\mathcal{O}_\mathcal{S}$ (resp. $\mathfrak{L}_\mathcal{S}$) with these rules. This is essentially the tensor fragment of Abramsky's linear λ-calculus [1]. The connection with multicategories was already made in by Hyland & de Paiva [20], who showed this type theory arises from Lambek's *monoidal multicategories* [26].

Lemma 7. *The composite forgetful functor* **RepMulticat** $\to$ **Multicat** $\to$ **Sig** *has a left adjoint, and the free representable multicategory on* $\mathcal{S}$ *is the syntactic multicategory* $\mathrm{Syn}(\mathcal{O}^\otimes_\mathcal{S})$. *The same holds for symmetric structure, if one replaces* **RepMulticat** *by* **SRepMulticat** *and* $\mathcal{O}^\otimes$ *by* $\mathfrak{L}^\otimes$.

Combining this lemma with Lemma 5, one sees that a multicategory equipped with representable and finite-product structure corresponds to a linear type theory with both $\otimes$ and &.

We can also obtain a linear version of Proposition 1. Hermida [17] showed that the 2-category of representable multicategories is 2-equivalent to the 2-category of monoidal categories, and Weber showed this extends to the symmetric case [45]. From these constructions one can extract functors $\mathcal{T} :$ **RepMulticat** $\to$ **MonCat** and $\mathcal{T}_{\mathrm{sym}} :$ **SRepMulticat** $\to$ **SMonCat** sending a (symmetric) representable multicategory to a (symmetric) monoidal structure on its nucleus, together with equivalences **RepMulticat** $\simeq$ **MonCat** and **SRepMulticat** $\simeq$ **SMonCat**. So we get the following.

Proposition 2. *The functors $\mathcal{N}$ and $\mathcal{N}_{\mathrm{sym}}$ fit into the following diagram of adjunctions, where in each case the right-hand adjunction is an equivalence:*



Moreover, $\mathrm{U} \circ \mathcal{T}$ *and* $\mathrm{U} \circ \mathcal{T}_{\mathrm{sym}}$ *are both equal to the canonical forgetful functor to* **Sig**. *Hence, the free monoidal (resp. symmetric monoidal) category on a signature* $\mathcal{S}$ *is canonically isomorphic to* $\mathcal{N}\big(\mathrm{Syn}(\mathcal{O}^{\otimes}_{\mathcal{S}})\big)$ *(resp.* $\mathcal{N}\big(\mathrm{Syn}(\mathfrak{L}^{\otimes}_{\mathcal{S}})\big)$*).*

We now turn to studying representability in the cartesian setting.

Definition 9. *A* representable clone *is a clone* $\mathbb{C}$ *equipped with a choice of representable structure on* $\mathrm{M}\mathbb{C}$. *A* representable clone homomorphism *is a clone homomorphism which preserves the representable structure as in Definition 8.*

A cartesian clone makes the *projections* primitive (recall (3)), but a representable clone makes the *pairing operation* primitive (recall (4)). It turns out these perspectives are equivalent. In the proof-theoretic setting such ideas are well-studied (*cf.* the equivalence of G-systems and N-systems in [42, §3.3]); the categorical statement has also been made by Pisani [34] and Shulman [40].

Proposition 3. *Equipping a clone* $\mathbb{C}$ *with representable structure is equivalent to equipping* $\mathbb{C}$ *with cartesian structure.*

In Proposition 2 we gave an equivalence of categories but in Proposition 1 we only gave an adjunction. We can now upgrade the latter to an equivalence. Indeed, $\overline{(-)} \circ \mathcal{P}$ is equal to the identity. On the other hand, if $(\mathbb{C}, \Pi)$ is a cartesian clone then by Proposition 3 and Lemma 4 we have a multi-natural isomorphism $\mathbb{C}(A_1, \ldots, A_n; B) \cong \mathbb{C}(\prod_{i=1}^n A_i; B) = \mathcal{P}(\overline{\mathbb{C}})(A_1, \ldots, A_n; B)$.

Corollary 1 ([34]). *The functors $\mathcal{P}$ and $\overline{(-)}$ of Proposition 1 define an adjoint equivalence* $\mathbf{CartClone} \simeq \mathbf{CartCat}$.

4.2 Recovering the semantic interpretation and syntactic model

We now show how the usual semantic interpretation, syntactic model, and soundness and completeness results can be derived from the multi-ary framework. Although we shall not pursue the point in detail for reasons of space, essentially the same argument holds for all the calculi considered in this paper.

Semantic interpretation and soundness. We recover the usual semantic interpretation of $\Lambda^{\times}$ in a cartesian category by Lemma 6 and Example 4 as follows. Let $\mathrm{U} : \mathbf{CartCat} \to \mathbf{Sig}$ be the functor sending a cartesian category $(\mathcal{C}, \Pi)$ to the signature with objects those of $\mathcal{C}$ and constants $\big\{\mathcal{C}(\prod_{i=1}^n A_i, B)\big\}_{A_1,\ldots,A_n,B \in \mathcal{C}}$. An interpretation $s : \mathcal{S} \to \mathrm{U}\mathcal{C}$ of basic types and constants in $\mathcal{C}$ is exactly an

interpretation $s : \mathcal{S} \to \mathrm{U}(\mathcal{PC})$ in the induced cartesian clone. The unique extension $s[\![-]\!] : \mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}}) \to \mathcal{PC}$ sends a term $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ to a multimap $s[\![x_1 : A_1, \ldots, x_n : A_n]\!] \to s[\![B]\!]$ in $\mathcal{PC}$, which is exactly a map $\prod_{i=1}^{n} s[\![A_i]\!] \to s[\![B]\!]$ in $\mathcal{C}$. It is not hard to show this coincides with the usual, inductively defined semantic interpretation. Unlike with the unary approach, we do not need to prove soundness with respect to $\beta\eta$ as a separate lemma: this holds immediately from the fact $s[\![-]\!]$ is a cartesian clone homomorphism.

Moreover, for any objects $A_1, \ldots, A_n$ in a cartesian clone one can construct a 'multi-isomorphism' $(A_1, \ldots, A_n) \cong \prod_{i=1}^{n} A_i$ (see [38, Lemma 4.2.16]). Hence, in a cartesian simple type theory with products, *contexts must coincide with product types*. Together with the preceding, this provides a mathematical explanation for the identification of contexts and product types in the interpretation of $\Lambda^{\times,\to}$.

