

ENRICHED LAWVERE THEORIES

Dedicated to Jim Lambek

JOHN POWER

ABSTRACT. We define the notion of enriched Lawvere theory, for enrichment over a monoidal biclosed category V that is locally finitely presentable as a closed category. We prove that the category of enriched Lawvere theories is equivalent to the category of finitary monads on V . Moreover, the V -category of models of a Lawvere V -theory is equivalent to the V -category of algebras for the corresponding V -monad. This all extends routinely to local presentability with respect to any regular cardinal. We finally consider the special case where V is \mathbf{Cat} , and explain how the correspondence extends to pseudo maps of algebras.

1. Introduction

In seeking a general account of what have been called notions of computation [13], one may consider a finitary 2-monad T on \mathbf{Cat} , and a T -algebra (A, a) , then make a construction of a category B and an identity on objects functor $j : A \rightarrow B$. In making that construction, one is allowed to use the structure on A determined by the 2-monad, and that is all. For instance, given the 2-monad for which an algebra is a small category with a monad on it, then one possible construction would be the Kleisli construction. For another example, given the 2-monad for which an algebra is a small monoidal category A together with a specified object S , then a possible construction is that for which $B(a, b)$ is defined to be $A(S \otimes a, S \otimes b)$, with the functor $j : A \rightarrow B$ sending a map h to $S \otimes h$.

We need a precise statement of what we mean by saying that these constructions only use structure determined by the 2-monad. In general, one may obtain one such definition by asserting that $B(a, b)$ must be of the form $A(fa, gb)$ for endofunctors f and g on A generated by the 2-monad; similarly for defining composition in B . Thus we need to know what exactly we mean by endofunctors on A and natural transformations between them that are generated by the 2-monad. This paper is motivated by a desire to make such notions precise.

In order to make precise the notion of functor generated by a finitary 2-monad T on a T -algebra (A, a) , we first generalise from consideration of a finitary 2-monad on \mathbf{Cat} to a finitary V -monad on V for any monoidal biclosed category that is locally finitely

This work has been done with the support of EPSRC grants GR/J84205: Frameworks for programming language semantics and logic and R34723: The Structure of Programming Languages: syntax and semantics.

Received by the editors 1998 December 14 and, in revised form, 1999 August 27.

Published on 1999 November 30.

1991 Mathematics Subject Classification: 18C10, 18C15, 18D05.

Key words and phrases: Lawvere theory, monad.

© John Power 1999. Permission to copy for private use granted.

presentable as a closed category: we shall make that definition precise in the next section. We move to this generality primarily for simplicity but also because the computing application will need it later, for instance considering not mere categories but categories enriched in *Poset* or in the category of ω -cpo's.

To support our main definition, we prove a theorem: we define the notion of a Lawvere V -theory, and we prove that to give a Lawvere V -theory is equivalent to giving a finitary V -monad on V . So, for a finitary V -monad T on V , there is a Lawvere V -theory $\mathcal{L}(T)$ for which the V -category of models of $\mathcal{L}(T)$, which we define, is equivalent to the V -category $T\text{-Alg}$ of algebras for the monad. Conversely, given any Lawvere V -theory \mathcal{T} , there is a corresponding finitary V -monad $M(\mathcal{T})$; and these constructions yield an equivalence between the category of finitary V -monads on V and that of Lawvere V -theories.

Once we have such a correspondence, we have the definition we seek: given a 2-monad T and a T -algebra (A, a) , an endofunctor f on A is generated by T whenever it is in the image of $\mathcal{L}(T)$ in the model of $\mathcal{L}(T)$ determined by the algebra (A, a) . Similarly, a natural transformation $\alpha : f \Rightarrow g$ is said to be generated by T whenever it is the image of a 2-cell of $\mathcal{L}(T)$ in the model of $\mathcal{L}(T)$ determined by the algebra (A, a) .

For cognoscenti of enriched categories, the definitions and results here, at least assuming our enrichment is over a symmetric monoidal category, should come as no great surprise. However, they are a little subtle. One's first guess for a definition of Lawvere V -theory may be a small V -category with finite products, subject to a condition asserting single-sortedness. But that is too crude: not only does it not yield our theorem relating Lawvere theories and finitary monads, as in the case for *Set* (see [1]), but it does not agree with the known and very useful equivalences between finitary monads and universal algebra as in [9], [6], and in application to computation in [11] and [10], with an account for computer scientists in [14]. So we consider something a little more subtle: if V is symmetric, we define a Lawvere V -theory to be a small V -category with finite cotensors, subject to a single-sortedness condition. A model is therefore a finite cotensor preserving V -functor into V . In order to extend to nonsymmetric V , as for instance if V is the category of small locally ordered categories with the lax Gray tensor product [11], we need to introduce a twist: so a Lawvere V -theory is a small V_t -category with finite cotensors, subject to a single-sortedness condition, and a model is a finite cotensor preserving V_t -functor into V_t : that allows us to speak of the V -category of models, and prove our equivalence with finitary V -monads on V and their V -categories of algebras. There has been considerable development of categories enriched in monoidal biclosed categories [4, 5, 6, 11] recently: the main problem with extending the usual theory is that functor V -categories need not exist in general; but functor V -categories with base V_t do, and that suffices for our purposes here. If one is only interested in the case of symmetric V here, one may simply ignore all subscripts t , and one will have correct statements; of course, exponentials, i.e., terms of the form $[X, -]_l$ and $[X, -]_r$, may also be abbreviated to $[X, -]$.

The special case that $V = \text{Cat}$ also contains a mild surprise. Given a 2-monad T on *Cat*, one's primary interest lies in pseudo maps of algebras, i.e., maps of algebras in which the structure need only be preserved up to coherent isomorphism rather than

strictly. Such maps are analysed in [2], with an explanation for computer scientists in [12]. One might imagine that, extending the above correspondence in this case, pseudo maps of algebras would correspond exactly to pseudo natural transformations between the corresponding models of the corresponding Lawvere 2-theory. But the position is a little more delicate than that: pseudo natural transformations allow too much flexibility to yield a bijection. However, given a pair of algebras, one still has an equivalence between the category of pseudo maps of algebras between the two algebras and that of pseudo natural transformations between the corresponding models of the corresponding Lawvere 2-theory. Hence one has a biequivalence between $T\text{-Alg}_p$ and the 2-category of models of the Lawvere theory, with pseudo natural transformations. We explain that in Section 5.

I am surprised I have not been able to find our main results, in the case of symmetric V , in the literature on enriched categories. There has been work on enriched monads, for instance [9] and [6], and there has been work on enriched finite limit theories, primarily [8] and [5]. But the closest work to this of which I am aware is by Dubuc [3], and that does not account for the finitary nature of Lawvere theories, which is central for us. Of course, what we say here is not explicitly restricted to finitariness: one could easily extend everything to cardinality $< \kappa$ for any regular κ .

The paper is organized as follows. In Section 2, we recall the basic facts about enriched categories we shall use to define our terms. In Section 3, we define Lawvere V -theories and their models, and prove that every finitary V -monad on V gives rise to a Lawvere V -theory with the same V -category of models. And in Section 4, we give the converse, i.e., to each Lawvere V -theory, we discover a finitary V -monad with the same V -category of models. We also prove the equivalence between the categories of Lawvere V -theories and finitary V -monads on V . Finally, in Section 5, we explain the more fundamental pseudo maps of algebras when $V = \text{Cat}$, and show how they relate to maps of Lawvere 2-theories.

For symmetric V , the standard reference for all basic structures other than monads is Kelly's book [7]. For nonsymmetric V , a reasonable reference is [11], but the central results were proved in the more general setting of categories enriched in bicategories as in [4, 5, 6]. For 2-categories, the best relevant reference is [2], and for an explanation directed towards computer scientists, see [12].

2. Background on enriched categories

A monoidal category V is called *biclosed* if for every object X of V , both $- \otimes X: V_0 \rightarrow V_0$ and $X \otimes -: V_0 \rightarrow V_0$ have right adjoints, denoted $[X, -]_r$ and $[X, -]_l$ respectively. For monoidal biclosed locally small V , it is evident how to define *V -categories*, *V -functors* and *V -natural transformations*, yielding the 2-category $V\text{-Cat}$ of small V -categories. The category V_0 enriches to a V -category with hom given by $[X, Y]_r$. Note that $[X, Y]_r$ cannot be replaced by $[X, Y]_l$ here, using the usual conventions of Kelly's book [7]. One can speak of *representable* V -functors and there is an elegant theory of V -adjunctions, see for instance [4]. If V_0 is complete, then it is shown in [7] that $V\text{-Cat}$ is complete. So we can

speak of the Eilenberg-Moore V -category for a V -monad. If V is also cocomplete, there is an elegant theory of limits and colimits in V -categories generalising the situation for symmetric monoidal closed V , see for instance [5]. Note that for a V -category C , there is a construction of what should clearly be called C^{op} , but C^{op} is a V_t -category, not a priori a V -category.

In general, there is no definition of a functor V -category. However, if V is complete, for a small V -category C , one does have a functor V -category of the form $[C^{op}, V_t]$, whose objects are V_t -functors from C^{op} into V_t , and with homs given by the usual construction for symmetric V . Details appear in [5], where it is denoted PC . This also agrees with Street's construction [16], but the latter is formulated in terms of modules.

A V -monad on a V -category C consists of a V -functor $T: C \rightarrow C$ and V -natural transformations $\eta: 1 \Rightarrow T$ and $\mu: T^2 \Rightarrow T$ satisfying the usual three axioms. The Eilenberg-Moore V -category $T\text{-Alg}$ has as objects the T_\circ -algebras, where T_\circ is the ordinary monad underlying T , on the ordinary category C_\circ underlying C . Given algebras (A, a) and (B, b) , the hom object $T\text{-Alg}((A, a), (B, b))$ is the equaliser in V of the diagram

$$\begin{array}{ccc}
 C(A, B) & \xrightarrow{C(a, B)} & C(T(A), B) \\
 & \searrow T & \nearrow C(T(A), b) \\
 & C(T(A), T(B)) &
 \end{array} \tag{1}$$

Composition in $T\text{-Alg}$ is determined by that in C . Moreover, there is a forgetful V -functor $U: T\text{-Alg} \rightarrow C$, which has a left adjoint. If C is complete, then so is $T\text{-Alg}$, and U preserves limits.

Given monads T and S on C , a *map of monads* from T to S is a V -natural transformation $\alpha: T \Rightarrow S$ that commutes with the unit and multiplication data of the monads. With the usual composition of V -natural transformations, this yields the ordinary category $Mnd(C)$ of monads on C .

A monoidal biclosed category V is called *locally finitely presentable as a closed category* if V_\circ is locally finitely presentable, if the unit I of V is finitely presentable, and if $x \otimes y$ is finitely presentable whenever x and y are. Henceforth, all monoidal biclosed categories to which we refer will be assumed to be locally finitely presentable as closed categories.

A V -category C is said to have *finite tensors* if for every finitely presentable object x of V , and for every object A of C , $[x, C(A, -)]_r: C \rightarrow V$ is representable, i.e., if there exists an object $x \otimes A$ of C together with a natural isomorphism $[x, C(A, -)]_r \cong C(x \otimes A, -)$. The V -category C has *finite cotensors* if it satisfies the dual condition that for every finitely presentable x in V and every A in C , the V_t -functor $[x, C(-, A)]_l: C^{op} \rightarrow V_t$ is representable, where V_t is the dual of V ; i.e., there exists an object A^x of C together with a natural isomorphism $[x, C(-, A)]_l \cong C(-, A^x)$. A V -category C is *cocomplete* whenever C_\circ is cocomplete, C has finite tensors, and $x \otimes -: C_\circ \rightarrow C_\circ$ preserves colimits for all finitely presentable x . A cocomplete V -category C is *locally finitely presentable* if

C_\circ is locally finitely presentable, C has finite cotensors, and $(-)^x : C_\circ \rightarrow C_\circ$ preserves filtered colimits for all finitely presentable x . In general, a V -functor is called *finitary* if the underlying ordinary functor is so, i.e., if it preserves filtered colimits. We therefore denote the full subcategory of $Mnd(C)$ determined by the finitary monads on C by $Mnd_f(C)$. In [6], finitary monads on an lfp V -category are characterized in terms of algebraic structure.

For any V that is locally finitely presentable as a closed category, V is necessarily a locally finitely presentable V -category. The cotensor A^X is given by $[X, A]_l$. Given a small finitely cotensored V_t -category C , we write $FC(C, V_t)$ for the full sub- V -category of $[C, V_t]$ determined by those V -functors that preserve finite cotensors. It follows from Freyd's adjoint functor theorem, and from the fact that the inclusion preserves cotensors, that $FC(C, V_t)$ is a full reflective sub- V -category of $[C, V_t]$, i.e., the inclusion has a left adjoint. It follows immediately from the definitions that the inclusion is also finitary.

Note that, in all that follows, if V is symmetric, one may simply disregard all subscripts t , and one will have correct statements, with correct proofs: if V is symmetric, then the symmetric monoidal category V_t is isomorphic to V , and of course $[X, -]_l$ and $[X, -]_r$ agree.

3. Enriched Lawvere theories

We seek a definition of a Lawvere V -theory. In order to validate that definition, we shall prove that the V -category of models of a Lawvere V -theory is finitarily monadic over V ; and for any finitary V -monad T on V , the V -category $T\text{-Alg}$ is the V -category of models of a Lawvere theory. More elegantly, we shall establish an equivalence between the ordinary category of finitary V -monads on V and the category of Lawvere V -theories.

First we shall give our definition of Lawvere V -theory. Observe that the full subcategory V_f of V determined by (the isomorphism classes of) the finitary objects of V has finite tensors given as in V and is small.

3.1. DEFINITION. *A Lawvere V -theory consists of a small V_t -category \mathcal{T} with finite cotensors, together with an identity on objects strictly finite cotensor preserving V_t -functor $\iota : V_f^{op} \rightarrow \mathcal{T}$.*

It is immediate that if $V = Set$, this definition agrees with the classical one. We extend the usual convention for Set by informally referring to \mathcal{T} as a Lawvere V -theory, leaving the identity on objects V_t -functor implicit. A map of Lawvere V -theories from \mathcal{T} to \mathcal{S} is a strict finite cotensor preserving V_t -functor from \mathcal{T} to \mathcal{S} that commutes with the V_t -functors from V_f^{op} . Together with the usual composition of V_t -functors, this yields a category we denote by Law_V .

We now extend the usual definition of a model of a Lawvere theory.

3.2. DEFINITION. *A model of a Lawvere V -theory \mathcal{T} is a finite cotensor preserving V_t -functor from \mathcal{T} to V_t . The V -category of models of \mathcal{T} is $FC(\mathcal{T}, V_t)$, the full sub- V -category of $[\mathcal{T}, V_t]$ determined by the finite cotensor preserving V_t -functors.*

3.3. EXAMPLE. Let T be a V -monad on V . Then, one can speak of the Kleisli V -category $Kl(T)$ for T , and there is a V -functor $j : V \rightarrow Kl(T)$ that is the identity on objects, has a right adjoint, and satisfies the usual universal property of Kleisli constructions, as explained in Street's article [15]. Since j has a right adjoint, it preserves tensors. So if we restrict j to the finitely presentable objects of V , we have an identity on objects (strictly) finite tensor preserving V -functor from V_f to the full sub- V -category $Kl(T)_f$ of $Kl(T)$ determined by the objects of V_f . Dualising, we have a Lawvere V -theory, which we denote by $\mathcal{L}(T)$. This construction extends to a functor $\mathcal{L} : Mnd(V) \rightarrow Law_V$. Since our primary interest is in finitary V -monads, we also use \mathcal{L} to denote its restriction to $Mnd_f(V)$.

3.4. THEOREM. *Let T be a finitary V -monad on V . Then $FC(\mathcal{L}(T), V_t)$ is equivalent to $T\text{-Alg}$.*

Proof. There is a comparison V -functor from $Kl(T)$ to $T\text{-Alg}$, and it preserves tensors since the canonical functors from V into each of $Kl(T)$ and $T\text{-Alg}$ do. By composition with the inclusion of $Kl(T)_f$ into $Kl(T)$, we have a V -functor $c : Kl(T)_f \rightarrow T\text{-Alg}$, and this yields a V -functor $\tilde{c} : T\text{-Alg} \rightarrow [Kl(T)_f^{op}, V_t]$ sending a T -algebra (A, a) to the V_t -functor $T\text{-Alg}(c(-), (A, a))$. This V -functor factors through $FC(Kl(T)_f^{op}, V_t)$ since c preserves finite tensors and since representables preserve finite cotensors. Moreover, it is fully faithful since every object of $T\text{-Alg}$ is a canonical colimit of a diagram in $Kl(T)_f$. So it remains to show that \tilde{c} is essentially surjective.

Suppose $h : Kl(T)_f^{op} \rightarrow V_t$ preserves finite cotensors. Let $A = h(I)$, where I is the unit of V . The behaviour of h on all objects is fully determined by its behaviour on I since h preserves finite cotensors and every object of $Kl(T)_f^{op}$, i.e., every object of V_f^{op} , is a finite cotensor of I . The behaviour of h on homs gives, for each finitely presentable x , a map in V from $Kl(T)(I, x)$ to $[A^x, A]_l$, or equivalently, since cotensors in V are given by a left exponential, and by the usual properties of maps in Kleisli categories and maps from units, a map in V from $[x, A]_l \otimes Tx$ to A . This must all be natural in x , thus, by finitariness of T , we have a map $a : TA \rightarrow A$. Functoriality of h , together with the V_t -functor from V_f^{op} into $Kl(T)^{op}$, force a to be an algebra map. It is routine to verify that $\tilde{c}((A, a))$ is isomorphic to h . ■

4. The converse

In this section, we start by proving a converse to Theorem 3.4. This involves constructing a finitary monad $M(\mathcal{T})$ from a Lawvere V -theory \mathcal{T} . Having proved that result, we establish that \mathcal{L} together with M form an equivalence of categories between the categories of finitary V -monads on V and Lawvere V -theories, for any monoidal biclosed V that is locally finitely presentable as a closed category.

First we recall Beck's monadicity theorem. Given a functor $U : C \rightarrow D$, a pair of arrows $h_1, h_2 : X \rightarrow Y$ in C is called a *U-split coequaliser pair* if there exist arrows $h : UY \rightarrow Z$, $i : Z \rightarrow UY$, and $j : UY \rightarrow UX$ in D such that $h \cdot Uh_1 = h \cdot Uh_2$, i

splits h , j splits h_1 , and $Uh_2 \cdot j = i \cdot h$. It follows that h is a coequaliser of Uh_1 and Uh_2 , and that coequaliser is preserved by all functors. Beck's theorem says

4.1. THEOREM. *A functor $U : C \rightarrow D$ is monadic if and only if*

- U has a left adjoint
- U reflects isomorphisms, and
- C has and U preserves coequalisers of U -split coequaliser pairs.

For a detailed account of Beck's theorem, see Barr and Wells' book [1]. Now we are ready to prove our main result.

4.2. THEOREM. *Let \mathcal{T} be an arbitrary Lawvere V -theory. Then there is a finitary monad $M(\mathcal{T})$ on V such that $M(\mathcal{T}) - \text{Alg}$ is equivalent to $FC(\mathcal{T}, V_t)$.*

Proof. Recall from the definition of Lawvere V -theory, we have a finite cotensor preserving V_t -functor from V_f^{op} to \mathcal{T} . By composition, this yields a V -functor $U : FC(\mathcal{T}, V_t) \rightarrow FC(V_f^{op}, V_t)$, but the latter is equivalent to V : one proof of this is by applying Theorem 3.4 to the identity monad. The V -functor U is given by evaluation at I . Moreover, it has a left adjoint since the inclusion of $FC(\mathcal{T}, V_t)$ into $[\mathcal{T}, V_t]$ has a left adjoint, as mentioned in Section 2, and since evaluation functors from V -presheaf categories have left adjoints given by tensors. Moreover, it is finitary since $(-)^x$ preserves filtered colimits for finitely presentable x . It remains to show that U is monadic.