Syntactic model. We extract the construction from Proposition 1. For a signature $\mathcal{S}$ the cartesian category $\overline{\mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}})}$ has objects the types of $\Lambda^{\times}_{\mathcal{S}}$ and morphisms $A \to B$ given by $\alpha\beta\eta$-equivalence classes of terms $x : A \vdash t : B$ for a fixed variable x. Composition is substitution and the identity on A is the variable $x : A$. The projections are $x : \prod_{i=1}^{n} A_i \vdash \pi_i^{A\bullet}(x) : A_i$ and the pairing of the maps $(x : C \vdash t_i : A_i)_{i=1,2}$ is $x : C \vdash \langle t_1, t_2 \rangle : A_1 \times A_2$. The usual proofs that this is indeed cartesian (see *e.g.* [9, Chapter 3]) have been replaced by the simple observation of Example 4.

Completeness. Once again, the proof is largely category-theoretic. Note first that the functor $\overline{(-)} : \mathbf{CartClone} \to \mathbf{CartCat}$ is faithful. One can prove this directly using Proposition 3 or infer it from Corollary 1 and the fact any equivalence is fully faithful. In any case, it follows by standard results (*e.g.* [37, Lemma 4.5.13]) that the unit η' of the adjunction $\overline{(-)} \dashv \mathcal{P}$ is monic. Just as in **Cat**, any monomorphism of clones is injective on objects and injective on multimaps. It suffices, therefore, to find a semantic interpretation $\iota[\![-]\!]$ which is equal to a component of η'. This is accomplished by the next lemma.

Lemma 8. *Let* $\mathcal{C} \underset{\mathrm{U}}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathcal{D} \underset{\mathrm{U}'}{\overset{F'}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathcal{E}$ *be adjunctions with units* $\eta : \mathrm{id}_{\mathcal{C}} \Rightarrow \mathrm{U}F$ *and* $\eta' : \mathrm{id}_{\mathcal{D}} \Rightarrow \mathrm{U}'F'$. *Then for any* $C \in \mathcal{C}$, *the unit* $\eta'_{FC} : FC \to \mathrm{U}'F'FC$ *is the unique map* h *such that the following diagram commutes:*

$$\begin{array}{ccc} \mathrm{U}FC & \overset{\mathrm{U}h}{\dashrightarrow} & \mathrm{U}\mathrm{U}'F'FC \\ {\scriptstyle \eta_C}\uparrow & & \uparrow{\scriptstyle \mathrm{U}\eta'_{FC}} \\ C & \xrightarrow[\eta_C]{} & \mathrm{U}FC \end{array}$$

In the setting of Proposition 1 this lemma implies that the component $\eta'_{F\mathcal{S}} : \mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}}) \to \mathcal{P}\big(\overline{\mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}})}\big)$ of the unit for the adjunction $\overline{(-)} \dashv \mathcal{P}$ is exactly the unique cartesian clone homomorphism $\iota[\![-]\!]$ extending the obvious interpretation $\iota := \mathcal{S} \hookrightarrow \overline{\mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}})}$ of base types and constants in the free cartesian category. By our preceding discussion, this clone homomorphism is injective on multimaps: so if $\iota[\![t]\!] = \iota[\![t']\!]$ then $t = t'$ in $\mathrm{Syn}(\Lambda^{\times}_{\mathcal{S}})$, hence $t =_{\beta\eta} t'$.

5 Closed structure

To define closed structure, we follow Lambek's definition and simply upgrade the hom-set definition of exponentials to multicategories.

Definition 10 ([26]). *A* closed multicategory *is a multicategory* $\mathbb{M}$ *equipped with an object* $[A, B]$ *and multimap* $\mathrm{eval}_{A,B} : [A, B], A \to B$ *for every* $A, B \in |\mathbb{M}|$*, such that composition induces isomorphisms as shown:*

$$\mathbb{M}(\Gamma, A; B) \underset{\mathrm{eval}_{A,B} \circ \langle (-), \mathrm{Id}_A \rangle}{\overset{\Lambda^A}{\underset{\cong}{\rightleftarrows}}} \mathbb{M}(\Gamma; [A, B]) \tag{5}$$

A (strict) closed multicategory functor *is a multicategory functor* f *which preserves all the data:* $f([A, B]) = [fA, fB]$*,* $f(\mathrm{eval}_{A,B}) = \mathrm{eval}_{fA,fB}$ *and* $f(\Lambda t) = \Lambda(ft)$*. We write* **ClMulticat** *for the category of closed multicategories and their functors, and* **ClSMulticat** *for the category of symmetric multicategories with closed structure, and functors preserving both of these.*

Example 6. If $(\mathcal{C}, \otimes, I, [-,=])$ is a closed (symmetric) monoidal category then the induced (symmetric) multicategory $\mathcal{TC}$ is also closed.

Closed multicategories allow us to model exponentials without requiring a tensor product. Writing out the rules in the internal language, we get the map Λ^A in (5) as the usual abstraction rule, and the evaluation map as the application $f : A \multimap B, x : A \vdash f\, x : B$. We then see that $\Delta, f : A \multimap B, x : A \vdash u[f\, x/y] : C$ whenever $\Delta, y : B \vdash u : C$, so we recover a small adaptation of Abramsky's rules for exponentials. Write $\mathcal{O}^{\multimap}_{\mathcal{S}}$ (resp. $\mathfrak{L}^{\multimap}_{\mathcal{S}}$) for the extension of $\mathcal{O}_{\mathcal{S}}$ (resp. $\mathfrak{L}_{\mathcal{S}}$) with the following rules and the $\beta\eta$-laws familiar from $\Lambda^{\to}$:

$$\frac{\Delta, y : B \vdash u : C \qquad \Theta \vdash t : A \multimap B \qquad \Gamma \vdash v : A}{\Delta, \Theta, \Gamma \vdash u[t\, v/y] : C} \quad , \quad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x\,.\, t : A \multimap B}$$

Lemma 9 ([20]). *The composite forgetful functor* **ClMulticat** $\to$ **Multicat** $\to$ **Sig** *has a left adjoint, and the free closed multicategory on* $\mathcal{S}$ *is the syntactic multicategory* $\mathrm{Syn}(\mathcal{O}^{\multimap}_{\mathcal{S}})$*. The same holds for symmetric structure, if one replaces* **ClMulticat** *by* **ClSMulticat** *and* $\mathcal{O}^{\multimap}$ *by* $\mathfrak{L}^{\multimap}$*.*

For the cartesian case, we follow the same procedure as in Section 4.

Definition 11. *A* closed clone *is a clone* $\mathbb{C}$ *equipped with a closed structure on* $\mathrm{M}\mathbb{C}$*. We write* **ClClone** *for the category of closed clones and clone homomorphisms preserving the closed structure as in Definition 10.*

Example 7. If $(\mathcal{C}, \Pi, \Rightarrow)$ is a cartesian closed category, the clone $\mathcal{PC}$ is closed.

Definition 11 recovers the usual $\beta\eta$-laws for exponentials in $\Lambda^{\to}$, complete with the weakenings that are usually implicit. Writing $f\, x$ for eval, we get the following equations in the internal language when $\Gamma := (x_i : A_i)_{i=1,\ldots,n}$:

$$(f\, x)\big[(\lambda x\,.\, t)[x_1/x_1, \ldots, x_n/x_n]/f, x/x\big] = t \ , \ \lambda x\,.\, (f\, x)\big[t[x_1/x_1, \ldots, x_n/x_n]/f\big] = t$$

Lemma 10. *The composite forgetful functor* **ClClone** $\to$ **Clone** $\to$ **Sig** *has a left adjoint, and the free closed clone on* $\mathcal{S}$ *is the syntactic clone* $\mathrm{Syn}(\Lambda^{\to}_{\mathcal{S}})$*.*

6 Cartesian closed structure

The development above makes defining cartesian closed structure straightforward. For reasons of space we restrict ourselves to the cartesian case, but similar remarks apply to the linear and ordered cases.

Definition 12. *A* cartesian closed clone *is a clone equipped with both closed structure and cartesian structure. We write* **CCClone** *for the category of cartesian closed clones and homomorphisms that strictly preserve both structures.*

By Lemmas 6 and 10, we already have a free property .

Lemma 11. *The composite forgetful functor* $\mathbf{CCClone} \to \mathbf{Clone} \to \mathbf{Sig}$ *has a left adjoint, and* $\mathrm{Syn}(\Lambda_{\mathcal{S}}^{\times,\to})$ *is the free cartesian closed clone on* $\mathcal{S}$*.*

The nucleus of any cartesian closed clone $(\mathbb{C}, \Pi, \Rightarrow)$ is also cartesian closed:

$$\overline{\mathbb{C}}(A \times B, C) = \mathbb{C}(A \times B; C) \cong \mathbb{C}(A, B; C) \cong \mathbb{C}(A; B \Rightarrow C) = \overline{\mathbb{C}}(A, B \Rightarrow C)$$

Similarly, by Examples 4 and 7, for any cartesian closed category $(\mathcal{C}, \Pi, \Rightarrow)$ the induced category $\mathcal{PC}$ is cartesian closed. Proposition 1 then restricts as follows.

Proposition 4. *The functor* $\overline{(-)} : \mathbf{CCClone} \to \mathbf{CCCat}$ *fits into the following diagram, in which the right-hand adjunction is an equivalence:*

$$\mathbf{Sig} \underset{\mathrm{U}}{\overset{\mathrm{F}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathbf{CCClone} \underset{\mathcal{P}}{\overset{\overline{(-)}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\simeq\bot}}}} \mathbf{CCCat}$$

Moreover, $\mathrm{U} \circ \mathcal{P}$ *is equal to the canonical forgetful functor* $\mathbf{CCCat} \to \underline{\mathbf{Sig}}$*. Hence, the free cartesian closed category on* $\mathcal{S}$ *is canonically isomorphic to* $\overline{\mathrm{Syn}(\Lambda_{\mathcal{S}}^{\times,\to})}$*.*

As in Section 4.2, the preceding two results are enough to recover the sound semantic interpretation of $\Lambda^{\times,\to}$, and the usual syntactic model.

7 Cartesian combinatory logic and SK-clones

In this section we begin a multi-ary investigation of cartesian combinatory logic, and give a categorical statement of the classical correspondence between combinatory logic and $\Lambda^{\to}$ (for which see *e.g.* [15,6]). In Section 8 we shall use this to define *SK-categories* and show they are sound and complete for $\Lambda^{\to}$.

We briefly recapitulate the rules of typed combinatory logic $\mathsf{CL}_{\mathcal{S}}$ over a signature $\mathcal{S}$; for a fuller account see *e.g.* [6]. Types are as in $\Lambda^{\to}$. Terms are given by the grammar $t, u ::= x \,|\, c \in \mathcal{S}(\Gamma; B) \,|\, (t\,u) \,|\, \mathsf{S} \,|\, \mathsf{K}$: we have variables, constants and an application operation as in $\Lambda^{\to}$ and, for any context Γ and types A, B and C, two *combinators* $\Gamma \vdash \mathsf{S}^{\Gamma}_{A,B,C} : \big(A \Rightarrow (B \Rightarrow C)\big) \Rightarrow \big((A \Rightarrow C) \Rightarrow (A \Rightarrow C)\big)$ and $\Gamma \vdash \mathsf{K}^{\Gamma}_{A,B} : A \Rightarrow (B \Rightarrow A)$. Substitution is as in $\Lambda^{\to}$, where the combinators $\mathsf{Z} \in \{\mathsf{S}, \mathsf{K}\}$ satisfy $\mathsf{Z}[u_1/x_1, \ldots, u_n/x_n] = \mathsf{Z}$ so that Z^{Γ} is the weakening of $\mathsf{Z}^{\diamond}$.

The correlate of β-equality is *weak equality* $=_{\mathrm{w}}$, which is the smallest congruence containing $\mathsf{S}\,x\,y\,z = (x\,z)\,(y\,z)$ and $\mathsf{K}\,x\,y = x$. The correlate of $\beta\eta$-equality is *extensional weak equality* $=_{\mathrm{wext}}$, which extends $=_{\mathrm{w}}$ with the rule

$$\frac{t\,x_1\,\cdots\,x_n = t'\,x_1\,\cdots\,x_n \qquad x_1,\ldots,x_n \text{ not free in } t \text{ or } t'}{t = t'}\ \mathrm{ext} \tag{6}$$

We write CL^{w} for combinatory logic with weak equality and $\mathsf{CL}^{\mathrm{wext}}$ for combinatory logic with extensional weak equality. The usual encoding of CL^{w} in $\Lambda^{\to}$ sends S and K to $\lambda f\,.\,\lambda g\,.\,\lambda x\,.\,(f\,x)\,(g\,x)$ and $\lambda x\,.\,\lambda y\,.\,x$, respectively.

The next definition may be obtained by seeing that CL^{w} can be presented as an algebraic theory, and that clones are equivalent to algebraic theories (*e.g.* [29,41]). We implicitly bracket application to the left, so $t\cdot u\cdot v := (t\cdot u)\cdot v$. We also write $(-)^{\Delta;\Theta}$ for the weakening map $\mathbb{C}(\Gamma;B)\to\mathbb{C}(\Delta,\Gamma,\Theta;B)$ sending t to $t\big[\mathsf{p}^{\Delta,\Gamma,\Theta}_{|\Delta|+1},\ldots,\mathsf{p}^{\Delta,\Gamma,\Theta}_{|\Delta|+|\Gamma|},\big]$; when Γ is empty we write just $(-)^{\Delta}$.