We shall apply Beck's monadicity theorem. It is routine to verify that U reflects isomorphisms, and one can readily calculate that U preserves the coequalisers of U -split coequaliser pairs: since $FC(\mathcal{T}, V_t)$ is a full reflective sub- V -category of $[\mathcal{T}, V_t]$, it is cocomplete; and the required coequaliser lifts to all objects as they are all finite cotensors of the unit I , so one need merely apply the associated cotensor to the given map and its splitting. That proves monadicity of the underlying ordinary functor of U ; extending that to the enriched functor follows immediately from the fact that U respects cotensors. ■

Our construction M extends routinely to a functor $M : Law_V \rightarrow Mnd_f(V)$. Recall from the previous section that we also have a functor $\mathcal{L} : Mnd_f(V) \rightarrow Law_V$. In fact we have

4.3. THEOREM. *The functors $M : Law_V \rightarrow Mnd_f(V)$ and $\mathcal{L} : Mnd_f(V) \rightarrow Law_V$ form an equivalence of categories.*

Proof. Given a finitary V -monad T on V , it follows from Theorem 3.4 and the construction of M that $M\mathcal{L}(T)$ is isomorphic to T . For the converse, given a Lawvere V -theory \mathcal{T} , we need to study the construction of $M(\mathcal{T})$. It is given by taking the left adjoint to the functor from $FC(\mathcal{T}, V_t)$ to V given by evaluation at the unit I . The left adjoint, applied to a finitely presentable object x of V , yields the representable functor $\mathcal{T}(x, -) : \mathcal{T} \rightarrow V_t$: this follows since x is a finite tensor of I , and by preservation of finite cotensors. Now applying Yoneda yields the result that $\mathcal{L}M(\mathcal{T})$ is isomorphic to \mathcal{T} . Thus we are done. ■

5. 2-Monads on \mathbf{Cat}

We now restrict our attention to the case that $V = \mathbf{Cat}$. So we consider finitary 2-monads on \mathbf{Cat} . An example is that there is a finitary 2-monad for which the algebras are small monoidal categories with a specified object S , as mentioned in the introduction. Another finitary 2-monad on \mathbf{Cat} has an algebra given by a small category with a monad on it, again as in the introduction. There are many variants of these examples: see [2] or [12] for many more examples and more detail.

When one does have a 2-monad T , the maps of primary interest are the *pseudo* maps of algebras. These correspond to the usual notion of structure preserving functor. They are defined as follows.

5.1. DEFINITION. *Given T -algebras (A, a) and (B, b) , a pseudo map of algebras from (A, a) to (B, b) consists of a functor $f : A \rightarrow B$ and a natural isomorphism*

$$\begin{array}{ccc} TA & \xrightarrow{Tf} & TB \\ a \downarrow & \Downarrow^{\bar{f}} & \downarrow b \\ A & \xrightarrow{f} & B \end{array}$$

such that

$$\begin{array}{ccc} T^2 A & \xrightarrow{T^2 f} & TB \\ Ta \downarrow & \Downarrow^{T\bar{f}} & \downarrow Tb \\ TA & \xrightarrow{Tf} & TB \\ a \downarrow & \Downarrow^{\bar{f}} & \downarrow b \\ A & \xrightarrow{f} & B \end{array} = \begin{array}{ccc} T^2 A & \xrightarrow{T^2 f} & TB \\ \mu_A \downarrow & & \downarrow \mu_B \\ TA & \xrightarrow{Tf} & TB \\ a \downarrow & \Downarrow^{\bar{f}} & \downarrow b \\ A & \xrightarrow{f} & B \end{array}$$

and

$$\begin{array}{ccc}
 A & \xrightarrow{f} & B \\
 id_A \downarrow & & \downarrow id_B \\
 A & \xrightarrow{f} & B
 \end{array}
 =
 \begin{array}{ccccc}
 A & \xrightarrow{f} & B & & \\
 \eta_A \downarrow & & \downarrow \eta_B & & \\
 TA & \xrightarrow{Tf} & TB & & \\
 a \downarrow & \Downarrow^{\bar{f}} & b \downarrow & & \\
 A & \xrightarrow{f} & B & &
 \end{array}$$

A 2-cell between pseudo maps (f, \bar{f}) and (g, \bar{g}) is a natural transformation between f and g that respects \bar{f} and \bar{g} .

By using the composition of Cat , it follows that T -algebras, pseudo maps of T -algebras, and 2-cells form a 2-category, which we denote by $T\text{-Alg}_p$. This agrees with the notation in some of the relevant literature, and this situation has undergone extensive study, in particular in [2]; and for an account directed towards computer scientists, see [12].

Similarly, given a small 2-category C , one has the notion of pseudo natural transformation between 2-functors $h, k : C \rightarrow Cat$: one has isomorphisms where the definition of natural transformation has commuting squares, and those isomorphisms are subject to two coherence conditions, expressing coherence with respect to composition and identities in C . Thus, for any Lawvere 2-theory \mathcal{T} , we have the 2-category $FC_p(\mathcal{T}, Cat)$, given by finite cotensor preserving 2-functors, pseudo natural transformations, and the evident 2-cells. If a finitary 2-monad T corresponds to the Lawvere 2-theory \mathcal{T} , one might guess that the 2-equivalence between $T\text{-Alg}$ and $FC(\mathcal{T}, Cat)$ would extend to a 2-equivalence between $T\text{-Alg}_p$ and $FC_p(\mathcal{T}, Cat)$, but it does not!

5.2. EXAMPLE. Let T be the identity 2-monad on Cat . Then $T\text{-Alg}_p$ is 2-equivalent to Cat . But $FC_p(\mathcal{T}, Cat)$ is not, as it contains more maps. First, all functors lie in $FC_p(\mathcal{T}, Cat)$ via the inclusion of Cat , which is 2-equivalent to $FC(\mathcal{T}, Cat)$, in $FC_p(\mathcal{T}, Cat)$. But also, one may vary any component of a pseudo natural transformation α , say α_2 , by an isomorphism, leaving every other component fixed, and vary the structural isomorphisms of α by conjugation, and one still has a pseudo natural transformation.

So the correspondence we seek is a little more subtle. The notion of 2-equivalence between 2-categories is quite rare. More commonly, one has a *biequivalence*. This amounts to relaxing fully faithfulness to an equivalence on homcategories, and similarly for essential surjectivity. In our case, we already have essential surjectivity. So we could ask whether our 2-functor $\tilde{c} : T\text{-Alg} \rightarrow FC(\mathcal{T}, Cat)$ extends to giving, for each pair of algebras $((A, a), (B, b))$, an equivalence between $T\text{-Alg}_p((A, a), (B, b))$ and the category of

pseudo natural transformations between $\tilde{c}((A, a))$ and $\tilde{c}((B, b))$, hence a biequivalence of 2-categories. In fact, we have

5.3. THEOREM. *The 2-functor $\tilde{c} : T\text{-Alg} \longrightarrow FC(\mathcal{T}, \text{Cat})$ extends to a biequivalence between $T\text{-Alg}_p$ and $FC_p(\mathcal{T}, \text{Cat})$.*

Proof. If one routinely follows the argument for essential surjectivity of \tilde{c} as in the proof of Theorem 3.4, and ones uses the 2-dimensional property of the colimit defining TA , one obtains a bijection between pseudo maps of T -algebras from (A, a) to (B, b) , and those pseudo natural transformations between $\tilde{c}((A, a))$ and $\tilde{c}((B, b))$ that respect finite cotensors strictly, i.e., up to equality, not just isomorphism. Now, since cotensors are pseudo limits, they are automatically bilimits, and so every pseudo natural transformation is isomorphic to one that respects finite cotensors strictly. Local fully faithfulness is straightforward. ■

References

- [1] M. Barr and C. Wells, Toposes, Triples, and Theories, Grundlehren der math. Wissenschaften 278, Springer-Verlag (1985).
- [2] R. Blackwell, G.M. Kelly, and A.J. Power, Two-dimensional monad theory. J. Pure Appl. Algebra 59 (1989) 1–41.
- [3] E. Dubuc, Kan extensions in enriched category theory, Lecture Notes in Math. 145, Springer-Verlag (1970).
- [4] R. Gordon and A.J. Power, Enrichment through variation, J. Pure Appl. Algebra 120 (1997) 167–185.
- [5] R. Gordon and A.J. Power, Gabriel Ulmer duality in categories enriched in bicategories, J. Pure Appl. Algebra 137 (1999) 29–48.
- [6] R. Gordon and A.J. Power, Algebraic structure on bicategory enriched categories, J. Pure Appl. Algebra 130 (1998) 119–132.
- [7] G.M. Kelly, Basic concepts of enriched category theory, London Math. Soc. Lecture Notes Series 64, Cambridge University Press (1982).
- [8] G.M. Kelly, Structures defined by finite limits in the enriched context 1, Cahiers de Top et Geom Differentielle 23 (1980) 3–42.
- [9] G.M. Kelly and A.J. Power, Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads, J. Pure Appl. Algebra 89 (1993) 163–179.

- [10] Y. Kinoshita, P.W. O'Hearn, A.J. Power, M. Takeyama, and R.D. Tennent, An axiomatic approach to binary logical relations with applications to data refinement, Proc Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 1281, Springer (1997) 191–212.
- [11] Y. Kinoshita and A.J. Power, Lax naturality through enrichment, *J. Pure Appl. Algebra* 112 (1996) 53–72.
- [12] A.J. Power, Why tricategories? *Information and Computation* 120 (1995) 251–262.
- [13] A.J. Power, Modularity in Denotational Semantics, Proc. Mathematical Foundations of Programming Semantics 97, Electronic Notes in Theoretical Computer Science 6 (1997).
- [14] Edmund Robinson, Variations on algebra: monadicity and generalisations of algebraic theories, *Math. Structures in Computer Science* (to appear).
- [15] Ross Street, The formal theory of monads, *J. Pure Appl. Algebra* 2 (1972) 149–168.
- [16] Ross Street, Enriched categories and cohomology, *Quaestione Math.* 6 (1983) 265–283.

*Department of Computer Science
University of Edinburgh
King's Buildings
Edinburgh, EH9 3JZ
Scotland
Email: ajp@dcs.ed.ac.uk*

This article may be accessed via WWW at <http://www.tac.mta.ca/tac/> or by anonymous ftp at <ftp://ftp.tac.mta.ca/pub/tac/html/volumes/6/n7/n7.{dvi,ps}>

THEORY AND APPLICATIONS OF CATEGORIES (ISSN 1201-561X) will disseminate articles that significantly advance the study of categorical algebra or methods, or that make significant new contributions to mathematical science using categorical methods. The scope of the journal includes: all areas of pure category theory, including higher dimensional categories; applications of category theory to algebra, geometry and topology and other areas of mathematics; applications of category theory to computer science, physics and other mathematical sciences; contributions to scientific knowledge that make use of categorical methods.

Articles appearing in the journal have been carefully and critically refereed under the responsibility of members of the Editorial Board. Only papers judged to be both significant and excellent are accepted for publication.

The method of distribution of the journal is via the Internet tools **WWW/ftp**. The journal is archived electronically and in printed paper format.

SUBSCRIPTION INFORMATION. Individual subscribers receive (by e-mail) abstracts of articles as they are published. Full text of published articles is available in .dvi and Postscript format. Details will be e-mailed to new subscribers and are available by **WWW/ftp**. To subscribe, send e-mail to **tac@mta.ca** including a full name and postal address. For institutional subscription, send enquiries to the Managing Editor, Robert Rosebrugh, **rrosebrugh@mta.ca**.

INFORMATION FOR AUTHORS. The typesetting language of the journal is **T_EX**, and **L^AT_EX** is the preferred flavour. **T_EX** source of articles for publication should be submitted by e-mail directly to an appropriate Editor. They are listed below. Please obtain detailed information on submission format and style files from the journal's WWW server at URL <http://www.tac.mta.ca/tac/> or by anonymous ftp from **ftp.tac.mta.ca** in the directory **pub/tac/info**. You may also write to **tac@mta.ca** to receive details by e-mail.

EDITORIAL BOARD.

John Baez, University of California, Riverside: **baez@math.ucr.edu**

Michael Barr, McGill University: **barr@barris.org**

Lawrence Breen, Université de Paris 13: **breen@math.univ-paris13.fr**

Ronald Brown, University of North Wales: **r.brown@bangor.ac.uk**

Jean-Luc Brylinski, Pennsylvania State University: **jlb@math.psu.edu**

Aurelio Carboni, Università dell'Insubria: **carboni@fis.unico.it**

P. T. Johnstone, University of Cambridge: **ptj@pmms.cam.ac.uk**

G. Max Kelly, University of Sydney: **maxk@maths.usyd.edu.au**

Anders Kock, University of Aarhus: **kock@imf.au.dk**

F. William Lawvere, State University of New York at Buffalo: **wlawvere@acsu.buffalo.edu**

Jean-Louis Loday, Université de Strasbourg: **loday@math.u-strasbg.fr**

Ieke Moerdijk, University of Utrecht: **moerdijk@math.ruu.nl**

Susan Niefield, Union College: **niefiels@union.edu**

Robert Paré, Dalhousie University: **pare@mscs.dal.ca**

Andrew Pitts, University of Cambridge: **ap@c1.cam.ac.uk**

Robert Rosebrugh, Mount Allison University: **rrosebrugh@mta.ca**

Jiri Rosicky, Masaryk University: **rosicky@math.muni.cz**

James Stasheff, University of North Carolina: **jds@charlie.math.unc.edu**

Ross Street, Macquarie University: **street@math.mq.edu.au**

Walter Tholen, York University: **tholen@mathstat.yorku.ca**

Myles Tierney, Rutgers University: **tierney@math.rutgers.edu**

Robert F. C. Walters, University of Sydney: **walters.b@maths.usyd.edu.au**

R. J. Wood, Dalhousie University: **rjwood@mscs.dal.ca**



Countable Lawvere Theories and Computational Effects

John Power^{1,2}

*Laboratory for the Foundations of Computer Science, University of Edinburgh, King's Buildings,
Edinburgh EH9 3JZ, SCOTLAND*

Abstract

Lawvere theories have been one of the two main category theoretic formulations of universal algebra, the other being monads. Monads have appeared extensively over the past fifteen years in the theoretical computer science literature, specifically in connection with computational effects, but Lawvere theories have not. So we define the notion of (countable) Lawvere theory and give a precise statement of its relationship with the notion of monad on the category *Set*. We illustrate with examples arising from the study of computational effects, explaining how the notion of Lawvere theory keeps one closer to computational practice. We then describe constructions that one can make with Lawvere theories, notably sum, tensor, and distributive tensor, reflecting the ways in which the various computational effects are usually combined, thus giving denotational semantics for the combinations.

Keywords: mathematical operational semantics, modularity, timed transition systems, comonads, distributive laws

1 Introduction

Historically, there have been two main category theoretic formulations of universal algebra. The earlier was by Bill Lawvere in his doctoral thesis in 1963 [16]. Nowadays, his central construct is usually called a Lawvere theory, more prosaically a single-sorted finite product theory [1,2]. The notion of Lawvere theory axiomatises the notion of the clone of an equational theory. So every equational theory generates a Lawvere theory, and every Lawvere theory is generated by an infinite class of equational theories, i.e., all those equational theories for which it forms the clone. The notion of equational theory can in turn be given a category-theoretic formulation in terms of the notion of a single-sorted finite product sketch, the notion of sketch having been introduced by Ehresmann [1,2,5].

¹ This work is supported by EPSRC grant GR/586372/01: A Theory of Effects for Programming Languages.

² Email: ajp@inf.ed.ac.uk

The second category-theoretic formulation of universal algebra, which was in terms of monads, arose less directly. There was an extant notion of monad or triple that existed in the study of algebraic topology for reasons distinct from universal algebra. But in the late 1960’s, Linton, apparently inspired by a surprising and profound characterisation of monads by Eilenberg and Moore in [6], proved that every equational theory gives rise, somewhat indirectly, to a monad on the category *Set* [1]. With some effort, one can see that the finitary monads on *Set* are precisely those monads that arise [1,14], where finitariness is a size condition. The notion of Lawvere theory can be shown directly to be equivalent to that of a finitary monad (see [27] for an enriched version), the central construction required to prove the equivalence being the Kleisli construction [1,17]. We give the details of a countable version of this in Section 2.

Moving forward to the late 1980’s, computer scientists, led by Eugenio Moggi, became enamoured of the notion of monad [19,20,21]. Moggi wanted to unify the study of what he called notions of computation, perhaps better called computational effects, which he presented as a list of imperative features that one might add to an otherwise purely functional programming language. These included features such as exceptions, side-effects, interactive input/output, nondeterminism and probabilistic nondeterminism. He also included partiality and continuations in his list, although they are of a somewhat different nature, for instance because partiality arises from recursion without any imperative behaviour. Here, we shall not address the latter two constructs.

In retrospect, it seems obvious that universal algebra was fundamental to Moggi’s idea although that was not clear to him at the time. The various computational effects arise from computationally natural operations, such as *raise* for exceptions, *lookup* and *update* for side-effects, *read* and *write* for interactive input/output, nondeterministic \vee for nondeterminism, and $[0, 1]$ -many binary operations $+$, for probabilistic nondeterminism, subject to computationally natural equations; and thus they arise from computationally natural equational theories by Linton’s construction. We give the details of the examples in Section 3.

Having noticed that Moggi’s monads arise from such computationally natural equational theories, one wonders whether the notion of Lawvere theory might be of any use. That question led to a series of papers, primarily by Martin Hyland, Gordon Plotkin and myself [10,11,22,23,24]. Providing that one is willing to make the routine extension of the notion of Lawvere theory to allow for countable arities, they prove to be particularly helpful. They inherently give rise to the first unified account of the computational operations associated with each computational effect; they allow one to distinguish between different sorts of computational effects; they allow one naturally to abstract from the setting of functional languages to a less context-dependent analysis; and they allow an elegant and natural theory of combining effects, based on constructions that have long appeared in the literature, such as the sum and tensor product of theories [7,29]. The constructions that allow one to combine Lawvere theories are of independent mathematical interest, so we explain our leading examples of such constructions in Section 4. These operations

give mathematical substance to the notion of monad transformer [3,4], replacing the latter notion by more primitive constructs.

Once one has this theory of computational effects, which is independent of functional programming, one can elegantly reincorporate it back into functional programming [28], allowing one a more modular analysis than Moggi had. It works surprisingly simply: the Lawvere theory itself directly gives rise to a canonical model of the relevant effects together with the first-order fragment of Moggi's computational λ -calculus [19,20,21,28]. We do not give the details here as they would lead us away from our main task, which is to explain Lawvere theories.

There are two caveats to the above analysis. First, Lawvere theories as normally defined are inherently finitary. But because of recursion and hence the leading role of Nat in computer science, the main computing interest lies in a mild generalisation from finitariness to a countability condition. But that is routine, the various theorems generalising without fuss. So in our technical development, we shall simply describe the countable version. Second, again because of recursion, computer science ultimately requires base categories such as ωCpo rather than Set . That involves enrichment of the notion of Lawvere theory [27]. The enrichment is routine, but to understand the details requires some knowledge of enriched category theory [13], so for simplicity of exposition, we shall not give the enriched version here.

The paper is organised as follows. We recall the definition of a countable Lawvere theory and explain the relationship of the definition with the notion of monad on Set in Section 2. We illustrate with examples arising from computational effects in Section 3. And we discuss some of the constructions one can make naturally in terms of countable Lawvere theories in Section 4. There is no substantial new technical content in the paper: it is a distillation of work primarily in [10,11,24,26,27]. For some of the ideas in this paper addressed from a more computational perspective, see [25].

2 Countable Lawvere theories

In this section, we first give the definition of a countable Lawvere theory. We then show how every countable Lawvere theory yields a monad on Set . The monads that thus arise are exactly those monads on Set that are of countable rank, which is a size condition [13,14]. The correspondence extends to an equivalence of categories between Law_c , the category of countable Lawvere theories, and Mnd_c , the category of monads on Set with countable rank [27].

Definition 2.1 Let \aleph_1 denote a skeleton of the category of countable sets and all functions between them.

Spelling out the meaning of the definition, the category \aleph_1 has an object for each natural number n and an object for \aleph_0 . Up to equivalence, \aleph_1 is the free category with countable coproducts on 1: the coproducts are given by cardinal sum. In referring to \aleph_1 , we implicitly make a choice of the structure of its countable

coproducts. Since \aleph_1 has countable coproducts, it is immediate that the opposite category \aleph_1^{op} has countable products.