Definition 13. *An* SK-clone *is a clone $\mathbb{C}$ equipped with a mapping $[-,=] : |\mathbb{C}|\times|\mathbb{C}|\to|\mathbb{C}|$, nullary multimaps $\mathsf{S}_{A,B,C}\in\mathbb{C}\big(\diamond;\big[[A,[B,C]],[[A,B],[A,C]]\big]\big)$ and $\mathsf{K}_{A,B}\in\mathbb{C}\big(\diamond;[A,[B,A]]\big)$ for every $A,B,C\in|\mathbb{C}|$, and a binary* application *operation $(-\cdot=) : \mathbb{C}(\Gamma;[A,B])\times\mathbb{C}(\Gamma;A)\to\mathbb{C}(\Gamma;B)$ for every $\Gamma\in|\mathbb{C}|^{\star}$ and $B\in|\mathbb{C}|$, such that the following axioms hold whenever they are well-typed:*

$$(t\cdot u)[v_1,\ldots,v_n] = t[v_1,\ldots,v_n]\cdot u[v_1,\ldots,v_n] \quad , \quad (\mathsf{K}_{A,B})^{A,B}\cdot\mathsf{p}_1\cdot\mathsf{p}_2 = \mathsf{p}_1$$

$$(\mathsf{S}_{A,B,C})^{[A,[B,C]],[A,B],A}\cdot\mathsf{p}_1\cdot\mathsf{p}_2\cdot\mathsf{p}_3 = (\mathsf{p}_1\cdot\mathsf{p}_3)\cdot(\mathsf{p}_2\cdot\mathsf{p}_3)$$

A homomorphism of SK-clones *is a clone homomorphism that preserves application, S and K: $f(\mathsf{S}_{A,B,C}) = \mathsf{S}_{fA,fB,fC}$, $f(\mathsf{K}_{A,B}) = \mathsf{K}_{fA,fB}$ and $f(t\cdot u) = ft\cdot fu$. We write* **SKClone** *for the category of SK-clones and their homomorphisms.*

Lemma 12. *The composite forgetful functor* **SKClone** $\to$ **Clone** $\to$ **Sig** *has a left adjoint, and the free SK-clone on $\mathcal{S}$ is the syntactic clone* $\mathrm{Syn}(\mathsf{CL}^{\mathrm{w}}_{\mathcal{S}})$.

A core feature of the syntax of combinatory logic, which is at the heart of the correspondence between the terms of $\mathsf{CL}^{\mathrm{wext}}$ and $\Lambda^{\to}$, is the admissibility of *bracket extension* algorithms (see *e.g.* [5, §7.1]). To express this in the typed setting, we use the following notation. For a binary operation $[-,=]$ on a set S we define $[-;=] : S^{\star}\times S\to S$ inductively as follows:

$$[\diamond;B] := B \quad , \quad [A;B] := [A,B] \quad , \quad [\Gamma,A;B] := [\Gamma;[A,B]]$$

With this notation, bracket abstraction amounts to saying that if $\Gamma := (x_i : A_i)_{i=1,\ldots,n}$ and $\Gamma\vdash t : B$ in CL^{w}, there exists a closed term $\diamond\vdash t^{\mathrm{c}} : [\Gamma;B]$ such that $(t^{\mathrm{c}})^{\Gamma}\,x_1\,\ldots\,x_n =_{\mathrm{w}} t$. The extensionality axiom (6) then says that t^{c} is unique: in other words, $t\mapsto t^{\Gamma}\,x_1\,\ldots\,x_n$ is an isomorphism.

We now translate this into clone-theoretic terms. For any SK-clone $\mathbb{C}$ we obtain the operation $t\mapsto t^{\Gamma}\,x_1\,\ldots\,x_n$ as the composite below:

$$i_{\Gamma;B} := \Big(\mathbb{C}(\diamond;[\Gamma;B]) \xrightarrow{\mathrm{w}^{\Gamma}} \mathbb{C}(\Gamma;[\Gamma;B]) \xrightarrow{(-)\cdot\mathsf{p}^{\Gamma}_1\cdot\ldots\cdot\mathsf{p}^{\Gamma}_{|\Gamma|}} \mathbb{C}(\Gamma;B)\Big) \tag{7}$$

For $\Gamma := \diamond$ this is just the identity. The admissibility of bracket abstraction in the syntax of CL^{w} is then captured by the next lemma. Typically bracket abstraction algorithms restrict to closed constants, because an open constant may have no corresponding closed term. We restrict in the same way. Call a signature $\mathcal{S}$ *nullary* if $\mathcal{S}(\Gamma; A) = \varnothing$ whenever $\Gamma \neq \diamond$, and write $\mathbf{Sig}_0 \hookrightarrow \mathbf{Sig}$ for the full subcategory of nullary signatures.

Lemma 13. *Let $\mathcal{S}$ be a nullary signature. Then for any $\Gamma \in |\mathrm{Syn}(\mathsf{CL}^{\mathrm{w}}_{\mathcal{S}})|^{\star}$ and $B \in |\mathrm{Syn}(\mathsf{CL}^{\mathrm{w}}_{\mathcal{S}})|$ there exists a map $(-)^{\mathrm{c}}$ such that $i_{\Gamma;B} \circ (-)^{\mathrm{c}} = \mathrm{id}_{\mathrm{Syn}(\mathsf{CL}^{\mathrm{w}}_{\mathcal{S}})}$.*

Because bracket abstraction is defined by induction on the syntax, we cannot straightforwardly define it in an arbitrary SK-clone. We can, however, consider the sub-category of SK-clones (= semantic models of CL^{w}) which admit bracket abstraction in the sense that each $i_{\Gamma;B}$ has a retraction. The *extensional* models are then those for which this retract $(-)^{\mathrm{c}}$ also satisfies uniqueness.

Definition 14. *An SK-clone $\mathbb{C}$ is* extensional *if for every $\Gamma \in |\mathbb{C}|^{\star}$ and $B \in |\mathbb{C}|$ the map $i_{\Gamma;B}$ defined in (7) is invertible. We write $\mathbf{SKClone}_{\mathrm{ext}}$ for the full subcategory of $\mathbf{SKClone}$ consisting of just the extensional SK-clones.*

Lemma 14. *The composite forgetful functor $\mathbf{SKClone}_{\mathrm{ext}} \to \mathbf{Clone} \to \mathbf{Sig}_0$ has a left adjoint, and the free extensional SK-clone on a nullary signature $\mathcal{S}$ is the syntactic clone $\mathrm{Syn}(\mathsf{CL}^{\mathrm{wext}}_{\mathcal{S}})$.*

7.1 Extensional SK-clones are closed clones

In this section we outline why $\mathbf{SKClone}_{\mathrm{ext}}$ is equivalent to $\mathbf{ClClone}$, thereby giving a category-theoretic equivalence not just between the syntax of $\mathsf{CL}^{\mathrm{wext}}$ and $\Lambda^{\to}$ but also between their models. The proof uses extensionality or the η-law to pass from arbitrary multimaps to nullary ones, from which one can build a *strict closed* clone. We shall rely heavily on the following simple observation.