Definition 2.2 A *countable Lawvere theory* consists of a small category L with countable products and a strict countable-product preserving identity-on-objects functor $I : \aleph_1^{op} \rightarrow L$. A map of countable Lawvere theories from L to L' is a strict countable-product preserving functor from L to L' that commutes with I and I' .

So the objects of any countable Lawvere theory L are exactly the objects of \aleph_1 , and every function between such objects yields a map in L . One often refers to the maps of a countable Lawvere theory as *operations*. Trivially, the definitions of countable Lawvere theory and map between them yield a category Law_c , with composition given by ordinary composition of functors. Note that in the definition of countable Lawvere theory, I need not be an inclusion.

Example 2.3 There is a Lawvere theory $Triv$ that is equivalent to the unit category 1: its objects are the objects of \aleph_1 , and there is one arrow from any object to any other object. The functor I is the identity-on-objects but is trivial on maps.

That is a useful example for us in constructing counter-examples to natural conjectures. Although trivial, it is important to the structure of the category Law_c as it is the terminal object of Law_c , therefore corresponding to the terminal object of Mnd_c , which is the monad sending every set to 1. Despite this example, it is generally harmless to pretend that I is faithful, as it is in all examples of primary interest. For most mathematical purposes, one understands a countable Lawvere theory by study of its models.

Definition 2.4 A *model* of a countable Lawvere theory L in any category C with countable products is a countable-product preserving functor $M : L \rightarrow C$.

Definition 2.5 For any countable Lawvere theory L and any category C with countable products, the category $Mod(L, C)$ is defined to have objects given by all models of L in C , with maps given by all natural transformations between them.

The semantic category C of primary interest is Set . So consider a model M of a countable Lawvere theory L in Set . The set $M1$ determines Mn up to coherent isomorphism for every n in L : for M preserves countable products of L , equivalently of \aleph_1^{op} , these are countable coproducts of \aleph_1 , which are given by cardinal sum, and so Mn must be the product of n copies of $M1$. So, to give a model M is equivalent to giving a set $X = M1$ together with, for each map of the form $f : m \rightarrow 1$ in L , a function from X^m to X , subject to the equations given by the composition and product structure of L . This analysis routinely extends to any category C with countable products.

The definition of map in $Mod(L, C)$ is more subtle than it may first appear. One can readily prove that the naturality condition implies that all natural transformations between models respect countable product structure, i.e., for any natural transformation α between models M and N , and for any n in \aleph_1 , the map $\alpha_n : Mn \rightarrow Nn$ is given by the product of n copies of $\alpha_1 : M1 \rightarrow N1$. So

the maps in $Mod(L, C)$, which we defined to be all natural transformations, could equally be defined to be all natural transformations that respect the product structure of L .

The requirement that M preserves projections, which is part of what preservation of products means, determines the behaviour of M on If for every function f : for projections in L amount to coprojections in \aleph_1 , and every function f is given by a family of coprojections.

The above discussion leads to the notion of a *single-sorted countable-product sketch*, which is a category-theoretic formulation of the notion of equational theory. The usual way in which to obtain countable Lawvere theories is by means of sketches, with the Lawvere theory given freely on the sketch: Barr and Wells' book [2] treats sketches in loving detail, with a leading example given by the sketch for semigroups, i.e., they describe a single-sorted countable-product sketch for which the induced countable Lawvere theory L_{SG} is determined by the property that the category $Mod(L_{SG}, Set)$ is equivalent to the usual category of semigroups. To give a sketch amounts to giving operations and equations, the operations being allowed to be of countable arity, i.e., an equational theory with operations possibly of countable arity. We shall give our leading examples in the next section.

There is a canonical forgetful functor $U_L : Mod(L, C) \rightarrow C$ given by evaluation at the object 1 of L , equally of \aleph_1 . If that forgetful functor has a left adjoint F_L , as it does whenever C is locally countably presentable, it follows from Beck's monadicity theorem [1] that it exhibits $Mod(L, C)$ as equivalent to the category $T_L\text{-Alg}$ for the induced monad T_L on C . In particular, Set is locally countably presentable, so every countable Lawvere theory L induces a monad T_L on Set . With only a little more effort, one can prove the following.

Proposition 2.6 *The construction sending a countable Lawvere theory L to the monad T_L determined by the forgetful functor $U_L : Mod(L, Set) \rightarrow Set$ given by evaluation at 1 extends to a functor from Law_c to Mnd .*

One can readily check that for every countable Lawvere theory L , the monad T_L is of countable rank. The issue of rank is something of a distraction for the purposes of this paper: our aim here is to compare the notion of countable Lawvere theory with that of monad, without much concern for exactly what class of monads is obtained beyond the fact of their including all our leading examples. We need to mention rank only in order to make a precise statement of the converse to the construction we have just described, but development of it would detract from our goal. So we leave it with a reference [13], but without further analysis.

For a converse, first observe that for any monad T on Set , the Kleisli category $Kl(T)$ has all coproducts and the canonical functor $I : Set \rightarrow Kl(T)$ preserves them: for the canonical functor I has a right adjoint and is identity-on-objects. So, restricting I to \aleph_1 , which is a skeleton of the full subcategory of Set determined by countable sets, we have (the opposite of) a countable Lawvere theory. With a little more effort, we have the following.

Proposition 2.7 *The construction sending a monad T on Set to the category*

$Kl(T)_{\aleph_1}^{op}$ determined by restricting $Kl(T)$ to the objects of \aleph_1 extends to a functor from Mnd to Law_c .

It is routine to verify that, for any countable Lawvere theory L , the countable Lawvere theory L_T is isomorphic in Law to L . But the corresponding statement if one began with a monad T on Set is not true because, as mentioned above, the construction T_L yields only monads with countable rank. However, if T has countable rank, the functor from T -Alg to $Mod(L_T, Set)$ induced by the restriction is an equivalence of categories, and so T is isomorphic to T_{L_T} in Mnd . An enriched, thereby more general, version of the following result appears in [27].

Theorem 2.8 *The construction sending a countable Lawvere theory L to T_L together with that sending a monad T with countable rank to L_T induce an equivalence of categories between the category Law_c of countable Lawvere theories and the category Mnd_c of monads with countable rank on Set . Moreover, the comparison functor exhibits an equivalence between the categories $Mod(L, Set)$ and T_L -Alg.*

We have seen that to give a monad with countable rank on Set is equivalent to giving a countable Lawvere theory. In our motivating class of computational examples, which we explore in Section 3, the countable Lawvere theory changes the emphasis from the assignment to each set X of the set TX of values associated with a computational effect to the study of the operations associated with that computational effect; and that change proves to be fundamental to modelling the commutative combination of effects, to explaining their sum [10,11] and to discussing distributivity.

The work in this section enriches without fuss. The only point that does not enrich routinely is the informal discussion about equational theories: as best we know, there is currently no enriched notion of equational theory corresponding to enriched Lawvere theories. However, as we have outlined, that part of the discussion could be phrased in terms of sketches, for which an enriched account does exist or at least can readily be gleaned from the literature [15].

3 Examples Arising from Computational Effects

In this section, we consider examples of computational effects from the perspective of countable Lawvere theories. These computational effects have, in the past, been studied in terms of monads, as advocated fifteen years ago by Moggi [19,20,21], and they have recently been studied in terms of countable Lawvere theories by Hyland, Plotkin, and myself, notably in [10,11]. The latter approach deals more directly with the various effects in that it takes the operations generating the effects as primitive, whereas the monad approach does not give a unified account of the operations that induce each effect at all.

Example 3.1 The countable Lawvere theory L_E for exceptions is the free countable Lawvere theory generated by an operation $raise : 0 \longrightarrow E$, where E is a countable set of *exceptions*. In terms of operations and equations, this corresponds

to an E -indexed family of nullary operations with no equations. Note our use of the countable set E for the codomain of the operation of the Lawvere theory; strictly speaking we should instead have used the corresponding object of \aleph_1 , namely \aleph_0 ; it is, however, conceptually convenient to allow ourselves such minor liberties.

The monad generated by L_E is $T_E = - + E$. More generally, if C is any category with countable powers and countable coproducts, $Mod(L_E, C)$ is equivalent to the category of algebras for the monad $- + \underline{E}$, where \underline{E} for the E -fold copower of 1, i.e., $\coprod_E 1$.

In the case of side-effects, a sketch, and hence the countable Lawvere theory, is essentially given in [24] and is easy to describe.

Example 3.2 The countable Lawvere theory L_S for side-effects, where $S = Val^{Loc}$, is the free countable Lawvere theory generated by the operations $lookup : Val \rightarrow Loc$ and $update : 1 \rightarrow Loc \times Val$ subject to the seven natural equations listed in [24], four of them specifying interaction equations for $lookup$ and $update$ and three of them specifying commutation equations. Note, as in the case of exceptions, the use of codomains, here Loc and $Loc \times Val$, to handle indexing at the Lawvere theory level. It is shown in [24] that this Lawvere theory corresponds to the side-effects monad. More generally, if C is any category with countable powers and copowers then, slightly generalising the result in [24], $Mod(L_S, C)$ is equivalent to the category of algebras for the monad $(S \times -)^S$ where we write $(S \times -)$ for the S -fold copower $\coprod_S -$, and $(-)^S$ for the S -fold power $\prod_S -$.

For the next example, given any endofunctor F on a category C , let $\mu y.Fy$ denote the initial F -algebra if it exists. Then, for an endofunctor Σ on a category C with binary sums, the free Σ -algebra on an object x is $\mu y.(\Sigma y + x)$, with one existing if and only the other does. These free algebras exist if, for example, C is locally countably presentable and Σ has countable rank.

Example 3.3 The countable Lawvere theory $L_{I/O}$ for interactive input/output is the free countable Lawvere theory generated by operations $read : I \rightarrow 1$ and $write : 1 \rightarrow O$, where I is a countable set of *inputs* and O of *outputs*. The monad for interactive input/output $T_{I/O}(X) = \mu Y.(O \times Y + Y^I + X)$ corresponds to this Lawvere theory: $T_{I/O}(X)$ is the free Σ -algebra on X , where $\Sigma Y = O \times Y + Y^I$ is the *signature* functor determined by the two operations; an algebra for Σ consists of an O -indexed family of unary operations and an I -ary operation. This is also the form of $T_{I/O}$ in the more general situation where it corresponds to $Mod(L_S, C)$ for a locally countably presentable category C .

Example 3.4 The countable Lawvere theory L_N for (binary) nondeterminism is the countable Lawvere theory freely generated by a binary operation $\vee : 2 \rightarrow 1$ subject to equations for associativity, commutativity and idempotence, i.e., the countable Lawvere theory for a semilattice; the corresponding monad on Set is the finite non-empty subset monad \mathcal{F}^+ .

Example 3.5 The countable Lawvere theory L_P for probabilistic nondeterminism is that freely generated by $[0, 1]$ -many binary operations $+_r : 2 \rightarrow 1$ subject to the

equations for associativity, commutativity and idempotence in [8]. The corresponding monad on Set is the distributions with finite support monad, \mathcal{D}_f .

The category Set is not the category of primary interest in denotational semantics. One is more interested in ωCpo , and variants, in order to model recursion. The relationship between countable Lawvere theories and monads with countable rank generalises without fuss to one between countable *enriched* Lawvere theories and strong monads with countable rank on the category in which the enrichment takes place. For that theory to work, it suffices that category be locally countably presentable as a cartesian closed category. The category ωCpo is an example of such a category. So the work here generalises to include ωCpo [10,11,27].

4 Constructions on Countable Lawvere Theories

The category Law_c has excellent category theoretic structure. We investigate some of its structure in this section. In particular, we investigate three binary operations on the category Law_c that yield denotational semantics for the most common combinations of the computational effects discussed in Section 3. The three operations are given by sum, tensor, and distributivity. Each of these binary operations, modulo size, yields one of Moggi’s monad transformers: given any binary operation, if you fix one argument, you immediately have, modulo size, a monad transformer, i.e., a function from the set of monads to itself [3,4]. The binary operations may be applied many times, yielding combinations of multiple effects [10,11]. Most of the work in this section is an abbreviated version of work appearing in [11].

4.1 Sum

The category Law_c has coproducts or sums. In contrast, the category Mnd does not have sums. Sum is the most common way in which the computational effects of Section 3 are combined. The leading examples of the sum of computational effects are given by the combination of exceptions with all the other computational effects we consider: side-effects, interactive input/output and nondeterminism, and by the combination of interactive input/output with all other effects we consider except for side-effects.

The construction of the sum is complicated, especially when attempted in terms of monads. But all our examples of countable Lawvere theories are given freely on equational theories, and in those terms, the sum is easy to describe: one takes all operations of both equational theories, subject to all axioms of both. The complication arises in passing from the induced equational theory to the Lawvere theory freely generated by it, as, in doing so, one may apply the operations of one theory to the operations of the other, yielding a potentially transfinite induction in describing the set of all derived operations.

Care is required. For instance, given Lawvere theories L and L' , there are always maps of Lawvere theories given by coprojections $L \rightarrow L + L'$ and $L' \rightarrow L + L'$. But those coprojection functors need not be faithful. For instance, suppose L was

Triv . Then $L + L'$ is also Triv , so the coprojection from L' is trivial.

So we know the sum always exists and is a straightforward, familiar construction on equational theories. But for the purposes of calculation, it is sometimes convenient to have a more explicit construction of the sum qua monad, and that is investigated in detail in [10,11]. Two of the main consequences of the work therein are as follows.

Proposition 4.1 *Given a set E and any countable Lawvere theory L , the sum of the monads $(- + E)$ and T_L exists and is given by the monad $T_L(- + E)$.*

Modulo our usual caveat regarding size, this result explains how the exceptions monad transformer, sending a monad T_L to the composite $T_L(- + E)$, arises: one takes the disjoint union of the two sets of operations and retains the equations for T_L . And this explanation brings with it the theory of coproducts, such as their associativity and commutativity, and their interaction with other operations.

Proposition 4.2 *Let $T_{I/O}$ denote the monad for interactive input/output, i.e., the monad determined by the countable Lawvere theory of Example 3.3, and let L be any countable Lawvere theory. Then we have $(T_L + T_{I/O})x = T_L(\mu y.(O \times T_L y + (T_L y)^I + x))$, or equivalently, $\mu z.T_L(O \times z + z^I + x)$ [4].*

The central fact that allows us to make the above calculations is that, for exceptions and interactive input/output, the monads are generated by operations subject to no equations, and hence by a signature endofunctor [10,11].

4.2 Tensor

We now consider the tensor product $L \otimes L'$ of countable Lawvere theories L and L' [7,29]. The tensor product, which we are about to describe, yields a symmetric monoidal structure on the category Law_c with a universal property that exactly, modulo size, yields the side-effects monad transformer when one takes the tensor product of side-effects qua monad with any other countable Lawvere theory qua monad. This symmetric monoidal structure, with its defining universal property, seems unlikely to extend from Law_c , equivalently Mnd_c , to the whole of the category Mnd , but we do not have a counter-example to prove that.

The category \aleph_1 not only has countable coproducts, but also has finite products, which we denote by $a \times a'$. The object $a \times a'$ may also be seen as the coproduct of a copies of a' . So, given an arbitrary map $f' : a' \rightarrow b'$ in a countable Lawvere theory, it is immediately clear what we mean by the morphism $a \times f' : a \times a' \rightarrow a \times b'$. We define $f \times a'$ by conjugation, and, in the following, we suppress the canonical isomorphisms.

Definition 4.3 Given countable Lawvere theories L and L' , the countable Lawvere theory $L \otimes L'$, called the tensor product of L and L' , is defined by the universal property of having maps of countable Lawvere theories from L and L' to $L \otimes L'$, with commutativity of all operations of L with respect to all operations of L' , i.e., given $f : a \rightarrow b$ in L and $f' : a' \rightarrow b'$ in L' , we demand commutativity of the

diagram

$$\begin{array}{ccc}
 a \times a' & \xrightarrow{a \times f'} & a \times b' \\
 f \times a' \downarrow & & \downarrow f \times b' \\
 b \times a' & \xrightarrow{b \times f'} & b \times b'
 \end{array}$$

The tensor product always exists because it is defined by operations and equations, or equivalently by a sketch [1,2]. Its existence also follows, indeed more profoundly and elegantly, by appeal to the work on pseudo-commutativity in [12].

Proposition 4.4 *The tensor product \otimes extends canonically to a symmetric monoidal structure on the category of countable Lawvere theories.*

A proof for this proposition is elementary. The unit for the tensor product is the initial Lawvere theory, i.e, the theory generated by no operations and no equations. This is the initial object of the category of Lawvere theories, so is also the unit for the sum; and it corresponds to the identity monad.

This result gives some indication of the definitiveness of the tensor product, but not much. What is much less common, and is central to the proof of the main theorem about the combination of side-effects with other effects, and indeed is central to the understanding of what commutativity means, is a characterisation of $L \otimes L'$ in terms of the categories of models of L and L' [10,11].

Theorem 4.5 *For any category C with countable products, there is a coherent equivalence of categories between $Mod(L \otimes L', C)$ and $Mod(L, Mod(L', C))$.*

Corollary 4.6 *Let L_S denote the countable Lawvere theory for side-effects, where $S = Val^{Loc}$, and let L denote any countable Lawvere theory. Then the monad $T_{L_S \otimes L}$ is isomorphic to $(T_L(S \times -))^S$.*

Proof. For any countable Lawvere theory L , the category $Mod(L, Set)$ is complete and cocomplete, so has countable products and countable coproducts. And it is shown in [24] that if C has countable products and countable coproducts, $Mod(L_S, C)$ is equivalent to the category $T\text{-Alg}$ for the monad $T- = (S \times -)^S$ on C , using the notation of Example 3.2. By the discussion immediately before Proposition 2.6, the category $Mod(L, Set)$ is equivalent to $T_L\text{-Alg}$. We denote the canonical adjunction by $F_L \dashv U_L : Mod(L, Set) \rightarrow Set$. Right adjoints preserve products, left adjoints preserve coproducts. So the monad $T_{L_S \otimes L}$, which, by Theorem 4.5, is the monad determined by the composite forgetful functor from $T\text{-Alg}$ to Set , must be given by $T_{L_S \otimes L}X = U_L(S \times F_L X)^S = (U_L F_L(S \times X))^S = (T_L(S \times X))^S$ as required. \square

This result shows that, under the hypotheses of the theorem, our theory of the tensor product of computational effects agrees with Moggi's definition of the

side-effects monad transformer.

Corollary 4.7 *The side-effects theory for $S = \text{Val}^{\text{Loc}}$ is the Loc-fold tensor product of the side-effects theory for $S = \text{Val}$.*

Proof. By Corollary 4.6, the tensor product of two side effects theories, one for S and the other for S' , is the side effects theory for $S \times S'$. Now use induction and the finiteness of Loc . \square

For a final example of the tensor product, let M be a monoid and consider the combination of any countable Lawvere theory L , equivalently T_L , with the monad $M \times -$. There is a canonical distributive law of the monad $M \times -$ over T_L determined by the unique strength $t : M \times T_L - \longrightarrow T_L(M \times -)$ of $M \times -$ over T_L . So $T_L(M \times -)$ acquires a canonical monad structure. The following result appears in [11].

Theorem 4.8 *Let L be any countable Lawvere theory, let M be a monoid, and let L_M be the countable Lawvere theory corresponding to the monad $M \times -$. Then $T_{L_M \otimes L}$ is isomorphic to $T_L(M \times -)$.*

Proof. To give a model of L in $(M \times -)\text{-Alg}$ is equivalent to giving a model $m : L \longrightarrow \text{Set}$ of L in Set , together with an M -action $\alpha : M \times m(1) \longrightarrow m(1)$ on $m(1)$, such that the corresponding map $\bar{\alpha} : m(1) \longrightarrow m(1)^M$ is a map of models. This in turn is equivalent to giving a T_L -algebra (X, β) and an M -action $\alpha : M \times X \longrightarrow X$ on X such that $\bar{\alpha} : X \longrightarrow X^M$ is a map of T_L -algebras. But that in turn is equivalent to giving a $T_L(M \times -)$ -algebra by generalities about distributive laws of monads [1]. These equivalences are all functorial, yielding an isomorphism from $T_{L_M \otimes L}\text{-Alg}$ to $T_L(M \times -)\text{-Alg}$ and hence an isomorphism of monads between $T_{L_M \otimes L}$ and $T_L(M \times -)$. \square

Corollary 4.9 *The tensor product of $M \times -$ and $M' \times -$ is $(M \times M') \times -$.*

4.3 Distributivity

The third sort of combination of effects that appears in practice is given by distributivity, just like product distributes over sum in a ring. Distributivity of two nondeterministic operations over each other is central to Matthew Hennessy's modelling of concurrency in [9]. It also applies to the combination of nondeterminism with probabilistic nondeterminism [18]. We shall not spell out here how to describe a distributive combination of countable Lawvere theories, but the idea is clear, broadly similar to the construction of the tensor product.

One wants to take the sum of both theories, then factor by equations that assert distributivity of the operations of one theory over the operations of the other. One can again characterise the category of models of the distributive combination of theories, but the characterisation is quite subtle. If a countable Lawvere theory L is commutative, the category $\text{Mod}(L, \text{Set})$ is symmetric monoidal closed; and one can use $\text{Mod}(L, \text{Set})$ together with that symmetric monoidal closed structure as a target symmetric monoidal category in which to model the underlying *operad* $O(L')$ of another theory L' . One can thus speak of the category $\text{Mod}(O(L'), \text{Mod}(L, \text{Set}))$

of models of $O(L')$ in $Mod(L, Set)$. The category $Mod(O(L'), Mod(L, Set))$, subject to taking a canonical pullback that is yet to be fully investigated, is then equivalent to the category of models of the distributive combination in Set . One must do something a little more subtle again if L is not commutative.

Unlike the sum and tensor product, the distributive combination of countable Lawvere theories is not symmetric, but it still yields a monoidal structure on Law_c . It does enrich, but not as directly and easily as do the other constructions.

References

- [1] M. Barr and C. Wells. *Toposes, Triples, and Theories* Grundlehren der math. Wissenschaften 278, Springer-Verlag, 1985.
- [2] M. Barr and C. Wells. *Category Theory for Computing Science* Prentice-Hall, 1990.
- [3] N. Benton, J. Hughes, and E. Moggi. *Monads and Effects* APPSEM '00 Summer School, 2000.
- [4] P. Cenciarelli and E. Moggi. em A syntactic approach to modularity in denotational semantics, CWI Technical Report, 1983.
- [5] C. Ehresmann. Esquisses et types des structures algébriques, *Bul. Inst. Polit. Iasi*, Vol. 14, pp. 1–14, 1968.
- [6] S. Eilenberg and J. C. Moore. Adjoint Functors and Triples, *Illinois J. Math.*, Vol. 9, pp. 381–398, 1965.
- [7] P. J. Freyd. Algebra-valued functors in general and tensor products in particular *Colloq. Math. Wroclaw* Vol. 14, pp. 89–106, 1966.
- [8] R. Heckmann. Probabilistic Domains, in *Proc. CAAP '94*, LNCS, Vol. 136, pp. 21–56, Berlin: Springer-Verlag, 1994.
- [9] M. C. B. Hennessy. *Algebraic Theory of Processes* Cambridge, Massachusetts: MIT Press, 1988.
- [10] J. M. E. H. Hyland, G. D. Plotkin, and A. J. Power. Combining computational effects: commutativity and sum, in *Proc. 2nd IFIP Conf on Theoretical Computer Science* (eds. Ricardo A. Baeza-Yates, Ugo Montanari and Nicola Santoro), pp. 474–484, Kluwer, 2002.
- [11] J. M. E. H. Hyland, G. D. Plotkin, and A. J. Power. Combining effects: sum and tensor, *Theoretical Computer Science* (to appear).
- [12] J. M. E. Hyland and A. J. Power. Pseudo-closed 2-categories and pseudo-commutativities *J. Pure Appl. Algebra*, Vol. 175, pp. 141–185, 2002.
- [13] G. M. Kelly. *Basic Concepts of Enriched Category Theory*, Cambridge: Cambridge University Press, 1982.
- [14] G. M. Kelly and A. J. Power. Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads, *J. Pure Appl. Algebra*, Vol. 89, pp. 163–179, 1993.
- [15] Y. Kinoshita, A. J. Power, and M. Takeyama, Sketches, *J. Pure Appl. Algebra*, Vol. 143, pp. 275–291, 1999.
- [16] F. W. Lawvere. *Functorial Semantics of Algebraic Theories*, PhD Thesis, Columbia University, 1963.
- [17] S. Mac Lane. *Categories for the Working Mathematician*, Berlin: Springer-Verlag, 1971.
- [18] M. W. Mislove. Nondeterminism and Probabilistic Choice: Obeying the Laws, in *International Conference on Concurrency Theory*, pp. 350–364, URL:citeseer.nj.nec.com/mislove00nondeterminism.html, 2000.
- [19] E. Moggi. Computational lambda-calculus and monads, in *Proc. LICS '89*, pp. 14–23, Washington: IEEE Press, 1989.
- [20] E. Moggi. *An abstract view of programming languages*, University of Edinburgh, Report ECS-LFCS-90-113, 1989.
- [21] E. Moggi. Notions of computation and monads, *Inf. and Comp.*, Vol. 93, No. 1, pp. 55–92, 1991.

- [22] G. D. Plotkin and A. J. Power. Adequacy for Algebraic Effects, in *Proc. FOSSACS 2001* (eds. F. Honsell and M. Miculan), LNCS, Vol. 2030, pp. 1–24, Berlin: Springer-Verlag, 2001.
- [23] G. D. Plotkin and A. J. Power. Semantics for Algebraic Operations (extended abstract), in *Proc. MFPS XVII* (eds. S. Brookes and M. Mislove), ENTCS, Vol. 45, Amsterdam: Elsevier, 2001.
- [24] G. D. Plotkin and A. J. Power. Notions of computation determine monads, in *Proc. FOSSACS 2002* (eds. M. Nielsen and U. Engberg), LNCS, Vol. 73, Amsterdam: Elsevier, 2002.
- [25] G. D. Plotkin and A. J. Power. Computational effects and operations: an overview, in *Proc. Workshop Domains 2002*, ENTCS, Vol. 2303, pp. 342–356, Berlin: Springer-Verlag, 2002.
- [26] G. D. Plotkin and A. J. Power. Algebraic operations and effects, in *Proc. MFCSIT 2000, Applied Categorical Structures*, Vol. 11, pp. 69–94, 2003.
- [27] A. J. Power. Enriched Lawvere Theories, in *Theory and Applications of Categories*, pp. 83–93, 2000.
- [28] A. J. Power. Canonical Models for Computational Effects, in *Proc. FOSSACS 2004* (ed. I. Walukiewicz), LNCS Vol. 2987, pp. 438–452, Berlin: Springer-Verlag, 2004.
- [29] H. Schubert, *Categories*, Springer-Verlag, 1972.

Freyd categories are Enriched Lawvere Theories

Staton, S.

2014, Article / Letter to editor (Electronic Notes in Theoretical Computer Science, 303, 0, (2014), pp. 197-206)

Doi link to publisher: <https://doi.org/10.1016/j.entcs.2014.02.010>

Version of the following full text: Author's version preprint

Downloaded from: <https://hdl.handle.net/2066/127954>

Download date: 2025-05-02

Note:

To cite this publication please use the final published version (if applicable).

Freyd categories are enriched Lawvere theories

Sam Staton

Radboud University Nijmegen

Abstract

Lawvere theories provide a categorical formulation of the algebraic theories from universal algebra. Freyd categories are categorical models of first-order effectful programming languages.

The notion of *sound limit doctrine* has been used to classify accessible categories. We provide a definition of Lawvere theory that is enriched in a closed category that is locally presentable with respect to a sound limit doctrine.

For the doctrine of finite limits, we recover Power's enriched Lawvere theories. For the empty limit doctrine, our Lawvere theories are Freyd categories, and for the doctrine of finite products, our Lawvere theories are distributive Freyd categories. In this sense, computational effects are algebraic.

Keywords: Freyd categories, Lawvere theories, monads and notions of computation.

1 Introduction

Strong monads have helped to organize the semantics of impure programming languages from at least two perspectives: firstly by examining the crucial properties of concrete models of programming languages; secondly by axiomatizing the equations between programs that must hold in all models [24,25].

However, more refined perspectives have since emerged.

- Firstly, the monads involved in many concrete models of impure programming languages actually arise as free algebras for equational theories, in the setting of enriched category theory (e.g. [28,29]).
- Secondly, when we separate first-order effectful computation from higher-order types, we arrive at the notion of Freyd categories as an axiomatization of first-order effectful computation. (Moggi's monad-models can be recovered as closed Freyd categories, see e.g. [19].)

In this paper I explain that the second development can be seen as an instance of the former.

*This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs*

Informal overview

The generalization from traditional equational theories to enriched ones proceeds as follows. Recall that in a traditional algebraic signature there is a set of n -ary operations for each natural number n , and so a structure for the signature comprises a function $X^n \rightarrow X$ for each n -ary operation.

This is enriched by replacing the category of sets by a category \mathcal{V} (perhaps the simplest interesting example of \mathcal{V} to have in mind is the category of posets and monotone functions). The n -ary operations are no longer required to form a set, but rather an object O_n of \mathcal{V} ; the arities n are no longer natural numbers, but rather ‘finitary’ objects of \mathcal{V} ; and a structure for such a signature in a given \mathcal{V} -enriched category \mathcal{A} comprises a morphism $O_n \rightarrow \mathcal{A}(X^n, X)$ in \mathcal{V} , where X^n is a power. (This line of thought goes back to Kelly’s work [14, 15]; our starting point is Power’s development [30]; see also [34] for an overview.)

A traditional equational theory determines a Lawvere theory, which is a category where the objects are natural numbers, and a morphism $m \rightarrow 1$ is a term in m variables modulo the equations, and in general a morphism $m \rightarrow n$ is a family of n terms-mod-equations in m variables. The categories arising in this way can be characterized as categories \mathbb{L} with finite products equipped with a functor $J : \mathbb{N}^{\text{op}} \rightarrow \mathbb{L}$, where \mathbb{N} is the category of natural numbers and functions between them; the functor J is required to be identity-on-objects (i.e. \mathbb{N}^{op} and \mathbb{L} have the same objects) and to preserve products.

Similarly a \mathcal{V} -enriched Lawvere theory [30] is defined to be a \mathcal{V} -enriched category \mathbb{L} with ‘finitary’ powers and an identity-on-objects finitary-power-preserving \mathcal{V} -functor $\mathbb{F}^{\text{op}} \rightarrow \mathbb{L}$, where \mathbb{F} is the category of finitary objects of \mathcal{V} .

On the other hand, the notion of Freyd category arose in the work of Levy, Power and Thielecke [19, 33] as a categorical framework for first order effectful programs. Recall the basic ideas of the categorical interpretation of type theory: that types are denoted by objects of a category, that a context is denoted by the product of its constituent types, and that a judgement $\Gamma \vdash t : \tau$ is interpreted as a morphism $\Gamma \rightarrow \tau$ in the category. A Freyd category comprises two categories with the same objects: one \mathbb{V} , whose morphisms denote pure, value judgements, and one \mathbb{C} , whose morphisms denote judgements of computations, together with an identity-on-objects functor $J : \mathbb{V} \rightarrow \mathbb{C}$. For example, \mathbb{C} might be the Kleisli category for a strong monad on \mathbb{V} . Since the order of effectful computation matters, \mathbb{C} typically does not have products, but it does have a product-like structure, and the functor J is required to preserve it. This was initially described in terms of premonoidal categories [32]. Subsequently, Levy used a formulation based on actions of monoidal categories [18, App. B] (see also [23]) and that is what we use in this paper.

Coming back to the definition of enriched Lawvere theory, notice that, naively put, there is some choice in what is meant by ‘finitary’ when it comes to the arities. When $\mathcal{V} = \mathbf{Set}$, ‘finitary’ means finite. Power takes \mathcal{V} to be a locally finitely presentable category, and ‘finitary’ means finitely presentable. If \mathbb{V} is a category with finite products, and \mathcal{V} is the functor category $[\mathbb{V}^{\text{op}}, \mathbf{Set}]$, then we can take ‘finitary’ to mean representable. In this case, an enriched Lawvere theory is the same thing as a Freyd category (Theorem 3.3).

Several authors have found profit in analyzing the ‘arities’ of monads and Law-

vere theories, including in the study of computational effects [4,7,16,20,21]. The line of work best suited for us is the classification of accessible categories by Adámek et al. [1]. This is based on a notion of sound limit doctrine \mathfrak{D} , and includes concepts of \mathfrak{D} -presentable object, which form our arities, and locally \mathfrak{D} -presentable categories. A locally \mathfrak{D} -presentable category is tightly connected with its subcategory of \mathfrak{D} -presentable objects, since each is determined by the other. We consider enriched Lawvere theories in this setting, following Lack and Rosický [16]

By considering different sound limit doctrines we recover familiar concepts:

- For the sound doctrine of finite limits, enriched Lawvere theories are the concept defined by Power [30].
- For the sound doctrine of finite products, enriched Lawvere theories are the same as distributive Freyd categories (Theorem 3.5).
- For the empty sound doctrine, enriched Lawvere theories are the same as Freyd categories (Theorem 3.3).

2 Preliminaries

2.1 Sound limit doctrines

A sound limit doctrine is a class of limits that admits a well-behaved refinement of the theory of accessible and locally presentable categories [1].

Definition 2.1 [[1]] A *doctrine* is a set \mathfrak{D} of small categories. A \mathfrak{D} -limit is a limiting cone whose diagram is indexed by a category in \mathfrak{D} . Dually a \mathfrak{D} -colimit is a colimiting cone whose diagram is indexed by a category in \mathfrak{D} . We write \mathfrak{D}^{op} for the doctrine $\{\mathbb{D}^{\text{op}} \mid \mathbb{D} \in \mathfrak{D}\}$.

A set of small categories \mathfrak{D} is a *sound limit doctrine* if for any functor $F : \mathcal{A} \rightarrow \mathbf{Set}$ the left Kan extension $[\mathcal{A}^{\text{op}}, \mathbf{Set}] \rightarrow \mathbf{Set}$ of F along the Yoneda embedding $\mathcal{A} \rightarrow [\mathcal{A}^{\text{op}}, \mathbf{Set}]$ preserves \mathfrak{D} -limits if and only if it preserves \mathfrak{D} -limits of representables.

The condition of soundness ensures that the theory of accessible and locally presentable categories, which is traditionally based on λ -small limits, makes sense for \mathfrak{D} -limits. Examples of sound limit doctrines include:

- \mathfrak{FinLim} : the doctrine of finite limits;
- $\mathfrak{FinProd}$: the doctrine of finite products;
- \emptyset : the empty doctrine.

Those are the three doctrines that we study in this paper.

Definition 2.2 [[1]] Let \mathfrak{D} be a sound limit doctrine. A small category \mathbb{C} is \mathfrak{D} -filtered if \mathbb{C} -indexed colimits commute in \mathbf{Set} with \mathfrak{D} -limits (i.e. the functor $\text{colim} : [\mathbb{C}, \mathbf{Set}] \rightarrow \mathbf{Set}$ preserves \mathfrak{D} -limits). A \mathfrak{D} -filtered colimit is a colimiting cone whose diagram is indexed by a \mathfrak{D} -filtered category.

For example:

- A \mathfrak{FinLim} -filtered category is normally just called a filtered category.

- A $\mathfrak{FinProd}$ -filtered category is sometimes called a sifted category [3]. Roughly speaking, sifted colimits are built from filtered colimits and reflexive coequalizers.
- All categories are trivially \emptyset -filtered, so a \mathfrak{D} -filtered colimit is the same thing as a colimit.

2.2 Locally presentable categories

Definition 2.3 Let \mathfrak{D} be a sound limit doctrine. Let \mathcal{A} be a category with all small colimits. An object a of \mathcal{A} is \mathfrak{D} -presentable if the representable functor $\mathcal{A}(a, -) : \mathcal{A} \rightarrow \mathbf{Set}$ preserves \mathfrak{D} -filtered colimits. The cocomplete category \mathcal{A} is locally \mathfrak{D} -presentable if there is a set \mathbb{F} of \mathfrak{D} -presentable objects such that every object of \mathcal{A} is a \mathfrak{D} -filtered colimit of objects from \mathbb{F} .

For example:

- A locally \emptyset -presentable category is a presheaf category $[\mathbb{F}^{\text{op}}, \mathbf{Set}]$ and the \emptyset -presentable objects are retracts of representables.
- Any locally \emptyset -presentable category is also locally $\mathfrak{FinProd}$ -presentable. More generally, the category of models for a multi-sorted algebraic theory is always a locally $\mathfrak{FinProd}$ -presentable category, and all locally $\mathfrak{FinProd}$ -presentable categories arise in this way (e.g. [3]). In particular, the category of sets is locally $\mathfrak{FinProd}$ -presentable, and the $\mathfrak{FinProd}$ -presentable objects are the finite sets.
- A locally \mathfrak{FinLim} -presentable category is normally called a locally finitely presentable category (e.g. [2]). Any locally $\mathfrak{FinProd}$ -presentable category is also locally \mathfrak{FinLim} -presentable. More generally, the category of models for an ‘essentially algebraic’ theory is always a locally finitely presentable category, and all locally finitely presentable categories arise in this way.

2.3 Locally presentable symmetric monoidal closed categories

The following definition is a mild generalization of the standard concept of a locally finitely presentable closed category [15].

Definition 2.4 Let \mathfrak{D} be a sound limit doctrine. A symmetric monoidal closed category $(\mathcal{V}, \otimes, I)$ is locally \mathfrak{D} -presentable as a symmetric monoidal closed category if it is locally \mathfrak{D} -presentable, if I is \mathfrak{D} -presentable, and $a \otimes b$ is \mathfrak{D} -presentable when a and b are.

If $(\mathcal{V}, \otimes, I)$ is locally \mathfrak{D} -presentable as a closed category, then we define a basis for \mathcal{V} to be a small full subcategory \mathbb{F} of \mathcal{V} whose objects are \mathfrak{D} -presentable, which is closed under \mathfrak{D}^{op} -colimits and \otimes and I , and which is such that every object of \mathcal{V} is a \mathfrak{D} -filtered colimit of objects from \mathbb{F} .

Note that an object of a locally \mathfrak{D} -presentable closed category is \mathfrak{D} -presentable if and only if it is a retract of an object in the basis. If \mathfrak{D} contains the category with one object and one idempotent non-identity morphism, e.g. if $\mathfrak{D} = \mathfrak{FinLim}$, then the basis is closed under retracts and so all bases are equivalent.

Recall (e.g. [2, Prop. 1.45]) that for any category \mathbb{F} with \mathfrak{D}^{op} -colimits, the category $[\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$ of \mathfrak{D} -limit-preserving set-valued functors and natural transfor-

mations has a universal property as a cocompletion. It has small colimits; the Yoneda embedding restricts to a functor $\mathbb{F} \hookrightarrow [\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$ that preserves \mathfrak{D}^{op} -colimits; and every \mathfrak{D}^{op} -colimit-preserving functor $G : \mathbb{F} \rightarrow \mathcal{A}$ with \mathcal{A} cocomplete extends to a colimit-preserving-functor $G^* : [\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}} \rightarrow \mathcal{A}$, unique up-to unique isomorphism. The extension G^* has a right adjoint, $G_* : \mathcal{A} \rightarrow [\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$, with $G_*(x) = \mathcal{A}(G(-), x)$.

Moreover, if (\mathbb{F}, \otimes, i) is a symmetric monoidal category with \mathfrak{D}^{op} -colimits and $(a \otimes -) : \mathbb{F} \rightarrow \mathbb{F}$ preserves \mathfrak{D}^{op} -colimits for all a , then $[\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$ is a symmetric monoidal closed category and the embedding $\mathbb{F} \rightarrow [\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$ preserves the symmetric monoidal structure [8,11].

Proposition 2.5 *Let \mathfrak{D} be a sound limit doctrine.*

- *Let \mathbb{F} be a small symmetric monoidal category \mathbb{F} with \mathfrak{D}^{op} -colimits such that $a \otimes -$ preserves \mathfrak{D}^{op} -colimits for all a in \mathbb{F} . Then $[\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$ is locally \mathfrak{D} -presentable as a closed category, with basis \mathbb{F} .*
- *Let \mathcal{V} be locally \mathfrak{D} -presentable as a closed category, with basis \mathbb{F} . It is equivalent to $[\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$.*

2.4 Actions and powers

The relationship between monoidal actions and enrichment is widely understood (e.g. [13]) and has proved useful in studying algebraic theories and notions of computation (e.g. [9], [10, Ch. 6], [18], [23]). Proposition 2.7 is the main step towards our two main theorems.