Lemma 15. *Let $\mathbb{C}$ be a clone and $X := \big\{X(\Gamma; B)\big\}_{\Gamma \in |\mathbb{C}|^{\star}, B \in |\mathbb{C}|}$ a family of sets together with an isomorphism $\big\{\nu_{\Gamma;A} : \mathbb{C}(\Gamma; A) \to X(\Gamma; A)\big\}_{\Gamma,A}$ between X and the hom-sets of $\mathbb{C}$ in the functor category $\big[|\mathbb{C}|^{\star} \times |\mathbb{C}|, \mathbf{Set}\big]$. Then X acquires a canonical clone structure and ν becomes an isomorphism of clones.*

We now introduce strict closed clones.

Definition 15. *A* strict closed clone *is a closed clone $(\mathbb{C}, \Rightarrow, \mathrm{eval})$ such that every $\Lambda^{A} : \mathbb{C}(\Gamma, A; B) \to \mathbb{C}(\Gamma, A \Rightarrow B)$ is the identity. We write $\iota : \mathbf{ClClone}_{\mathrm{st}} \hookrightarrow \mathbf{ClClone}$ for the full subcategory consisting of just the strict closed clones.*

Any closed clone $(\mathbb{C}, \Rightarrow, \mathrm{eval})$ determines a strict closed clone $\mathrm{S}\mathbb{C}$ and a clone isomorphism $\lambda_{\mathbb{C}} : \mathbb{C} \to \mathrm{S}\mathbb{C}$ by applying Lemma 15 to the isomorphisms $\mathbb{C}(\Gamma; B) \cong \mathbb{C}(\diamond; \Gamma \Rightarrow B)$ arising from the closed structure. This extends to a functor $\mathrm{S} : \mathbf{ClClone} \to \mathbf{ClClone}_{\mathrm{st}}$ sending $f : (\mathbb{C}, \Rightarrow, \mathrm{eval}) \to (\mathbb{D}, \Rightarrow, \mathrm{eval})$ to

the composite $\lambda_{\mathbb{D}} \circ f \circ \lambda_{\mathbb{C}}^{-1}$. A short calculation shows that the isomorphisms λ make $\mathrm{S} : \mathbf{ClClone} \leftrightarrows \mathbf{ClClone}_{\mathrm{st}} : \iota$ into an equivalence of categories.

We play a similar game for turning extensional SK-clones into (strict) closed clones. Indeed, for any extensional SK-clone we have isomorphisms $\mathbb{C}(\Gamma; B) \cong \mathbb{C}(\diamond; [\Gamma; B])$ defining a strict closed clone $\mathrm{L}\mathbb{C}$ with $(\mathrm{L}\mathbb{C})(\Gamma; B) := \mathbb{C}(\diamond; [\Gamma; B])$, and hence a functor $\mathrm{L} : \mathbf{SKClone}_{\mathrm{ext}} \to \mathbf{ClClone}_{\mathrm{st}}$ in a similar fashion to S.

Finally, for any closed clone $(\mathbb{C}, \Rightarrow, \mathrm{eval})$ we get an extensional SK-clone $\mathrm{E}\mathbb{C}$ with the same underlying clone by taking application to be application in $\Lambda^{\to}$, so $t \cdot u := \mathrm{eval}_{A,B}[t, u]$, and encoding the combinators as usual.

Theorem 1. *There exist equivalences of categories*

$$\mathbf{SKClone}_{\mathrm{ext}} \underset{\mathrm{E}' := \mathrm{E} \circ \iota}{\overset{\mathrm{L}}{\underset{\simeq}{\rightleftarrows}}} \mathbf{ClClone}_{\mathrm{st}} \overset{\iota}{\underset{\mathrm{S}}{\overset{\simeq}{\rightleftarrows}}} \mathbf{ClClone}.$$

8 A categorical model of $\Lambda^{\to}$

In Propositions 1 and 4 we recovered a unary semantic interpretation of $\Lambda^{\times}$ and $\Lambda^{\times,\to}$ from our clone-theoretic ones. But we do not have a corresponding result for $\Lambda^{\to}$. In this section we fill this gap: we introduce *SK-categories* and show they play the role for $\Lambda^{\to}$ that cartesian closed categories play for $\Lambda^{\times,\to}$. Our definition is inspired by *closed categories* ([11,10]), which axiomatise an 'internal' version of the hom-functor $\mathcal{C}(-,=)$ in the form of a functor $[-,=] : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$. Closed categories have a unit object, corresponding to requiring a unit type (*cf.* [31]); our definition avoids this (see also [39,43]).

Recall that in the presence of contravariance, *dinaturality* and *extranaturality* are the right replacements for naturality (see *e.g.* [30, §IX.4]).

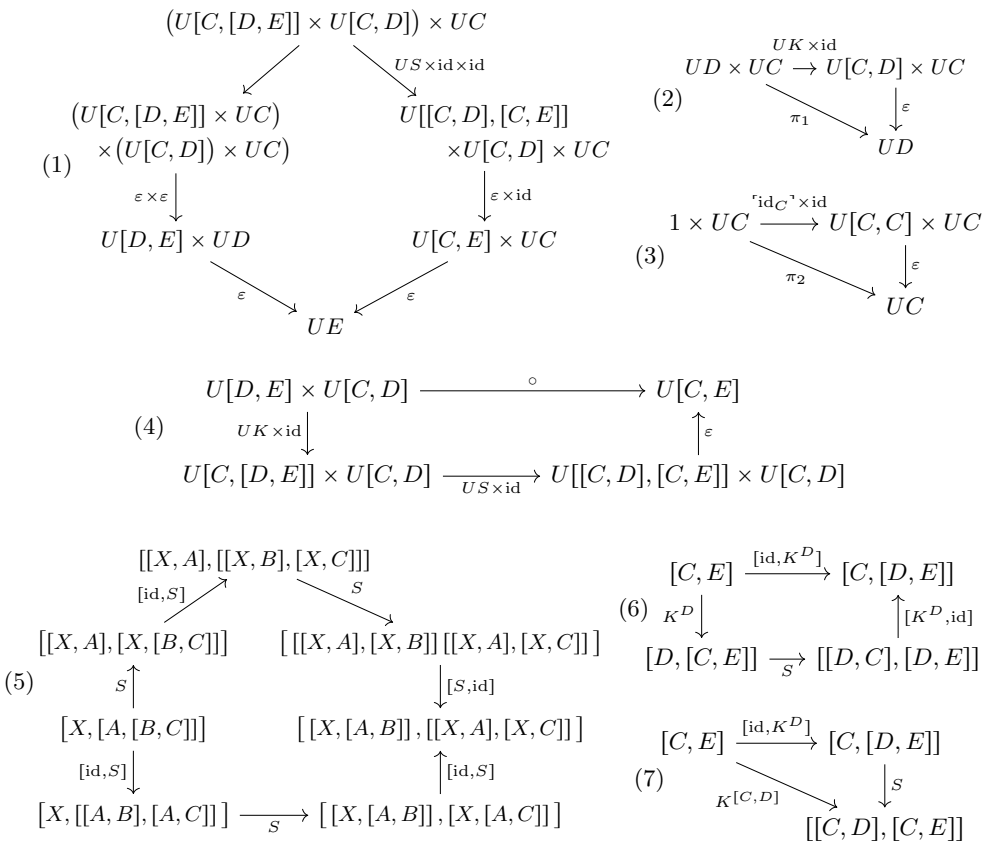
Definition 16. *An* SK-category *consists of a category $\mathcal{C}$ and functors $[-,=] : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ and $U : \mathcal{C} \to \mathbf{Set}$, together with*

1. *Maps $S_{C,D,E} : [C,[D,E]] \to [[C,D],[C,E]]$ dinatural in C and natural in D and E;*
2. *Maps $K_D^C : D \to [C,D]$ extranatural in C and natural in D;*
3. *Maps $\varepsilon_{C,D} : U[C,D] \times \mathrm{U}C \to UD$ extranatural in C and natural in D;*

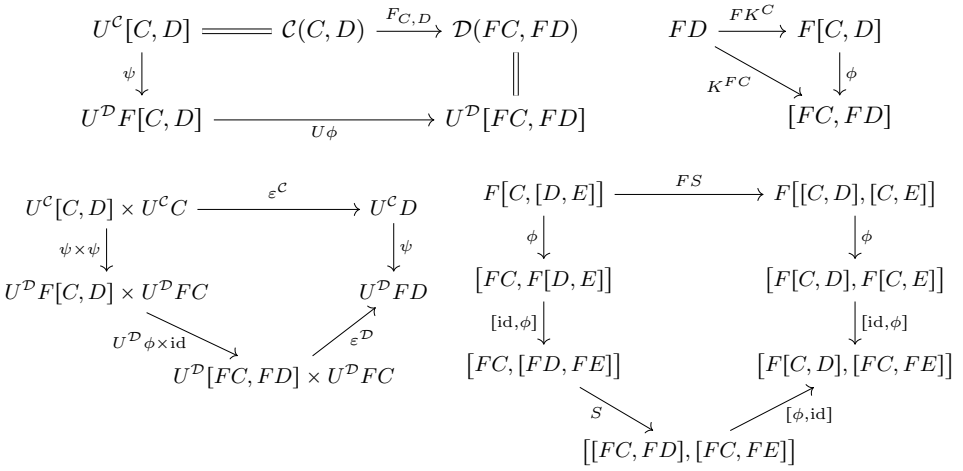
This data is subject to the condition that $U \circ [-,=] = \mathcal{C}(-,=) : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathbf{Set}$ and the 7 axioms of Figure 1a. An SK-functor *(F, ϕ, ψ) is a functor $F : \mathcal{C} \to \mathcal{D}$ with natural transformations as below, such that the axioms of Figure 1b hold.*

$$\begin{array}{ccc} \mathcal{C}^{\mathrm{op}} \times \mathcal{C} & \xrightarrow{F^{\mathrm{op}} \times F} & \mathcal{D}^{\mathrm{op}} \times \mathcal{C} \\ {\scriptstyle [-,=]}\downarrow & \overset{\phi}{\Longrightarrow} & \downarrow{\scriptstyle [-,=]} \\ \mathcal{C} & \xrightarrow[F]{} & \mathcal{D} \end{array} \qquad \begin{array}{ccc} \mathcal{C} & \xrightarrow{\quad F \quad} & \mathcal{D} \\ & {\scriptstyle U}\searrow \quad \overset{\psi}{\Longrightarrow} \quad \swarrow{\scriptstyle U} & \\ & \mathbf{Set} & \end{array}$$

We call (F, ϕ, ψ) strict *if ϕ is the identity, and write* **SKCat** *for the category of SK-categories and strict SK-functors.*



(a) Axioms for an SK-category. In (1) the unlabelled arrow is the canonical map $\langle\langle \pi_1\pi_1, \pi_2\rangle, \langle \pi_2\pi_1, \pi_2\rangle\rangle : (X \times Y) \times Z \to (X \times Z) \times (X \times Z)$. In (3) we write $\ulcorner \mathrm{id}_C \urcorner$ for the set map $* \mapsto \mathrm{id}_C : 1 \to \mathrm{U}[C,C]$.



(b) Axioms for an SK-functor

Fig. 1: Extra axioms for Definition 16

We think of UC as the set of multimaps $\diamond \to C$ and ε as a formal application operation $(- \cdot =)$. Axioms (1) and (2) are the weak equality laws from CL. Axioms (3) and (4) ensure compatibility between the category structure and the corresponding CL constructions: for example, axiom (3) implies $U(f)(x) = f \cdot x$, and axiom (4) says that composition coincides with $\mathsf{S}\,(\mathsf{K}\,-)\,(=)$, corresponding to the weak equality $\mathsf{S}\,(\mathsf{K}\,f)\,g\,x = f\,(g\,x)$. Axioms (5) – (7) are coherence laws.