Definition 2.6 Let (\mathbb{C}, \otimes, i) be a monoidal category and let \mathcal{A} be an ordinary category. An *action* of \mathbb{C} on \mathcal{A} is a functor $M : \mathbb{C} \times \mathcal{A} \rightarrow \mathcal{A}$ together with natural isomorphisms

$$M(i, x) \cong x \quad M(c \otimes d, x) \cong M(c, M(d, x))$$

satisfying the evident coherence conditions.

Note that any monoidal category acts on itself in the obvious way.

Recall [14] that, for a symmetric monoidal category $(\mathcal{V}, \otimes, I)$, a \mathcal{V} -enriched category $\underline{\mathcal{C}}$ is like an ordinary category except that between a pair of objects x, y in $\underline{\mathcal{C}}$, we have an object $\underline{\mathcal{C}}(x, y)$ of \mathcal{V} instead of a set of morphisms. Any enriched category $\underline{\mathcal{C}}$ has an underlying ordinary category \mathcal{C} with the same objects and with hom-sets $\mathcal{C}(x, y) = \mathcal{V}(I, \underline{\mathcal{C}}(x, y))$. An *enrichment* of an ordinary category \mathcal{C} in \mathcal{V} is defined to be a \mathcal{V} -enriched category $\underline{\mathcal{C}}$ whose underlying ordinary category is \mathcal{C} . For example, if \mathcal{V} is a symmetric monoidal closed category, then the closed structure provides an enrichment of \mathcal{V} in itself.

Finally, recall the definition of powers (aka cotensors) in a category $\underline{\mathcal{C}}$ enriched in a symmetric monoidal closed category $(\mathcal{V}, \otimes, I)$. For $x \in \underline{\mathcal{C}}$, $a \in \mathcal{V}$, a *power* is an object x^a together with an isomorphism $\mathcal{V}(a, \underline{\mathcal{C}}(y, x)) \cong \underline{\mathcal{C}}(y, x^a)$ that is natural in y .

Proposition 2.7 Let \mathfrak{D} be a sound limit doctrine. Let $(\mathcal{V}, \otimes, I)$ be locally \mathfrak{D} -presentable as a closed category with a basis \mathbb{F} . Let \mathcal{C} be an ordinary category. The following data are equivalent.

- (i) An action M of the monoidal category \mathbb{F}^{op} on \mathcal{C} such that for each x in \mathcal{C} the functor $M(-, x) : \mathbb{F}^{\text{op}} \rightarrow \mathcal{C}$ preserves \mathfrak{D} -limits.
- (ii) An enrichment of \mathcal{C} in \mathcal{V} with powers by objects in \mathbb{F} .

Proof notes. From (ii) to (i): let $M(a, x)$ be the power x^a . From (i) to (ii): we define the hom-object $\underline{\mathcal{C}}(x, y)$ in \mathcal{V} by working up to the equivalence $\mathcal{V} \simeq [\mathbb{F}^{\text{op}}, \mathbf{Set}]_{\mathfrak{D}}$: let $\underline{\mathcal{C}}(x, y)(a) = \mathcal{C}(x, y^a)$. \square

Proposition 2.7 is probably known quite widely. An instance ($\mathfrak{D} = \emptyset$) of Proposition 2.7 is implicit in Levy's work on call-by-push-value [18] and more recently explicit in Melliès work ([22, Prop. 11], [21, Lecture 6]).

3 Enriched Lawvere theories and Freyd categories

We now consider a definition of Lawvere theory enriched in a locally \mathfrak{D} -presentable closed category. We recall the definitions of Freyd category and distributive Freyd category, and show that they are instances of the concept of Lawvere theory.

3.1 Enriched Lawvere theories

Definition 3.1 Let \mathfrak{D} be a sound limit doctrine, and let $(\mathcal{V}, \otimes, I)$ be locally \mathfrak{D} -presentable as a closed category, with a basis \mathbb{F} . A \mathcal{V} -Lawvere theory is given by

- a category \mathbb{L} enriched in \mathcal{V} with powers by objects of \mathbb{F} .
- an identity-on-objects \mathcal{V} -functor $\mathbb{F}^{\text{op}} \rightarrow \mathbb{L}$ that preserves powers by objects of \mathbb{F} .

The choice of basis \mathbb{F} is irrelevant to the following extent. Define a *change of basis* $(\rho, r, s) : \mathbb{F} \rightarrow \mathbb{F}'$ to be given by, for each a in \mathbb{F} a choice of a section/retraction pair, $(a \xrightarrow{s_a} \rho_a \xrightarrow{r_a} a) = \text{id}$, with ρ_a in \mathbb{F}' . This choice determines an assignment from Lawvere theories \mathbb{L}' wrt \mathbb{F}' to Lawvere theories \mathbb{L} wrt \mathbb{F} : let $\mathbb{L}(a, b)$ be the equalizer

$$\mathbb{L}(a, b) \longrightarrow \mathbb{L}'(\rho_a, \rho_b) \begin{array}{c} \xrightarrow{s_b \cdot r_b \cdot -} \\[-1ex] \xrightarrow{- \cdot s_a \cdot r_a} \end{array} \mathbb{L}'(\rho_a, \rho_b)$$

(We could simplify this situation by requiring bases to be closed under retracts, but this would complicate our main theorems, 3.3 and 3.5.)

When $\mathfrak{D} = \mathfrak{FinLim}$, Definition 3.1 is the definition of Power [30]. When, moreover, \mathcal{V} is the category of sets with cartesian product structure, this is the original definition of Lawvere [17]. For a broader study of notions of Lawvere theory, including this one, see the article by Lack and Rosický [16]. It follows from the results in [16, §7] that, for a locally \mathfrak{D} -presentable closed category \mathcal{V} , to give a Lawvere \mathcal{V} -theory is to give an enriched monad on \mathcal{V} that preserves \mathfrak{D} -filtered colimits.

3.2 Freyd categories

We recall a formulation of Freyd categories proposed by Levy [18, App. B].

Definition 3.2 A *Freyd category* is given by

- a small category \mathbb{V} with finite products;
- a small category \mathbb{C} ;
- an action of \mathbb{V} on \mathbb{C} (with the finite products providing a symmetric monoidal structure for \mathbb{V});
- an identity on objects functor $\mathbb{V} \rightarrow \mathbb{C}$ that preserves the actions.

Theorem 3.3 *The following data are equivalent.*

- *A Freyd category.*
- *A Lawvere theory enriched in a locally \emptyset -presentable cartesian closed category.*

Proof notes. A Freyd category, i.e. an identity-on-objects action-preserving-functor $\mathbb{V} \rightarrow \mathbb{C}$, can equivalently be described as an identity-on-objects action-preserving-functor $\mathbb{V}^{\text{op}} \rightarrow \mathbb{C}^{\text{op}}$, which (by Prop. 2.7) is the same thing as a $[\mathbb{V}^{\text{op}}, \mathbf{Set}]$ -enriched power-preserving functor $\mathbb{V}^{\text{op}} \rightarrow \mathbb{C}^{\text{op}}$, which is the same thing as a Lawvere theory enriched in a locally \emptyset -presentable cartesian closed category.

3.3 Distributive Freyd categories

Recall that a distributive category is a category with finite sums and products such that for all objects a the functor $a \times (-)$ preserves sums. This is a model for simple first order type theory with sums and products. A distributive Freyd category [19,31,23], then, is a model for an effectful first order language with sums and products.

Definition 3.4 A *distributive Freyd category* is given by

- a distributive category \mathbb{V} ;
- a category \mathbb{C} with finite coproducts;
- an action of \mathbb{V} on \mathbb{C} that distributes over coproducts (i.e. $M(a, -)$ preserves coproducts for all a in \mathbb{V});
- an identity on objects functor $\mathbb{V} \rightarrow \mathbb{C}$ that preserves the action and coproducts.

Theorem 3.5 *The following data are equivalent.*

- *A distributive Freyd category.*
- *A Lawvere theory enriched in a locally $\mathfrak{FinProd}$ -presentable cartesian closed category.*

Remark. In this paper we focused on three sound limit doctrines: finite limits, finite products, and the empty doctrine. I am only aware of three other kinds of sound limit doctrine: terminal objects (whose enriched Lawvere theories are like distributive Freyd categories but with an initial object instead of all finite coproducts), finite connected limits, and λ -small limits for a regular cardinal λ .

Other work in this direction.

Power [31] already used ordinary Lawvere theories to build examples of distributive Freyd categories. He moreover showed how to build an enriched monad on $[\mathbb{V}^{\text{op}}, \mathbf{Set}]$ from a Freyd category $\mathbb{V} \rightarrow \mathbb{C}$, and how to build an enriched monad on $[\mathbb{V}^{\text{op}}, \mathbf{Set}]_{\text{fin}\mathfrak{pro}\mathfrak{d}}$ from a distributive Freyd category $\mathbb{V} \rightarrow \mathbb{C}$. (There, enriched monads are explained in terms of closed Freyd categories.)

Several other authors have discussed the relationships between Freyd categories, monads and the Yoneda embedding [4,5,6,12].

My own main starting point was my work with Møgelberg [23]. We considered ‘effect theories’, which are a programming language syntax for those distributive Freyd categories where \mathbb{V} is a free finite coproduct completion of a category with finite products. In that work we used effect theories in the same way that one uses classical algebraic theories, by considering their models and comodels. Subsequently I developed ‘parameterized algebraic theories’ [35,36], which are an alternative syntax and deduction system for the same structures (with syntax inspired by [26,27]). One could say that the programming language syntax is for distributive Freyd categories $\mathbb{V} \rightarrow \mathbb{C}$ whereas the algebraic syntax is for the corresponding enriched Lawvere theories $\mathbb{V}^{\text{op}} \rightarrow \mathbb{C}^{\text{op}}$.

The purpose of this paper was to emphasise the relationship between effectful computation and universal algebra.

Acknowledgements.

It has been helpful about this topic with many people over the years, including Danel Ahman, James Chapman, Jeff Egger, Marcelo Fiore, Chung-Kil Hur, Martin Hyland, Alexander Kurz, Paul Levy, Paul-André Melliès, Rasmus Møgelberg, John Power, Mathys Rennela, Alex Simpson, Stephan Spahn, and Tarmo Uustalu.

This research has been supported by ERC Grant ‘Quantum Computation, Logic, and Security’.

References

- [1] Adámek, J., F. Borceux, S. Lack and J. Rosický, *A classification of accessible categories*, J. Pure Appl. Algebra **175** (2002), pp. 7–30.
- [2] Adámek, J. and J. Rosický, “Locally presentable and accessible categories,” Cambridge University Press, 1994.
- [3] Adámek, J. and J. Rosický, *On sifted colimits and generalized varieties*, Theory Appl. Categ. **8** (2001), pp. 33–53.
- [4] Altenkirch, T., J. Chapman and T. Uustalu, *Monads need not be endofunctors*, in: *Proc. FOSSACS 2010*, Lecture Notes in Computer Science **6014** (2010), pp. 297–311.
- [5] Asada, K., *Arrows are strong monads*, in: *Proc. MSFP 2010*, 2010, pp. 33–42.
- [6] Atkey, R., *What is a categorical model of arrows?*, in: *Proc. MSFP 2008*, Electr. Notes Theor. Comput. Sci. **229**, 2011, pp. 19–37.
- [7] Berger, C., P.-A. Melliès and M. Weber, *Monads with arities and their associated theories*, J. Pure Appl. Algebra **216** (2012), pp. 2029–2048.
- [8] Day, B., *On closed categories of functors*, in: *Reports of the Midwest Category Seminar, IV*, Lect. Notes Math. **137** (1970), pp. 1–38.

- [9] Fiore, M. P., *Second-order and dependently-sorted abstract syntax*, in: *Proc. LICS 2008*, 2008, pp. 57–68.
- [10] Hur, C.-K., “Categorical Equational Systems: Algebraic Models and Equational Reasoning,” Ph.D. thesis, University of Cambridge (2009).
- [11] Im, G. and G. Kelly, *A universal property of the convolution monoidal structure*, *J. Pure Appl. Algebra* **43** (1986), pp. 75–88.
- [12] Jacobs, B., C. Heunen and I. Hasuo, *Categorical semantics for arrows*, *J. Funct. Program.* **19** (2009), pp. 403–438.
- [13] Janelidze, G. and G. Kelly, *A note on actions of a monoidal category*, *Theory Appl. Categ.* **9** (2001), pp. 61–91.
- [14] Kelly, G. M., “Basic concepts of enriched category theory,” Cambridge University Press, 1982.
- [15] Kelly, G. M., *Structures defined by finite limits in the enriched context, I*, *Cahiers de topologie et géométrie différentielle catégoriques* **23** (1982), pp. 3–42.
- [16] Lack, S. and J. Rosický, *Notions of Lawvere theory*, *Applied Categorical Structures* **19** (2011), pp. 363–391.
- [17] Lawvere, F. W., “Functorial Semantics of Algebraic Theories,” Ph.D. thesis, Columbia University (1963), Reprints in TAC no. 5.
- [18] Levy, P. B., “Call-by-Push-Value. A Functional/Imperative Synthesis,” *Semantic Structures in Computation* **2**, Springer, 2004.
- [19] Levy, P. B., J. Power and H. Thielecke, *Modelling environments in call-by-value programming languages*, *Information and Computation* **185** (2003), pp. 182–210.
- [20] Mellies, P.-A., *Segal condition meets computational effects*, in: *Proc. LICS 2010*, 2010, pp. 150–159.
- [21] Mellies, P.-A., *Local stores in string diagrams* (2011), slides from a lecture course, available online at <http://tinyurl.com/mellies-itu-2011>.
- [22] Mellies, P.-A., *Parametric monads and enriched adjunctions* (2012), unpublished draft available at <http://www.pps.univ-paris-diderot.fr/~mellies/tensorial-logic.html>.
- [23] Møgelberg, R. E. and S. Staton, *Linearly-used state in models of call-by-value*, in: *Proc. CALCO 2011*, 2011, pp. 298–313.
- [24] Moggi, E., *Computational lambda-calculus and monads*, in: *Proc. LICS'89* (1989), pp. 14–23.
- [25] Moggi, E., *Notions of computation and monads*, *Information and Computation* **93** (1991), pp. 55–92.
- [26] Plotkin, G., *Some varieties of equational logic*, in: *Algebra, meaning and computation*, Springer, 2006 .
- [27] Plotkin, G. and M. Pretnar, *Handlers of algebraic effects*, in: *Proc. ESOP 2009* (2009), pp. 80–94.
- [28] Plotkin, G. D. and A. J. Power, *Computational effects and operations: An overview*, in: *Proc. Domains VI*, Electr. Notes Theor. Comput. Sci. **73** (2004), pp. 140–163.
- [29] Plotkin, G. D. and J. Power, *Notions of computation determine monads*, in: *Proc. FOSSACS 2002* (2002), pp. 342–356.
- [30] Power, J., *Enriched Lawvere theories*, *Theory Appl. Categ.* (1999), pp. 89–93.
- [31] Power, J., *Generic models for computational effects*, *Theor. Comput. Sci.* **364** (2006), pp. 254–269.
- [32] Power, J. and E. Robinson, *Premonoidal categories and notions of computation*, *Math. Struct. in Comput. Sci.* **7** (1997), pp. 453–468.
- [33] Power, J. and H. Thielecke, *Closed Freyd- and κ -categories*, in: *Proc. ICALP'99*, Lecture Notes in Comput. Sci. **1644**, 1999, pp. 625–634.
- [34] Robinson, E., *Variations on algebra: Monadicity and generalisations of equational theories*, *Formal Asp. Comput.* **13** (2002), pp. 308–326.
- [35] Staton, S., *An algebraic presentation of predicate logic*, in: *Proc. FOSSACS 2013*, 2013, pp. 401–417.
- [36] Staton, S., *Instances of computational effects: An algebraic perspective*, in: *Proc. LICS 2013*, 2013, pp. 519–519.

CHAPTER

E

Combinatory Grammar

Categorial

categorial grammar
combinatory categorial grammar

In this chapter, we provide an overview of **categorial grammar** ([Ajdukiewicz 1935](#), [Bar-Hillel 1953](#)), an early lexicalized grammar model, as well as an important modern extension, **combinatory categorial grammar**, or CCG ([Steedman 1996](#), [Steedman 1989](#), [Steedman 2000](#)). CCG is a heavily lexicalized approach motivated by both syntactic and semantic considerations. It is an exemplar of a set of computationally relevant approaches to grammar that emphasize putting grammatical information in a rich lexicon, including Lexical-Functional Grammar (LFG) ([Bresnan, 1982](#)), Head-Driven Phrase Structure Grammar (HPSG) ([Pollard and Sag, 1994](#)), and Tree-Adjoining Grammar (TAG) ([Joshi, 1985](#)).

The categorial approach consists of three major elements: a set of categories, a lexicon that associates words with categories, and a set of rules that govern how categories combine in context.

E.1 CCG Categories

Categories are either atomic elements or single-argument functions that return a category as a value when provided with a desired category as argument. More formally, we can define \mathcal{C} , a set of categories for a grammar as follows:

- $\mathcal{A} \subseteq \mathcal{C}$, where \mathcal{A} is a given set of atomic elements
- $(X/Y), (X\backslash Y) \in \mathcal{C}$, if $X, Y \in \mathcal{C}$

The slash notation shown here is used to define the functions in the grammar. It specifies the type of the expected argument, the direction it is expected be found, and the type of the result. Thus, (X/Y) is a function that seeks a constituent of type Y to its right and returns a value of X ; $(X\backslash Y)$ is the same except it seeks its argument to the left.

The set of atomic categories is typically very small and includes familiar elements such as sentences and noun phrases. Functional categories include verb phrases and complex noun phrases among others.

E.2 The Lexicon

The lexicon in a categorial approach consists of assignments of categories to words. These assignments can either be to atomic or functional categories, and due to lexical ambiguity words can be assigned to multiple categories. Consider the following

sample lexical entries.

flight : N
Miami : NP
cancel : $(S \setminus NP)/NP$

Nouns and proper nouns like *flight* and *Miami* are assigned to atomic categories, reflecting their typical role as arguments to functions. On the other hand, a transitive verb like *cancel* is assigned the category $(S \backslash NP)/NP$: a function that seeks an *NP* on its right and returns as its value a function with the type $(S \backslash NP)$. This function can, in turn, combine with an *NP* on the left, yielding an *S* as the result. This captures subcategorization information with a computationally useful, internal structure.

Ditransitive verbs like *give*, which expect two arguments after the verb, would have the category $((S \setminus NP)/NP)/NP$: a function that combines with an *NP* on its right to yield yet another function corresponding to the transitive verb $(S \setminus NP)/NP$ category such as the one given above for *cancel*.

E.3 Rules

The rules of a categorial grammar specify how functions and their arguments combine. The following two rule templates constitute the basis for all categorial grammars.

$$X/Y \ Y \Rightarrow X \quad (\text{E.1})$$

$$Y \ X \setminus Y \Rightarrow X \quad (\text{E.2})$$

The first rule applies a function to its argument on the right, while the second looks to the left for its argument. We'll refer to the first as **forward function application**, and the second as **backward function application**. The result of applying either of these rules is the category specified as the value of the function being applied.

Given these rules and a simple lexicon, let's consider an analysis of the sentence *United serves Miami*. Assume that *serves* is a transitive verb with the category $(S \setminus NP) / NP$ and that *United* and *Miami* are both simple *NPs*. Using both forward and backward function application, the derivation would proceed as follows:

$$\begin{array}{c}
 \frac{\text{United} \quad \text{serves} \quad \text{Miami}}{\text{NP} \qquad (\text{S} \backslash \text{NP}) / \text{NP} \qquad \text{NP}} \\
 \qquad\qquad\qquad \searrow \\
 \qquad\qquad\qquad \text{S} \backslash \text{NP} \\
 \hline
 \qquad\qquad\qquad \text{S}
 \end{array}$$

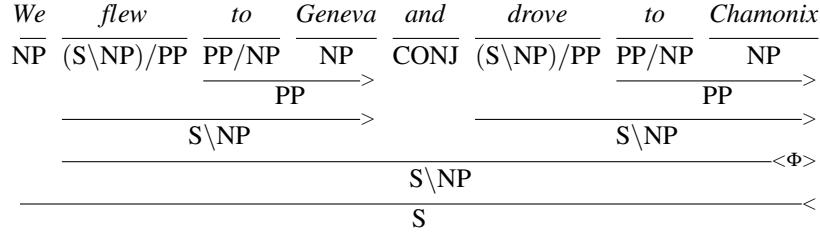
Categorial grammar derivations are illustrated growing down from the words, rule applications are illustrated with a horizontal line that spans the elements involved, with the type of the operation indicated at the right end of the line. In this example, there are two function applications: one forward function application indicated by the $>$ that applies the verb *serves* to the *NP* on its right, and one backward function application indicated by the $<$ that applies the result of the first to the *NP United* on its left.

English permits the coordination of two constituents of the same type, resulting in a new constituent of the same type. The following rule provides the mechanism

to handle such examples.

$$X \ CONJ \ X \Rightarrow X \quad (\text{E.3})$$

This rule states that when two constituents of the same category are separated by a constituent of type *CONJ* they can be combined into a single larger constituent of the same type. The following derivation illustrates the use of this rule.



Here the two $S \setminus NP$ constituents are combined via the conjunction operator $\langle \Phi \rangle$ to form a larger constituent of the same type, which can then be combined with the subject NP via backward function application.

These examples illustrate the lexical nature of the categorial grammar approach. The grammatical facts about a language are largely encoded in the lexicon, while the rules of the grammar are boiled down to a set of three rules. Unfortunately, the basic categorial approach does not give us any more expressive power than we had with traditional CFG rules; it just moves information from the grammar to the lexicon. To move beyond these limitations CCG includes operations that operate over functions.

The first pair of operators permit us to **compose** adjacent functions.

$$X/Y \ Y/Z \Rightarrow X/Z \quad (\text{E.4})$$

$$Y\backslash Z \ X\backslash Y \Rightarrow X\backslash Z \quad (\text{E.5})$$

forward composition

The first rule, called **forward composition**, can be applied to adjacent constituents where the first is a function seeking an argument of type Y to its right, and the second is a function that provides Y as a result. This rule allows us to compose these two functions into a single one with the type of the first constituent and the argument of the second. Although the notation is a little awkward, the second rule, **backward composition** is the same, except that we're looking to the left instead of to the right for the relevant arguments. Both kinds of composition are signalled by a **B** in CCG diagrams, accompanied by a $<$ or $>$ to indicate the direction.

backward composition

type raising

The next operator is **type raising**. Type raising elevates simple categories to the status of functions. More specifically, type raising takes a category and converts it to a function that seeks as an argument a function that takes the original category as its argument. The following schema show two versions of type raising: one for arguments to the right, and one for the left.

$$X \Rightarrow T/(T\backslash X) \quad (\text{E.6})$$

$$X \Rightarrow T\backslash(T/X) \quad (\text{E.7})$$

The category T in these rules can correspond to any of the atomic or functional categories already present in the grammar.

A particularly useful example of type raising transforms a simple NP argument in subject position to a function that can compose with a following VP . To see how

4 APPENDIX E • COMBINATORY CATEGORIAL GRAMMAR

this works, let's revisit our earlier example of *United serves Miami*. Instead of classifying *United* as an *NP* which can serve as an argument to the function attached to *serves*, we can use type raising to reinvent it as a function in its own right as follows.

$$NP \Rightarrow S/(S\backslash NP)$$

Combining this type-raised constituent with the forward composition rule (E.4) permits the following alternative to our previous derivation.

$$\begin{array}{c} \frac{\begin{array}{ccc} United & \text{serves} & Miami \\ \hline NP & (S\backslash NP)/NP & NP \end{array}}{S/(S\backslash NP)} \\ \xrightarrow{T} \\ \frac{S/(S\backslash NP)}{S/NP} \\ \xrightarrow{B} \\ \hline S \end{array}$$

By type raising *United* to $S/(S\backslash NP)$, we can compose it with the transitive verb *serves* to yield the (S/NP) function needed to complete the derivation.

There are several interesting things to note about this derivation. First, it provides a left-to-right, word-by-word derivation that more closely mirrors the way humans process language. This makes CCG a particularly apt framework for psycholinguistic studies. Second, this derivation involves the use of an intermediate unit of analysis, *United serves*, that does not correspond to a traditional constituent in English. This ability to make use of such non-constituent elements provides CCG with the ability to handle the coordination of phrases that are not proper constituents, as in the following example.

(E.8) We flew IcelandAir to Geneva and SwissAir to London.

Here, the segments that are being coordinated are *IcelandAir to Geneva* and *SwissAir to London*, phrases that would not normally be considered constituents, as can be seen in the following standard derivation for the verb phrase *flew IcelandAir to Geneva*.

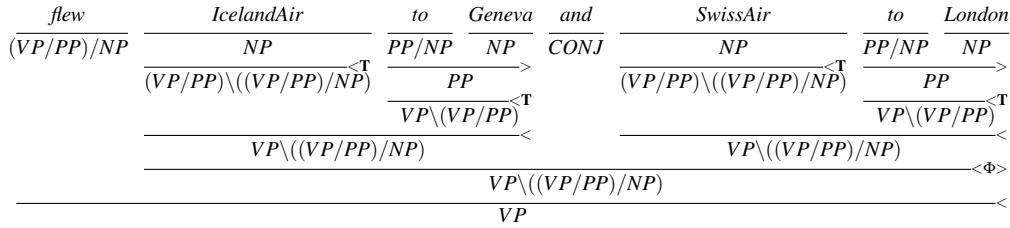
$$\begin{array}{c} \frac{\begin{array}{cccc} \text{flew} & \text{IcelandAir} & \text{to} & \text{Geneva} \\ \hline (VP/PP)/NP & NP & PP/NP & NP \end{array}}{VP/PP} \\ \xrightarrow{B} \\ \hline VP \end{array}$$

In this derivation, there is no single constituent that corresponds to *IcelandAir to Geneva*, and hence no opportunity to make use of the $\langle\Phi\rangle$ operator. Note that complex CCG categories can get a little cumbersome, so we'll use *VP* as a shorthand for $(S\backslash NP)$ in this and the following derivations.

The following alternative derivation provides the required element through the use of both backward type raising (E.7) and backward function composition (E.5).

$$\begin{array}{c} \frac{\begin{array}{ccccc} \text{flew} & \text{IcelandAir} & \text{to} & \text{Geneva} \\ \hline (VP/PP)/NP & NP & PP/NP & NP \end{array}}{(VP/PP)\backslash((VP/PP)/NP)} \\ \xleftarrow{T} \\ \frac{(VP/PP)\backslash((VP/PP)/NP)}{VP\backslash(VP/PP)} \\ \xleftarrow{B} \\ \hline VP \end{array}$$

Applying the same analysis to *SwissAir to London* satisfies the requirements for the $\langle\Phi\rangle$ operator, yielding the following derivation for our original example (E.8).

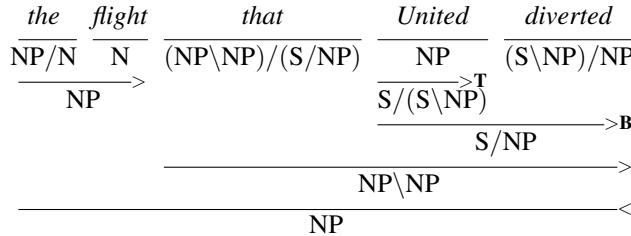


Finally, let's examine how these advanced operators can be used to handle **long-distance dependencies** (also referred to as syntactic movement or extraction). As mentioned in Appendix D, long-distance dependencies arise from many English constructions including wh-questions, relative clauses, and topicalization. What these constructions have in common is a constituent that appears somewhere distant from its usual, or expected, location. Consider the following relative clause as an example.

the flight that United diverted

Here, *divert* is a transitive verb that expects two *NP* arguments, a subject *NP* to its left and a direct object *NP* to its right; its category is therefore $(S\backslash NP)/NP$. However, in this example the direct object *the flight* has been “moved” to the beginning of the clause, while the subject *United* remains in its normal position. What is needed is a way to incorporate the subject argument, while dealing with the fact that *the flight* is not in its expected location.

The following derivation accomplishes this, again through the combined use of type raising and function composition.



As we saw with our earlier examples, the first step of this derivation is type raising *United* to the category $S/(S\backslash NP)$ allowing it to combine with *diverted* via forward composition. The result of this composition is S/NP which preserves the fact that we are still looking for an *NP* to fill the missing direct object. The second critical piece is the lexical category assigned to the word *that*: $(NP\backslash NP)/(S/NP)$. This function seeks a verb phrase missing an argument to its right, and transforms it into an *NP* seeking a missing element to its left, precisely where we find *the flight*.

E.4 CCGbank

As with phrase-structure approaches, treebanks play an important role in CCG-based approaches to parsing. CCGbank ([Hockenmaier and Steedman, 2007](#)) is the largest and most widely used CCG treebank. It was created by automatically translating phrase-structure trees from the Penn Treebank via a rule-based approach. The method produced successful translations of over 99% of the trees in the Penn Treebank resulting in 48,934 sentences paired with CCG derivations. It also provides a

6 APPENDIX E • COMBINATORY CATEGORIAL GRAMMAR

lexicon of 44,000 words with over 1200 categories. Appendix C will discuss how these resources can be used to train CCG parsers.

E.5 Ambiguity in CCG

As is always the case in parsing, managing ambiguity is the key to successful CCG parsing. The difficulties with CCG parsing arise from the ambiguity caused by the large number of complex lexical categories combined with the very general nature of the grammatical rules. To see some of the ways that ambiguity arises in a categorial framework, consider the following example.

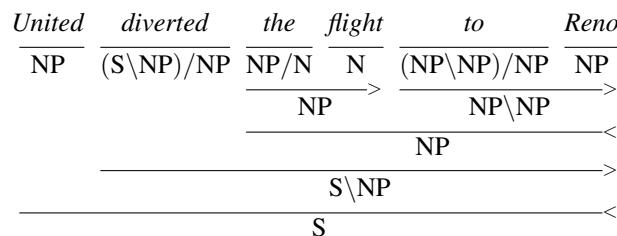
(E.9) United diverted the flight to Reno.

Our grasp of the role of *the flight* in this example depends on whether the prepositional phrase *to Reno* is taken as a modifier of *the flight*, as a modifier of the entire verb phrase, or as a potential second argument to the verb *divert*. In a context-free grammar approach, this ambiguity would manifest itself as a choice among the following rules in the grammar.

$$\begin{aligned} \text{Nominal} &\rightarrow \text{Nominal PP} \\ \text{VP} &\rightarrow \text{VP PP} \\ \text{VP} &\rightarrow \text{Verb NP PP} \end{aligned}$$

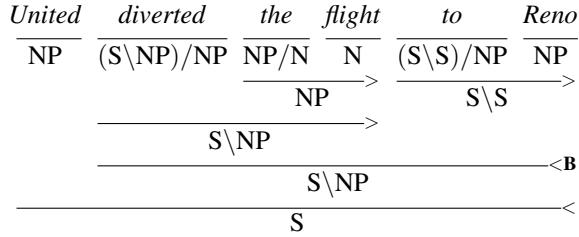
In a phrase-structure approach we would simply assign the word *to* to the category *P* allowing it to combine with *Reno* to form a prepositional phrase. The subsequent choice of grammar rules would then dictate the ultimate derivation. In the categorial approach, we can associate *to* with distinct categories to reflect the ways in which it might interact with other elements in a sentence. The fairly abstract combinatoric rules would then sort out which derivations are possible. Therefore, the source of ambiguity arises not from the grammar but rather from the lexicon.

Let's see how this works by considering several possible derivations for this example. To capture the case where the prepositional phrase *to Reno* modifies *the flight*, we assign the preposition *to* the category $(NP\backslash NP)/NP$, which gives rise to the following derivation.

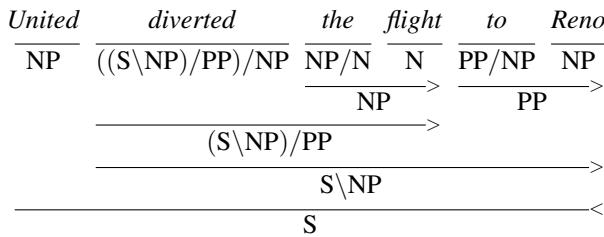


Here, the category assigned to *to* expects to find two arguments: one to the right as with a traditional preposition, and one to the left that corresponds to the *NP* to be modified.

Alternatively, we could assign *to* to the category $(S\backslash S)/NP$, which permits the following derivation where *to Reno* modifies the preceding verb phrase.



A third possibility is to view *divert* as a ditransitive verb by assigning it to the category $((\text{S}\backslash\text{NP})/\text{PP})/\text{NP}$, while treating *to Reno* as a simple prepositional phrase.



While CCG parsers are still subject to ambiguity arising from the choice of grammar rules, including the kind of spurious ambiguity discussed above, it should be clear that the choice of lexical categories is the primary problem to be addressed in CCG parsing.

E.6 CCG Parsing

Since the rules in combinatory grammars are either binary or unary, a bottom-up, tabular approach based on the CKY algorithm should be directly applicable to CCG parsing. Unfortunately, the large number of lexical categories available for each word, combined with the promiscuity of CCG’s combinatoric rules, leads to an explosion in the number of (mostly useless) constituents added to the parsing table. The key to managing this explosion of zombie constituents is to accurately assess and exploit the most likely lexical categories possible for each word—a process called supertagging.

These following sections describe an approach to CCG parsing that make use of supertags, structuring the parsing process as a heuristic search through the use of the A* algorithm.

E.6.1 Supertagging

Chapter 17 introduced the task of part-of-speech tagging, the process of assigning the correct lexical category to each word in a sentence. **Supertagging** is the corresponding task for highly lexicalized grammar frameworks, where the assigned tags often dictate much of the derivation for a sentence (Bangalore and Joshi, 1999).

CCG supertaggers rely on treebanks such as CCGbank to provide both the overall set of lexical categories as well as the allowable category assignments for each word in the lexicon. CCGbank includes over 1000 lexical categories, however, in practice, most supertaggers limit their tagsets to those tags that occur at least 10

8 APPENDIX E • COMBINATORY CATEGORIAL GRAMMAR

times in the training corpus. This results in a total of around 425 lexical categories available for use in the lexicon. Note that even this smaller number is large in contrast to the 45 POS types used by the Penn Treebank tagset.

As with traditional part-of-speech tagging, the standard approach to building a CCG supertagger is to use supervised machine learning to build a sequence labeler from hand-annotated training data. To find the most likely sequence of tags given a sentence, it is most common to use a neural sequence model, either RNN or Transformer.

It's also possible, however, to use the CRF tagging model described in Chapter 17, using similar features; the current word w_i , its surrounding words within l words, local POS tags and character suffixes, and the supertag from the prior timestep, training by maximizing log-likelihood of the training corpus and decoding via the Viterbi algorithm as described in Chapter 17.

Unfortunately the large number of possible supertags combined with high per-word ambiguity leads the naive CRF algorithm to error rates that are too high for practical use in a parser. The single best tag sequence \hat{T} will typically contain too many incorrect tags for effective parsing to take place. To overcome this, we instead return a probability distribution over the possible supertags for each word in the input. The following table illustrates an example distribution for a simple sentence, in which each column represents the probability of each supertag for a given word *in the context of the input sentence*. The “...” represent all the remaining supertags possible for each word.

United	serves	Denver
$N/N: 0.4$	$(S \setminus NP)/NP: 0.8$	$NP: 0.9$
$NP: 0.3$	$N: 0.1$	$N/N: 0.05$
$S/S: 0.1$
$S \setminus S: .05$		
...		

To get the probability of each possible word/tag pair, we'll need to sum the probabilities of all the supertag sequences that contain that tag at that location. This can be done with the forward-backward algorithm that is also used to train the CRF, described in Appendix A.

E.6.2 CCG Parsing using the A* Algorithm

The A* algorithm is a heuristic search method that employs an agenda to find an optimal solution. Search states representing partial solutions are added to an agenda based on a cost function, with the least-cost option being selected for further exploration at each iteration. When a state representing a complete solution is first selected from the agenda, it is guaranteed to be optimal and the search terminates.

The A* cost function, $f(n)$, is used to efficiently guide the search to a solution. The f -cost has two components: $g(n)$, the exact cost of the partial solution represented by the state n , and $h(n)$ a heuristic approximation of the cost of a solution that makes use of n . When $h(n)$ satisfies the criteria of not overestimating the actual cost, A* will find an optimal solution. Not surprisingly, the closer the heuristic can get to the actual cost, the more effective A* is at finding a solution without having to explore a significant portion of the solution space.

When applied to parsing, search states correspond to edges representing completed constituents. Each edge specifies a constituent's start and end positions, its

grammatical category, and its f -cost. Here, the g component represents the current cost of an edge and the h component represents an estimate of the cost to complete a derivation that makes use of that edge. The use of A* for phrase structure parsing originated with Klein and Manning (2003), while the CCG approach presented here is based on the work of Lewis and Steedman (2014).

Using information from a supertagger, an agenda and a parse table are initialized with states representing all the possible lexical categories for each word in the input, along with their f -costs. The main loop removes the lowest cost edge from the agenda and tests to see if it is a complete derivation. If it reflects a complete derivation it is selected as the best solution and the loop terminates. Otherwise, new states based on the applicable CCG rules are generated, assigned costs, and entered into the agenda to await further processing. The loop continues until a complete derivation is discovered, or the agenda is exhausted, indicating a failed parse. The algorithm is given in Fig. E.1.

```

function CCG-ASTAR-PARSE(words) returns table or failure

    supertags  $\leftarrow$  SUPERTAGGER(words)
    for i  $\leftarrow$  from 1 to LENGTH(words) do
        for all {A | (words[i], A, score)  $\in$  supertags} do
            edge  $\leftarrow$  MAKEEDGE(i - 1, i, A, score)
            table  $\leftarrow$  INSERTEDGE(table, edge)
            agenda  $\leftarrow$  INSERTEDGE(agenda, edge)
    loop do
        if EMPTY?(agenda) return failure
        current  $\leftarrow$  POP(agenda)
        if COMPLETEDPARSE?(current) return table
        table  $\leftarrow$  INSERTEDGE(table, current)
        for each rule in APPLICABLERULES(current) do
            successor  $\leftarrow$  APPLY(rule, current)
            if successor  $\notin$  agenda or chart
                agenda  $\leftarrow$  INSERTEDGE(agenda, successor)
            else if successor  $\in$  agenda with higher cost
                agenda  $\leftarrow$  REPLACEEDGE(agenda, successor)

```

Figure E.1 A*-based CCG parsing.

E.6.3 Heuristic Functions

Before we can define a heuristic function for our A* search, we need to decide how to assess the quality of CCG derivations. We'll make the simplifying assumption that the probability of a CCG derivation is just the product of the probability of the supertags assigned to the words in the derivation, ignoring the rules used in the derivation. More formally, given a sentence S and derivation D that contains supertag sequence T , we have:

$$P(D, S) = P(T, S) \quad (\text{E.10})$$

$$= \prod_{i=1}^n P(t_i | s_i) \quad (\text{E.11})$$

10 APPENDIX E • COMBINATORY CATEGORIAL GRAMMAR

To better fit with the traditional A* approach, we'd prefer to have states scored by a cost function where lower is better (i.e., we're trying to minimize the cost of a derivation). To achieve this, we'll use negative log probabilities to score derivations; this results in the following equation, which we'll use to score completed CCG derivations.

$$P(D, S) = P(T, S) \tag{E.12}$$

$$= \sum_{i=1}^n -\log P(t_i | s_i) \tag{E.13}$$

Given this model, we can define our f -cost as follows. The f -cost of an edge is the sum of two components: $g(n)$, the cost of the span represented by the edge, and $h(n)$, the estimate of the cost to complete a derivation containing that edge (these are often referred to as the **inside** and **outside costs**). We'll define $g(n)$ for an edge using Equation E.13. That is, it is just the sum of the costs of the supertags that comprise the span.

For $h(n)$, we need a score that approximates but *never overestimates* the actual cost of the final derivation. A simple heuristic that meets this requirement assumes that each of the words in the outside span will be assigned its *most probable supertag*. If these are the tags used in the final derivation, then its score will equal the heuristic. If any other tags are used in the final derivation the f -cost will be higher since the new tags must have higher costs, thus guaranteeing that we will not overestimate.