Every extensional SK-clone determines an SK-category. Because we follow [11] and ask for an *equality* $U[A,B] = \mathcal{C}(A,B)$ in the definition of SK-categories, but in general an extensional SK-clone $(\mathbb{C}, [-,=], \mathsf{S}, \mathsf{K}, \cdot)$ only has an *isomorphism* $\mathbb{C}(A;B) \cong \mathbb{C}(\diamond; [A,B])$, we need to strictify in the same manner as Section 7.1. As a notational shorthand, we write I, B and B' for the closed multimaps satisfying the equations below in the internal language of $\mathbb{C}$ (see *e.g.* [15,6]):

$$\mathsf{I}^A \cdot x = x \ , \ \mathsf{B}^{B\Rightarrow C, A\Rightarrow B, A} \cdot x \cdot y \cdot z = x \cdot (y \cdot z) \ , \ (\mathsf{B}')^{A\Rightarrow B, B\Rightarrow C, A} \cdot x \cdot y \cdot z = y \cdot (x \cdot z)$$

The category $\mathrm{N}\mathbb{C}$ has objects $|\mathbb{C}|$ and hom-sets $(\mathrm{N}\mathbb{C})(A,B) := \mathbb{C}(\diamond; [A,B])$ (*cf.* [14]). The identity on A is I_A and the composite of t and t' is $\mathsf{B} \cdot t \cdot t'$. For U we take $UA := \mathbb{C}(\diamond; A)$ with the action on maps given by application. For $[-,=]$ the action on objects is given by the SK-structure, with the action on maps given by $[X,t] := \mathsf{B} \cdot t$ and $[t,X] := \mathsf{B}' \cdot t$. The maps S and K are given by the corresponding combinators, and ε is the application operation in $\mathbb{C}$. This extends to a functor $\mathrm{N} : \mathbf{SKClone}_{\mathrm{ext}} \to \mathbf{SKCat}$.

The internal language of SK-categories is $\mathsf{CL}^{\mathrm{wext}}$, and hence $\Lambda^{\to}$. We write U for the functor which sends an SK-category $(\mathcal{C}, U, [-,=], S, K, \varepsilon)$ to the signature with base types $|\mathcal{C}|$ and constants $U[\Gamma, B]$.

Proposition 5. *The forgetful functor* $\mathrm{U} : \mathbf{SKCat} \to \mathbf{Sig}$ *has a left adjoint, and the free SK-category on* $\mathcal{S}$ *is* $\mathrm{N}\big(\mathrm{Syn}(\mathsf{CL}^{\mathrm{wext}}_{\mathcal{S}})\big) \cong (\mathrm{N} \circ \mathrm{E})\big(\mathrm{Syn}(\Lambda^{\to}_{\mathcal{S}})\big)$.

Using Theorem 1, we now obtain a version of Propositions 1 and 4 for $\Lambda^{\to}$.

Theorem 2. *The composite* $\mathrm{N} \circ \iota : \mathbf{ClClone}_{\mathrm{st}} \to \mathbf{SKCat}$ *is invertible; hence we get the diagram below, in which the right-hand adjunction is an equivalence:*

$$\mathbf{Sig} \underset{\mathrm{U}}{\overset{\mathrm{F}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathbf{ClClone} \underset{\mathrm{Cl}}{\overset{\mathrm{N}\circ\mathrm{E}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\simeq\bot}}}} \mathbf{SKCat}$$

Moreover, $\mathrm{U} \circ \mathrm{Cl}$ *is equal to the forgetful functor* $\mathbf{SKCat} \to \mathbf{Sig}$*, so the free SK-category on* $\mathcal{S}$ *is canonically isomorphic to* $(\mathrm{N} \circ \mathrm{E})(\mathrm{Syn}(\Lambda^{\to}_{\mathcal{S}}))$.

Recall that a *closed monoidal category* is a monoidal category $(\mathcal{D}, \otimes, I)$ such that every $(-) \otimes D$ has a right adjoint $[D, -]$, and that in a closed category $\mathcal{C}$ giving every $[C, -]$ a $\mathcal{C}$-enriched left adjoint is equivalent to giving closed monoidal structure ([11,10,43]). Theorem 2 and Proposition 4 imply a cartesian version.

Corollary 2. *Equipping a category* $\mathcal{C}$ *with cartesian closed structure is equivalent to equipping* $\mathcal{C}$ *with SK-structure and natural isomorphisms* $\mathcal{C}(I, [C,D]) \cong \mathcal{C}(C,D)$ *and* $\mathcal{C}(C \otimes D, E) \cong \mathcal{C}(C, [D,E])$ *for every* $C, D, E \in \mathcal{C}$.

Acknowledgements. I thank Nathanael Arkor and Dylan McDermott for useful discussions on early drafts of this paper, and the reviewers for their many useful comments. I am grateful to Nayan Rajesh for pointing out the adjunctions between cartesian categories and cartesian clones, and between cartesian closed categories and cartesian closed clones, are in fact equivalences. Finally, I thank Marcelo Fiore for introducing me to clones.

References

1. Abramsky, S.: Computational interpretations of linear logic. Theoretical Computer Science **111**(1-2), 3–57 (1993). https://doi.org/10.1016/0304-3975(93)90181-r
2. Abramsky, S.: Temperley–Lieb algebra: From knot theory to logic and computation via quantum mechanics. In: Mathematics of Quantum Computation and Quantum Technology. Chapman and Hall/CRC (2007)
3. Arkor, N., Fiore, M.: Algebraic models of simple type theories. In: Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science. ACM (2020). https://doi.org/10.1145/3373718.3394771
4. Arkor, N., McDermott, D.: Abstract clones for abstract syntax. In: Kobayashi, N. (ed.) 6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference). LIPIcs, vol. 195, pp. 30:1–30:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.FSCD.2021.30
5. Barendregt, H.P.: The lambda calculus: its syntax and semantics, Studies in Logic and the Foundations of Mathematics), vol. 103. North-Holland (1985), revised edition
6. Bimbó, K.: Combinatory logic pure, applied, and typed. Taylor & Francis (2012)
7. Blanco, N., Zeilberger, N.: Bifibrations of polycategories and classical linear logic. Electronic Notes in Theoretical Computer Science **352**, 29–52 (Oct 2020). https://doi.org/10.1016/j.entcs.2020.09.003
8. Church, A.: A formulation of the simple theory of types. The Journal of Symbolic Logic **5**(2), 56–68 (1940), http://www.jstor.org/stable/2266170
9. Crole, R.L.: Categories for Types. Cambridge University Press (1994). https://doi.org/10.1017/CBO9781139172707
10. Day, B.J., Laplaza, M.L.: On embedding closed categories. Bulletin of the Australian Mathematical Society **18**(3), 357–371 (1978). https://doi.org/10.1017/s0004972700008236
11. Eilenberg, S., Kelly, G.M.: Closed categories. In: Proceedings of the Conference on Categorical Algebra, pp. 421–562. Springer Berlin Heidelberg (1966). https://doi.org/10.1007/978-3-642-99902-4_22
12. Fiore, M., Mahmoud, O.: Second-order algebraic theories. In: Mathematical Foundations of Computer Science 2010, pp. 368–380. Springer Berlin Heidelberg (2010). https://doi.org/10.1007/978-3-642-15155-2_33
13. Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. In: Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science. pp. 193–. LICS '99, IEEE Computer Society, Washington, DC, USA (1999), http://dl.acm.org/citation.cfm?id=788021.788948
14. Fox, T.: Combinatory logic and cartesian closed categories. Master's thesis, McGill University (1971), https://escholarship.mcgill.ca/concern/theses/6h440t871