Putting this all together, we arrive at the following definition of a suitable f -cost for an edge.

$$\begin{aligned} f(w_{i,j}, t_{i,j}) &= g(w_{i,j}) + h(w_{i,j}) \\ &= \sum_{k=i}^j -\log P(t_k | w_k) + \\ &\quad \sum_{k=1}^{i-1} \min_{t \in \text{tags}} (-\log P(t | w_k)) + \sum_{k=j+1}^N \min_{t \in \text{tags}} (-\log P(t | w_k)) \end{aligned} \tag{E.14}$$

As an example, consider an edge representing the word *serves* with the supertag N in the following example.

(E.15) United serves Denver.

The g -cost for this edge is just the negative log probability of this tag, $-\log_{10}(0.1)$, or 1. The outside h -cost consists of the most optimistic supertag assignments for *United* and *Denver*, which are N/N and NP respectively. The resulting f -cost for this edge is therefore 1.443.

E.6.4 An Example

Fig. E.2 shows the initial agenda and the progress of a complete parse for this example. After initializing the agenda and the parse table with information from the supertagger, it selects the best edge from the agenda—the entry for *United* with the tag N/N and f -cost 0.591. This edge does not constitute a complete parse and is therefore used to generate new states by applying all the relevant grammar rules. In this case, applying forward application to *United*: N/N and *serves*: N results in the creation of the edge *United serves*: $N[0,2], 1.795$ to the agenda.

Skipping ahead, at the third iteration an edge representing the complete derivation *United serves Denver; S[0,3], .716* is added to the agenda. However, the algorithm does not terminate at this point since the cost of this edge (.716) does not place it at the top of the agenda. Instead, the edge representing *Denver* with the category *NP* is popped. This leads to the addition of another edge to the agenda (type-raising *Denver*). Only after this edge is popped and dealt with does the earlier state representing a complete derivation rise to the top of the agenda where it is popped, goal tested, and returned as a solution.

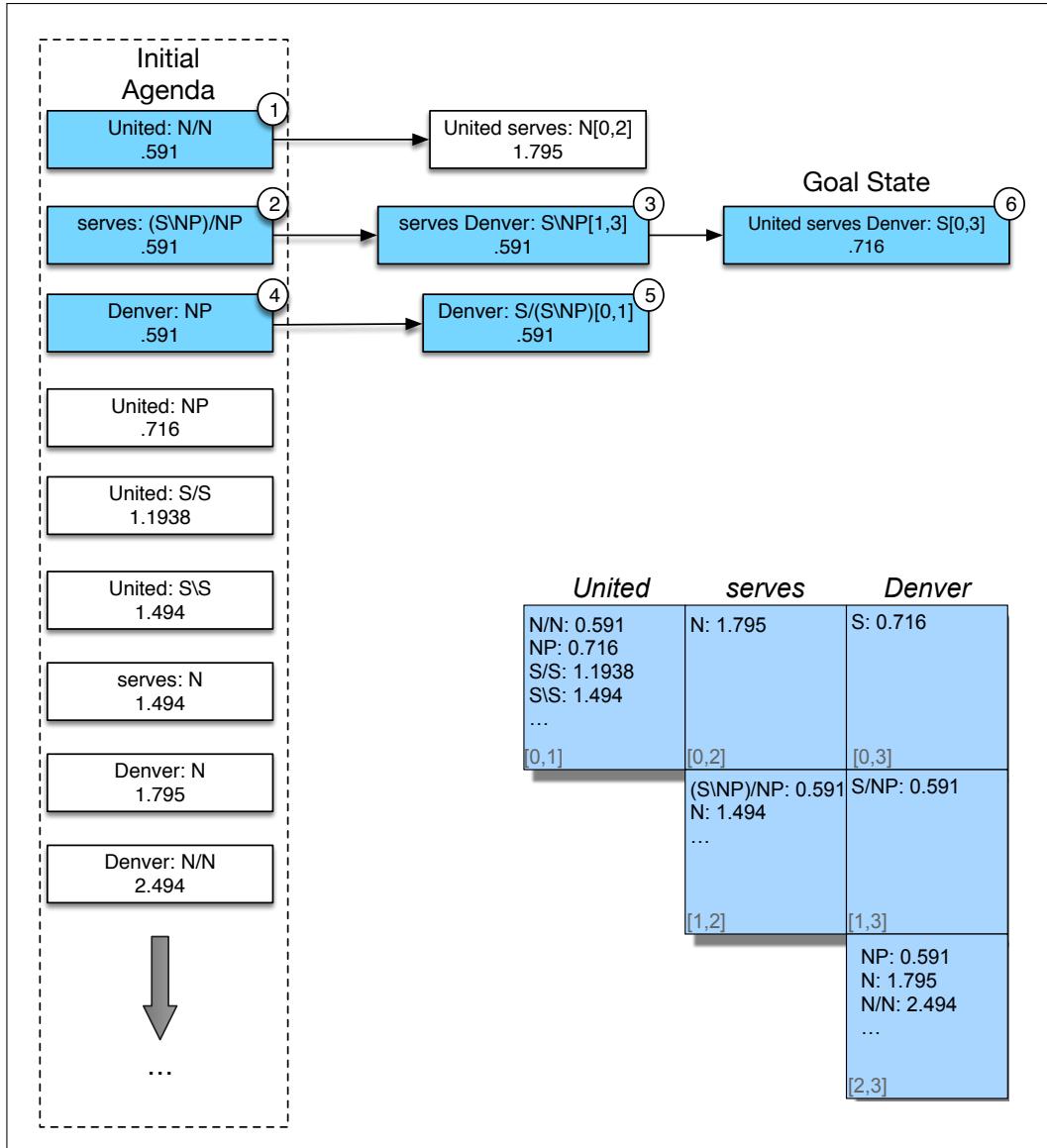


Figure E.2 Example of an A* search for the example “United serves Denver”. The circled numbers on the blue boxes indicate the order in which the states are popped from the agenda. The costs in each state are based on f-costs using negative \log_{10} probabilities.

The effectiveness of the A* approach is reflected in the coloring of the states in Fig. E.2 as well as the final parsing table. The edges shown in blue (including all the

12 APPENDIX E • COMBINATORY CATEGORIAL GRAMMAR

initial lexical category assignments not explicitly shown) reflect states in the search space that never made it to the top of the agenda and, therefore, never contributed any edges to the final table. This is in contrast to the PCKY approach where the parser systematically fills the parse table with all possible constituents for all possible spans in the input, filling the table with myriad constituents that do not contribute to the final analysis.

E.7 Summary

This chapter has introduced combinatory categorial grammar (CCG):

- Combinatorial categorial grammar (CCG) is a computationally relevant lexicalized approach to grammar and parsing.
- Much of the difficulty in CCG parsing is disambiguating the highly rich lexical entries, and so CCG parsers are generally based on **supertagging**.
- Supertagging is the equivalent of part-of-speech tagging in highly lexicalized grammar frameworks. The tags are very grammatically rich and dictate much of the derivation for a sentence.

Bibliographical and Historical Notes

- Ajdukiewicz, K. 1935. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27. English translation “Syntactic Connexion” by H. Weber in McCall, S. (Ed.) 1967. *Polish Logic*, pp. 207–231, Oxford University Press.
- Bangalore, S. and A. K. Joshi. 1999. [Supertagging: An approach to almost parsing](#). *Computational Linguistics*, 25(2):237–265.
- Bar-Hillel, Y. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.
- Bresnan, J., ed. 1982. *The Mental Representation of Grammatical Relations*. MIT Press.
- Hockenmaier, J. and M. Steedman. 2007. [CCGbank: a corpus of CCG derivations and dependency structures extracted from the penn treebank](#). *Computational Linguistics*, 33(3):355–396.
- Joshi, A. K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. Zwicky, eds, *Natural Language Parsing*, 206–250. Cambridge University Press.
- Klein, D. and C. D. Manning. 2003. [A* parsing: Fast exact Viterbi parse selection](#). *HLT-NAACL*.
- Lewis, M. and M. Steedman. 2014. [A* ccg parsing with a supertag-factored model](#). *EMNLP*.
- Pollard, C. and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.
- Steedman, M. 1989. Constituency and coordination in a combinatory grammar. In M. R. Baltin and A. S. Kroch, eds, *Alternative Conceptions of Phrase Structure*, 201–231. University of Chicago.
- Steedman, M. 1996. *Surface Structure and Interpretation*. MIT Press. Linguistic Inquiry Monograph, 30.
- Steedman, M. 2000. *The Syntactic Process*. The MIT Press.

Complete Axioms for Categorical Fixed-point Operators

Alex Simpson* and Gordon Plotkin†
*LFCS, Division of Informatics, University of Edinburgh,
JCMB, King's Buildings, Edinburgh, EH9 3JZ
{als,gdp}@dcs.ed.ac.uk*

Abstract

We give an axiomatic treatment of fixed-point operators in categories. A notion of iteration operator is defined, embodying the equational properties of iteration theories. We prove a general completeness theorem for iteration operators, relying on a new, purely syntactic characterisation of the free iteration theory.

We then show how iteration operators arise in axiomatic domain theory. One result derives them from the existence of sufficiently many bifree algebras (exploiting the universal property Freyd introduced in his notion of algebraic compactness). Another result shows that, in the presence of a parameterized natural numbers object and an equational lifting monad, any uniform fixed-point operator is necessarily an iteration operator.

1. Introduction

Fixed points play a central rôle in domain theory. Traditionally, one works with a category such as **Cppo**, the category of ω -continuous functions between ω -complete pointed partial orders. This possesses a least-fixed-point operator, whose properties are well understood. For example, a theorem of Bekić states that least simultaneous fixed points can be found in sequence by a form of Gaussian elimination, see e.g. [33]. More generally, the equational theory between fixed-point terms (μ -terms), induced by the least-fixed-point operator, has been axiomatized as the free *iteration theory* of Bloom and Ésik [3]. (This theory is known to be decidable.) Also, Eilenberg [6] and Plotkin [25] gave an order-free characterisation of the least-fixed-point operator as the unique fixed-point operator satisfying a condition known as *uniformity*, expressed with respect to the subcategory \mathbf{Cppo}_\perp of strict maps in **Cppo**, see e.g. [15].

Nowadays, one appreciates that **Cppo** is one of many possible categories of “domain-like” structures, each with

an associated fixed-point operator. Not only are there many familiar order-theoretic variations on the notions of complete partial order and continuous function, but there are also many categories of “domains” based on somewhat different principles — for example, categories of games and strategies [21], realizability-based categories [20] and categories of abstract geometric structures [12]. Thus one needs generally applicable methods for establishing properties of the associated fixed-point operators.

In this paper, we analyse the equational properties of fixed-point operators in arbitrary categories of “domain-like” structures. In Section 2, we consider the basic notions of *(parameterized) fixed-point operator*, *Conway operator* and *iteration operator*, developed from analogous notions in Bloom and Ésik’s study of iteration theories [3]. Our definitions are straightforward adaptations of Bloom and Ésik’s to the general setting of a category with finite products. In particular, the notion of *iteration operator* is intended to capture all desirable equational properties of a fixed-point operator, as exemplified by the many completeness results for the free iteration theory in [3].

As in the case of the fixed-point operator on **Cppo**, we also consider a notion of *(parameterized) uniformity* for (parameterized) fixed-point operators. We define this in general assuming a suitable functor $J : \mathcal{S} \rightarrow \mathcal{D}$ from a category \mathcal{S} of “strict” maps. In practice, (parameterized) uniformity serves two purposes. First, it is often satisfied by a unique (parameterized) fixed-point operator, and so characterises that operator. Second, any parametrically uniform Conway operator is an iteration operator, so parameterized uniformity is a convenient tool for establishing that the equations of an iteration operator are satisfied.

In Section 3, we examine the equational theory of iteration operators. We use a syntax of multisorted fixed-point terms (μ -terms), which can be interpreted in any category with an iteration operator. In any such category, Bloom and Ésik’s axioms for iteration theories [3] are sound. Bloom and Ésik provide numerous completeness theorems, demonstrating that the iteration theory axioms are also complete for deriving the valid equations in many familiar cat-

*Research supported by EPSRC grant GR/K06109.

†Research supported by EPSRC grant GR/M56333.

egories with iteration operators. The first main contribution of this paper is a precise characterisation of the circumstances in which the iteration theory axioms are complete (Theorem 1). This result accounts for all the examples in [3]. It shows that, in non-degenerate categories, the soundness of the iteration theory axioms implies their completeness. This explains the ubiquity of completeness results for the free iteration theory.

Our completeness theorem follows from a new, purely syntactic characterisation of the free iteration theory as a maximal theory satisfying two properties: *closed consistency* and *typical ambiguity* (Theorem 2). This result, which is of interest in its own right, was inspired by Statman’s characterisation of $\beta\eta$ -equality in the simply-typed λ -calculus [31].

The remainder of the paper is devoted to providing conditions for establishing the existence (and uniqueness) of parametrically uniform Conway operators (hence iteration operators). In one common setting, which arises in axiomatic domain theory [13, 10, 12], one has that the category \mathcal{D} of “domains” is obtained as the co-Kleisli category of a comonad on the category of strict maps \mathcal{S} . (For example, \mathbf{Cpo} is the co-Kleisli category of the lifting comonad on \mathbf{Cpo}_\perp .) In axiomatic domain theory, \mathcal{S} satisfies a curious property, first identified by Freyd [13, 14]: a wide class of endofunctors on \mathcal{S} have initial algebras whose inverses are final coalgebras (in Freyd’s terminology, \mathcal{S} is *algebraically compact*). Following [7], we call such initial/final algebras/coalgebras *bifree algebras*. (In the example of \mathbf{Cpo}_\perp , every \mathbf{Cpo} -enriched endofunctor has a bifree algebra [10].)

In Section 5, we give a quick overview of initial algebras, final coalgebras and bifree algebras, including a couple of minor new propositions. Then, in Section 6, we show how bifree algebras in \mathcal{S} can induce properties of fixed-point operators in \mathcal{D} . This programme was begun by Freyd and others [13, 5, 24, 28]. A further step was taken by Moggi, who, in unpublished work, gave a direct verification of the Bekič equality. Here, we give the complete story, showing how the presence of sufficiently many bifree algebras determines a unique parametrically uniform Conway operator (hence iteration operator).

In Section 7 we show how the Conway operator identities can be established without assuming the existence of the bifree algebras used in Section 6. This is possible when the category \mathcal{S} of “strict” maps arises as the category of algebras for a “lifting monad” on a suitable category of “pre-domains” \mathcal{C} . (For example, \mathbf{Cpo}_\perp is the category of algebras for the usual lifting monad on the category \mathbf{Cpo} of, not necessarily pointed, ω -complete partial orders.) Axiomatically, we assume that \mathcal{C} is a category with finite products, a monad embodying the equational properties of partial map classifiers (an *equational lifting monad* [4]), partial func-

tion spaces (*Kleisli exponentials* [22, 28]), and a (parameterized) natural numbers object. These conditions are always satisfied by the categories of predomains that arise in axiomatic and synthetic domain theory [10, 12, 20, 11, 26, 30]. Theorem 4 states that such categories support at most one uniform recursion operator (a *T-fixed-point operator*), and moreover it determines a unique parametrically uniform Conway operator on the associated category of domains. Thus, in the presence of a lifting monad and a parameterized natural numbers object, uniformity alone implies all equational properties of fixed points.

2. Fixed-point operators

In this section we give an overview of the various notions of fixed-point operator we shall be concerned with. We work with a category, \mathcal{D} , with distinguished finite products, to be thought of as a category of “domains”. We write $\mathbf{1}$ for the terminal object.

Definition 2.1 (Fixed-point operator) A *fixed-point operator* is a family of functions $(\cdot)^* : \mathcal{D}(A, A) \rightarrow \mathcal{D}(\mathbf{1}, A)$ such that, for any $f : A \longrightarrow A$, $f \circ f^* = f^*$.

Definition 2.2 (Parameterized fixed-pt. op.) A *parameterized fixed-point operator* is a family of functions $(\cdot)^\dagger : \mathcal{D}(X \times A, A) \rightarrow \mathcal{D}(X, A)$ satisfying:

1. (Naturality.)

For any $g : X \longrightarrow Y$ and $f : Y \times A \longrightarrow A$,
 $f^\dagger \circ g = (f \circ (g \times \text{id}_A))^\dagger : X \longrightarrow A$.

2. (Parameterized fixed-point property.)

For any $f : X \times A \longrightarrow A$,
 $f \circ \langle \text{id}_X, f^\dagger \rangle = f^\dagger : X \longrightarrow A$.

Observe that a parameterized fixed-point operator corresponds to a family of fixed-point operators $(\cdot)^{*x}$ in the co-Kleisli categories $\mathcal{D}_{X \times (\perp)}$ of $X \times (-)$ comonads. In this formulation, naturality states that, for any $g : X \longrightarrow Y$, the induced functor $H_g : \mathcal{D}_{Y \times (\perp)} \rightarrow \mathcal{D}_{X \times (\perp)}$ preserves the fixed-point operators in the sense that, for any endomorphism in $\mathcal{D}_{Y \times (\perp)}$, given by a morphism $f : Y \times A \longrightarrow A$ in \mathcal{D} , it holds that $H_g(f^{*Y}) = (H_g f)^{*x}$.

In practice, well-behaved fixed-point operators satisfy many other equations that do not follow from the fixed-point property alone.

Definition 2.3 (Dinaturality) A fixed-point operator is said to be *dinatural* if, for every $f : A \longrightarrow B$ and $g : B \longrightarrow A$, it holds that $(f \circ g)^* = f \circ (g \circ f)^*$.

Definition 2.4 (Conway operator) A *Conway operator* is a parameterized fixed-point operator that, in addition, satisfies:

3. (Parameterized dinaturality.)

For any $f : X \times B \longrightarrow A$ and $g : X \times A \longrightarrow B$,
 $f \circ \langle \text{id}_X, (g \circ \langle \pi_1, f \rangle)^\dagger \rangle = (f \circ \langle \pi_1, g \rangle)^\dagger : X \longrightarrow A$.

4. (Diagonal property.)

For any $f : X \times A \times A \longrightarrow A$, $(f \circ (\text{id}_X \times \Delta))^\dagger = (f^\dagger)^\dagger : X \longrightarrow A$ (where $\Delta : A \longrightarrow A \times A$ is the diagonal map).

It is easily seen that (parameterized) dinaturality implies the (parameterized) fixed-point property, so 2 of Definition 2.2 is redundant in the axiomatization of Conway operators.

The reason for singling out dinaturality is that it is a concept that makes sense for unparameterized fixed-point operators. It is also a powerful property. In special circumstances, it alone characterises a unique well-behaved fixed-point operator [29].

Mainly, however, we shall be interested in well behaved parameterized fixed-point operators, and the notion of Conway operator is appropriate. Conway operators are so named because their axioms correspond to those of the *Conway theories* of Bloom and Ésik [3]. They have also arisen independently in work of M. Hasegawa [16] and Hyland, who established a connection with Joyal, Street and Verity's notion of *trace* [18]. The definition of trace makes sense in any braided monoidal category. Hasegawa and Hyland showed that, in the special case that the monoidal product is cartesian, traces are in one-to-one correspondence with Conway operators.

There are many alternative axiomatizations for Conway operators. The axioms for a trace provide one possibility. Other options are discussed in [3, 16]. The following important property often appears in variant axiomatizations.

Proposition 2.5 (Bekić property) *For any Conway operator, given $\langle f, g \rangle : X \times A \times B \longrightarrow A \times B$, it holds that $\langle f, g \rangle^\dagger = \langle h, (g \circ \langle (\text{id}_X, h) \times \text{id}_B \rangle)^\dagger \rangle : X \longrightarrow A \times B$, where $h = (f \circ \langle \text{id}_{X \times A}, g^\dagger \rangle)^\dagger : X \longrightarrow A$.*

In spite of such consequences, there are basic equalities that Conway operators do not necessarily satisfy; for example, it is not true in general that $f^* = (f \circ f)^*$ for an endomorphism f . The *commutative identities* of Bloom and Ésik [3] ensure that such “missing” equalities do hold.

Definition 2.6 (Iteration operator) An *iteration operator* is a Conway operator that, in addition, satisfies:

5. (Commutative identities.)