15. Gilezan, S.: A note on typed combinators and typed lambda terms. Novi Sad Journal of Mathematics **23**(1), 319–329 (1993), https://sites.dmi.uns.ac.rs/nsjom/Papers/23_1/NSJOM_23_1_319_329.pdf
16. Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and Types. Cambridge University Press, New York, NY, USA (1989), http://www.paultaylor.eu/stable/Proofs+Types.html
17. Hermida, C.: Representable multicategories. Advances in Mathematics **151**(2), 164–225 (2000). https://doi.org/https://doi.org/10.1006/aima.1999.1877
18. Hyland, M.: Towards a notion of lambda monoid. Electronic Notes in Theoretical Computer Science **303**, 59–77 (2014). https://doi.org/10.1016/j.entcs.2014.02.004
19. Hyland, M.: Classical lambda calculus in modern dress. Mathematical Structures in Computer Science **27**(5), 762–781 (2015). https://doi.org/10.1017/s0960129515000377
20. Hyland, M., de Paiva, V.: Full intuitionistic linear logic (extended abstract). presented at the 9th International Congress of Logic, Methodology and Philosophy of Science held in Uppsala, Sweden, August 7-14, 1991. Annals of Pure and Applied Logic **64**(3), 273–291 (1993). https://doi.org/10.1016/0168-0072(93)90146-5
21. Jacobs, B.: Simply typed and untyped lambda calculus revisited. In: Applications of Categories in Computer Science, pp. 119–142. Cambridge University Press (1992). https://doi.org/10.1017/cbo9780511525902.008
22. Jacobs, B.: Categorical logic and type theory. Elsevier Science (1999)
23. Johnstone, P.T.: Sketches of an Elephant: A Topos Theory Compendium Volume 2 (Oxford Logic Guides). Clarendon Press (2002)
24. Lambek, J.: The mathematics of sentence structure. The American Mathematical Monthly **65**(3), 154 (1958). https://doi.org/10.2307/2310058
25. Lambek, J.: Deductive systems and categories II: Standard constructions and closed categories. In: Category theory, homology theory and their applications I, pp. 76–122. Springer (1969). https://doi.org/10.1007/BFb0079385
26. Lambek, J.: Multicategories revisited. In: Gray, J.W., Scedrov, A. (eds.) Categories in Computer Science and Logic: Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference Held June 14-20, 1987 with Support from the National Science Foundation, vol. 92, pp. 217–240. American Mathematical Society (1989). https://doi.org/10.1090/conm/092
27. Lambek, J., Scott, P.J.: Introduction to Higher Order Categorical Logic. Cambridge University Press, New York, NY, USA (1986)
28. Leinster, T.: Higher operads, higher categories. No. 298 in London Mathematical Society Lecture Note Series, Cambridge University Press (2004). https://doi.org/10.1017/CBO9780511525896
29. Linton, F.E.J.: Some aspects of equational categories. In: Proceedings of the Conference on Categorical Algebra, pp. 84–94. Springer Berlin Heidelberg (1966). https://doi.org/10.1007/978-3-642-99902-4_3
30. Mac Lane, S.: Categories for the Working Mathematician, Graduate Texts in Mathematics, vol. 5. Springer-Verlag New York, second edn. (1998). https://doi.org/10.1007/978-1-4757-4721-8
31. Manzyuk, O.: Closed categories vs. closed multicategories. Theory and Applications of Categories **26**(5), 132–175 (2012), http://www.tac.mta.ca/tac/volumes/26/5/26-05.pdf
32. May, J.P.: The Geometry of Iterated Loop Spaces. Springer Berlin Heidelberg (1972). https://doi.org/10.1007/bfb0067491
33. Melliès, P.A.: Ribbon tensorial logic. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '18, ACM (Jul 2018). https://doi.org/10.1145/3209108.3209129

34. Pisani, C.: Sequential multicategories. Theory and Applications of Categories **29**(19), 496—541 (2014), http://www.tac.mta.ca/tac/volumes/29/19/29-19.pdf
35. Pitts, A.M.: Categorical logic. In: Handbook of Logic in Computer Science, chap. 2, pp. 39–123. Oxford University Press, Oxford, UK (2000)
36. Polakow, J., Pfenning, F.: Natural deduction for intuitionistic non-commutative linear logic. In: Lecture Notes in Computer Science, pp. 295–309. Springer Berlin Heidelberg (1999). https://doi.org/10.1007/3-540-48959-2_21
37. Riehl, E.: Category Theory in Context. Dover Publications, Incorporated (2016), https://math.jhu.edu/~eriehl/context.pdf
38. Saville, P.: Cartesian closed bicategories: type theory and coherence. Ph.D. thesis, University of Cambridge (2020). https://doi.org/10.17863/CAM.55080
39. Shulman, M.: Closed category, https://ncatlab.org/nlab/show/closed+category, revision 49 (May 2018)
40. Shulman, M.: LNL polycategories and doctrines of linear logic. Logical Methods in Computer Science **19**(2) (2023). https://doi.org/10.46298/lmcs-19(2:1)2023
41. Taylor, W.: Characterizing Mal'cev conditions. Algebra Universalis **3**(1), 351 (Dec 1973). https://doi.org/10.1007/BF02945141
42. Troelstra, A.S., Schwichtenberg, H.: Basic proof theory. No. 43 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, second edn. (2000)
43. Uustalu, T., Veltri, N., Zeilberger, N.: Eilenberg-Kelly reloaded. Electronic Notes in Theoretical Computer Science **352**, 233–256 (Oct 2020). https://doi.org/10.1016/j.entcs.2020.09.012
44. Uustalu, T., Veltri, N., Zeilberger, N.: Deductive systems and coherence for skew prounital closed categories. In: Sacerdoti Coen, C., Tiu, A. (eds.) Proceedings Fifteenth Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, Paris, France, 29th June 2020. Electronic Proceedings in Theoretical Computer Science, vol. 332, pp. 35–53. Open Publishing Association (2021). https://doi.org/10.4204/EPTCS.332.3
45. Weber, M.: Free products of higher operad algebras. Theory and Applications of Categories **28**(2), 24–65 (2013), http://www.tac.mta.ca/tac/volumes/28/2/28-02.pdf
46. Zeilberger, N., Giorgetti, A.: A correspondence between rooted planar maps and normal planar lambda terms. Logical Methods in Computer Science **11** (2015). https://doi.org/10.2168/lmcs-11(3:22)2015