Given $f : X \times A^m \longrightarrow A$ and morphisms $\rho_1, \dots, \rho_m : A^m \longrightarrow A^m$ such that each $\rho_i = \langle p_{i1}, \dots, p_{im} \rangle$ is a tuple of projections (i.e. each p_{ij} is one of the m projections $\pi_1, \dots, \pi_m : A^m \longrightarrow A$), it holds that

$$\langle f \circ (\text{id}_X \times \rho_1), \dots, f \circ (\text{id}_X \times \rho_m) \rangle^\dagger = \Delta_m \circ (f \circ (\text{id}_X \times \Delta_m))^\dagger : X \longrightarrow A^m$$

(Here A^m is the m -fold power $A \times \dots \times A$, and $\Delta_m : A \longrightarrow A^m$ is the diagonal.)¹

The complex formulation of the commutative identities means that they can be hard to establish in practice. One way of reducing the complexity is to look for simpler equational axiomatizations. For example, Ésik [9] has recently proved that it suffices to consider certain instances of the commutative identities generated, in an appropriate sense, by finite groups. However, in many situations, it is more convenient to derive the commutative identities from (more easily established) non-equational properties that imply them. Many examples of such properties are given by Bloom and Ésik [3]. In this paper we shall be concerned with one such property: (*parameterized*) *uniformity*.

To define (parameterized) uniformity, we suppose given a category \mathcal{S} with finite products and the same objects as \mathcal{D} , together with a functor $J : \mathcal{S} \rightarrow \mathcal{D}$ that strictly preserves finite products and is the identity on objects. We use the symbol \longrightarrow for maps in \mathcal{S} , and we call morphisms in \mathcal{D} in the image of J *strict*. (We really just need the subcategory of \mathcal{D} consisting of strict maps, but the functorial formulation will be helpful in Section 6. Observe that morphisms given purely by the finite product structure on \mathcal{D} are strict.)

Definition 2.7 (Uniformity) A fixed-point operator is said to be *uniform* (with respect to J) if, for any $f : A \longrightarrow A$, $g : B \longrightarrow B$ and $h : A \longrightarrow B$, $Jh \circ f = g \circ Jh$ implies $g^* = Jh \circ f^*$.

Definition 2.8 (Parameterized uniformity) A parameterized fixed-point operator is said to be *parametrically uniform* if, for any $f : X \times A \longrightarrow A$, $g : X \times B \longrightarrow B$ and $h : A \longrightarrow B$, $Jh \circ f = g \circ (\text{id}_X \times Jh)$ implies $g^\dagger = Jh \circ f^\dagger$.

Observe that parameterized uniformity is just the statement that the fixed-point operator $(\cdot)^{*x}$ in each co-Kleisli category $\mathcal{D}_{X \times (\perp)}$ is uniform with respect to the composite functor $\mathcal{S} \rightarrow \mathcal{D} \rightarrow \mathcal{D}_{X \times (\perp)}$. Hasegawa gives an interesting reformulation of parameterized uniformity directly in terms of a trace [16]. If \mathcal{S} is defined to be the subcategory of morphisms given purely by the finite product structure on \mathcal{D} , then parameterized uniformity is exactly the *functorial dagger implication for base morphisms* of [3].

Proposition 2.9 Any parametrically uniform Conway operator is an iteration operator.

The proof is an easy application of the strictness of all diagonals $\Delta_m : A \longrightarrow A^m$.

The converse to proposition 2.9 does not hold in general, see [8].

¹Strictly speaking, we consider only instances of the commutative identities of [3] in which their “surjective base morphism” ρ is a diagonal Δ_m . The general commutative identities of [3] follow from such instances, using properties of Conway operators.

3. Completeness

In this section we introduce Bloom and Ésik's *iteration theories* [3], using a syntax of multisorted fixed-point terms (μ -terms). We prove a very general completeness theorem (Theorem 1) for the free iteration theory relative to interpretations in categories with iteration operators. The completeness theorem follows from a new syntactic characterisation of the free iteration theory (Theorem 2).

We assume given a nonempty collection of base types (or sorts), over which α, β, \dots range. Types σ, τ, \dots are either base types or product types $\sigma_1 \times \dots \times \sigma_n$ (for $n \geq 0$). We use σ^n as an abbreviation for the n -fold power $\sigma \times \dots \times \sigma$. We assume also a signature given by a set Σ of function symbols, each with an associated typing information of the form $(\alpha_1, \dots, \alpha_n; \beta)$ (there is no loss of generality in considering only base types here). We loosely refer to both $(\alpha_1, \dots, \alpha_n; \beta)$ and n as the *arity* of the function symbol. Constants are considered as function symbols with arity 0. We assume a countably infinite set of variable symbols x, y, \dots . A variable is a pair, written x^σ , consisting of a variable symbol and a type (we omit the type superscript when convenient). Terms and their types are given by: each variable x^σ is a term of type σ ; if t_1, \dots, t_n are terms of (base) types $\alpha_1, \dots, \alpha_n$ and f is a function symbol of arity $(\alpha_1, \dots, \alpha_n; \beta)$ then $f(t_1, \dots, t_n)$ is a term of (base) type β ; if t_1, \dots, t_n are terms of types $\sigma_1, \dots, \sigma_n$ then $\langle t_1, \dots, t_n \rangle$ is a term of type $\sigma_1 \times \dots \times \sigma_n$; if t is a term of type $\sigma_1 \times \dots \times \sigma_n$ then $\pi_i t$, where $1 \leq i \leq n$, is a term of type σ_i ; if t is a term of type σ then $\mu x^\sigma. t$ is a term of type σ . As usual, the variable x is bound by μ in $\mu x. t$. We identify terms up to α -equivalence, writing $t \equiv t'$ for the identity of terms. We write $t(x_1^{\sigma_1}, \dots, x_k^{\sigma_k}) : \tau$ for a term of type τ all of whose free variables are contained in $x_1^{\sigma_1}, \dots, x_k^{\sigma_k}$. We call a term with no free variables *closed*. We write the substitution of n terms t_1, \dots, t_n for n distinct free variables x_1, \dots, x_n (of the correct types) in a term t as $t[t_1, \dots, t_n / x_1, \dots, x_n]$. Given $t(\vec{x}, x_1^{\sigma_1}, \dots, x_n^{\sigma_n}) : \sigma_1 \times \dots \times \sigma_n$, we use the convenient notation $\mu\langle x_1^{\sigma_1}, \dots, x_n^{\sigma_n} \rangle. t$ to represent the term $\mu x^{\sigma_1 \times \dots \times \sigma_n}. t[\pi_1 x, \dots, \pi_n x / x_1, \dots, x_n]$.

A theory, \mathcal{T} , is a typed congruence relation on terms that: contains the product equations, i.e. $\mathcal{T} \vdash \pi_i \langle t_1, \dots, t_k \rangle = t_i$ and $\mathcal{T} \vdash t = \langle \pi_1 t, \dots, \pi_k t \rangle$ (for $t : \sigma_1 \times \dots \times \sigma_k$); and is closed under substitution (i.e. if $\mathcal{T} \vdash t = t'$ and $s : \sigma$ then $\mathcal{T} \vdash t[s/x^\sigma] = t'[s/x^\sigma]$). For any theory, $\mathcal{T} \vdash t = t'$ if and only if $\mathcal{T} \vdash (t = t')[\langle x_1^{\sigma_1}, \dots, x_n^{\sigma_n} \rangle / x^{\sigma_1 \times \dots \times \sigma_n}]$ where x_1, \dots, x_n are fresh variables. Thus a theory is determined by its equations between terms whose only free variables are of base type. We say that \mathcal{T} is *consistent* if there are two terms t, t' of the same type such that $\mathcal{T} \not\vdash t = t'$. We say that \mathcal{T} is *closed-consistent* if there are two such terms that are closed.

We now axiomatize *Conway theories*, in which μ corresponds to a Conway operator, and *iteration theories*, identifying the equational properties of an iteration operator.

Definition 3.1 (Conway theory) A theory \mathcal{T} is said to be a *Conway theory* if it satisfies two axioms:

1. (Dinaturality.)

For any $t(\vec{z}, y^\tau) : \sigma$ and $t'(\vec{z}, x^\sigma) : \tau$,
 $\mathcal{T} \vdash \mu x. t[t'/y] = t[\mu y. t'[x/y]] : \sigma$.

2. (Diagonal property.)

For $t(\vec{z}, x^\sigma, y^\sigma) : \sigma$, $\mathcal{T} \vdash \mu x. t[x/y] = \mu x. \mu y. t : \sigma$.

These axioms are just the multisorted version of the axiomatization given by Corollary 6.2.5 of [3], where dinaturality and the diagonal property are called the *composition identity* and the *double dagger identity* respectively.

Definition 3.2 (Iteration theory) We say that a Conway theory \mathcal{T} is an *iteration theory* if it satisfies the following axiom schema.

3. (Amalgamation.)

For any terms $t_1, \dots, t_n(\vec{z}, x_1^\sigma, \dots, x_n^\sigma) : \sigma$ and $s(\vec{z}, y^\sigma) : \sigma$, suppose $t_i[y, \dots, y / x_1, \dots, x_n] \equiv s$, for all i with $1 \leq i \leq n$, then it follows that $\mathcal{T} \vdash \mu\langle x_1, \dots, x_n \rangle. \langle t_1, \dots, t_n \rangle = \langle \mu y. s, \dots, \mu y. s \rangle : \sigma^n$.

Amalgamation is very close to the commutative identities of [3] (as in Definition 2.6). An alternative formulation is employed in [17], whose *alphabetic identification identity* is equivalent to amalgamation.

We write \mathcal{F} for the smallest Conway theory (generated by the given base types and signature), and \mathcal{I} for the smallest iteration theory. As is shown in [3], \mathcal{I} completely captures the valid identities in a wide class of models, including **Cppo**. We write \mathcal{I}^* for the smallest iteration theory in which all closed terms (with identical types) are equated. Although \mathcal{I}^* is not a closed-consistent theory, it is nonetheless consistent. In fact, \mathcal{I}^* exactly captures the valid identities in **Cppo**. Our aim in this section is to prove a general completeness theorem, accounting for all such completeness results for \mathcal{I} and \mathcal{I}^* .

First, we consider the interpretation of μ -terms in any category \mathcal{D} with finite products and a family of functions $(\cdot)^\dagger : \mathcal{D}(X \times A, A) \rightarrow \mathcal{D}(X, A)$ satisfying the naturality property of Definition 2.2. An *interpretation* is determined by a function $\llbracket \cdot \rrbracket$ from base types to objects of \mathcal{D} , together with a function mapping each function symbol f , of arity $(\alpha_1, \dots, \alpha_k; \beta)$, to a map $\llbracket f \rrbracket : \llbracket \alpha_1 \rrbracket \times \dots \times \llbracket \alpha_k \rrbracket \longrightarrow \llbracket \beta \rrbracket$. The first function extends (using products) to one from arbitrary types to objects of \mathcal{D} , and the second extends to one mapping any term $t(x_1^{\sigma_1}, \dots, x_k^{\sigma_k}) : \tau$ to a morphism

$\llbracket t \rrbracket : \llbracket \sigma_1 \rrbracket \times \dots \times \llbracket \sigma_k \rrbracket \longrightarrow \llbracket \tau \rrbracket$. Such an interpretation determines a theory $\mathcal{T}_{\llbracket \cdot \rrbracket}$ defined by $\mathcal{T}_{\llbracket \cdot \rrbracket} \vdash t(\vec{x}) = t'(\vec{x})$ iff $\llbracket t \rrbracket = \llbracket t' \rrbracket$. We have no reason to favour one interpretation in \mathcal{D} over another, so we shall be more interested in the theory determined by the category itself. This theory, $\mathcal{T}_{\mathcal{D}}$, is defined by $\mathcal{T}_{\mathcal{D}} \vdash t(\vec{x}) = t'(\vec{x})$ iff, for all interpretations $\llbracket \cdot \rrbracket$ in \mathcal{D} , it holds that $\llbracket t \rrbracket = \llbracket t' \rrbracket$.

It is immediate from the definitions that if $(\cdot)^\dagger$ is a Conway operator then $\mathcal{T}_{\mathcal{D}}$ is a Conway theory (i.e. $\mathcal{F} \subseteq \mathcal{T}_{\mathcal{D}}$). It is also straightforward (although not quite immediate) that if $(\cdot)^\dagger$ is an iteration operator then $\mathcal{T}_{\mathcal{D}}$ is an iteration theory (i.e. $\mathcal{I} \subseteq \mathcal{T}_{\mathcal{D}}$). Further, if every hom-set $\mathcal{D}(1, X)$ is a singleton² (i.e. if 1 is a zero object in \mathcal{D}) then $\mathcal{I}^* \subseteq \mathcal{T}_{\mathcal{D}}$.

Theorem 1 (Completeness) *If $(\cdot)^\dagger$ on \mathcal{D} is an iteration operator then:*

1. *If there exists a hom-set $\mathcal{D}(1, X)$ containing at least two distinct morphisms then $\mathcal{T}_{\mathcal{D}} = \mathcal{I}$.*
2. *If 1 is a zero object and there exists a hom-set $\mathcal{D}(X, Y)$ containing two distinct morphisms then $\mathcal{T}_{\mathcal{D}} = \mathcal{I}^*$.*

The only examples not captured by one of the conditions above are categories \mathcal{D} equivalent to the terminal category, in which case $\mathcal{T}_{\mathcal{D}}$ is the inconsistent theory.

We prove the theorem by obtaining a syntactic characterisation of the free iteration theory. Suppose that $(\cdot)^*$ is any function from base types to types. Suppose also that θ is a function mapping each function symbol f of arity $(\alpha_1, \dots, \alpha_n; \beta)$ to a term $f\theta(z_1^{\alpha_1^*}, \dots, z_n^{\alpha_n^*}) : \beta^*$. Then $(\cdot)^*$ extends (by substitution) to an endofunction on types. Similarly, θ extends to a function on terms, mapping each $t(x_1^{\sigma_1}, \dots, x_k^{\sigma_k}) : \tau$ to a term $t\theta(x_1^{\sigma_1^*}, \dots, x_k^{\sigma_k^*}) : \tau^*$, defined inductively on the structure of t by:

$$\begin{aligned} x^\sigma \theta &= x^{\sigma^*} \\ f(t_1, \dots, t_n)\theta &= f\theta[t_1\theta, \dots, t_n\theta / z_1, \dots, z_n] \\ (\mu x^\sigma. t)\theta &= \mu x^{\sigma^*}. (t\theta) \\ \langle t_1, \dots, t_n \rangle \theta &= \langle t_1\theta, \dots, t_n\theta \rangle \\ (\pi_i t)\theta &= \pi_i(t\theta). \end{aligned}$$

We say that \mathcal{T} is *typically ambiguous* if $\mathcal{T} \vdash t = t' : \sigma$ implies, for all $(\cdot)^*$ and θ as above, $\mathcal{T} \vdash t\theta = t'\theta : \sigma^*$. In the terminology of [9], \mathcal{T} is typically ambiguous iff it contains the *vector forms* of all its equations. \mathcal{F} , \mathcal{I} and \mathcal{I}^* are all examples of typically ambiguous theories. In each case, one proves that $t = t'$ implies $t\theta = t'\theta$ by a straightforward induction on the derivation of $t = t'$.

Theorem 2

1. *The only consistent typically ambiguous iteration theories are \mathcal{I} and \mathcal{I}^* .*

²Note that the existence of $(\cdot)^\dagger$ implies that $\mathcal{D}(1, X)$ is always nonempty.

2. *If \mathcal{T} is a closed-consistent typically ambiguous Conway theory then $\mathcal{T} \subseteq \mathcal{I}$.*
3. *If \mathcal{T} is a consistent typically ambiguous Conway theory then $\mathcal{T} \subseteq \mathcal{I}^*$.*

The proof is outlined in the next section.

The theorem characterises \mathcal{I}^* as the greatest element in the partially ordered set of consistent typically ambiguous Conway theories (ordered by inclusion). Also, when Σ is nonempty, \mathcal{I} is the maximum closed-consistent typically ambiguous Conway theory. (If Σ is empty then \mathcal{I} is not closed-consistent, indeed $\mathcal{I} = \mathcal{I}^*$.) For countable Σ , there are 2^{\aleph_0} typically ambiguous Conway theories (this follows from the analysis of the group identities in [9]).

Theorem 1 follows easily from Theorem 2. First observe that for any \mathcal{D} and $(\cdot)^\dagger$ (natural in X), the theory $\mathcal{T}_{\mathcal{D}}$ is always typically ambiguous (because any $(\cdot)^*$ and θ determine a mapping between interpretations). Also $\mathcal{T}_{\mathcal{D}}$ is consistent if and only if \mathcal{D} is non-trivial (i.e. not equivalent to the terminal category); and it is closed-consistent if and only if Σ is nonempty and there exists a hom-set $\mathcal{D}(1, X)$ with at least two distinct elements. With these observations, Theorem 1 is immediate.

4. Proof of Theorem 2

The first part of the proof follows the standard proof of the completeness of the free iteration theory relative to an interpretation in regular trees, see, in particular, Sections 5.4 and 6.4–5 of [3] (see also the recent [17]). We outline the main steps, because the associated notion of normal form and its properties are needed for Lemma 4.3.

The notion of normal form is defined for terms $s(x_1^{\alpha_1}, \dots, x_k^{\alpha_k}) : \sigma$ (with only free variables of base type), by induction on the structure of σ . For base types β , the normal forms are terms $\pi_1(\mu(y_1^{\beta_1}, \dots, y_m^{\beta_m}). \langle u_1, \dots, u_m \rangle)$ where β_1 is β such that, for each i with $1 \leq i \leq m$, one of three possibilities holds for u_i : (i) u_i is y_i (note it cannot be y_j for $j \neq i$); (ii) u_i is a free variable from x_1, \dots, x_k ; (iii) u_i is of the form $f_i(y_{p_i(1)}, \dots, y_{p_i(a_i)})$ for some function symbol f_i (with arity a_i) and function $p_i : \{1, \dots, a_i\} \rightarrow \{1, \dots, m\}$. For product types $\sigma_1 \times \dots \times \sigma_n$ the normal forms are terms $\langle s_1, \dots, s_n \rangle$ where each s_i is in normal form. The restriction to free variables of base type is just a syntactic convenience to avoid considering additional sub-terms (such as projections on variables of product type).

Lemma 4.1 *For any term $t(x_1^{\alpha_1}, \dots, x_k^{\alpha_k}) : \sigma$, there exists a term s in normal form such that $\mathcal{F} \vdash t = s : \sigma$.*

We next define two interpretations of normal forms of base type as (regular) trees. Consider a term $u(x_1^{\alpha_1}, \dots, x_k^{\alpha_k}) : \beta_1 \times \dots \times \beta_m$ of the form

$\mu\langle y_1, \dots, y_m \rangle. \langle u_1, \dots, u_m \rangle$ subject to the restrictions imposed in the definition of normal form. For mnemonic benefit, we henceforth use y_i (where $1 \leq i \leq m$) to represent the term $\pi_i(u)$. To each such term y_i we assign trees, $\llbracket y_i \rrbracket$ and $\llbracket y_i \rrbracket^*$, whose nodes are labelled by elements from the set $\Sigma^+ = \Sigma \cup \{-\beta_1, \dots, -\beta_m\} \cup \{x_1^{\alpha_1}, \dots, x_k^{\alpha_k}\}$.

Formally, such a tree, t , is a partial function from \mathbb{N}^* (finite sequences of natural numbers) to Σ^+ whose domain of definition is nonempty, closed under prefixes and satisfies: for any $w \in \mathbb{N}^*$, $t(wn)$ is defined if and only if $t(w) \in \Sigma$, $1 \leq n \leq \text{arity}(t(w))$ and the result type of $t(wn)$ is the n -th argument type of $t(w)$. Given such a tree, t , and an element $w \in \mathbb{N}^*$ such that $t(w)$ is defined, we write $t@w$ for the subtree $w' \mapsto t(ww')$. We say that t is *regular* if the set of all its subtrees, $\{t@w \mid t(w) \text{ is defined}\}$, is finite.

To define $\llbracket y_i \rrbracket$, we first define a partial function ρ_i from \mathbb{N}^* to $\{1, \dots, m\}$. On the empty sequence, ϵ , define $\rho_i(\epsilon) = i$. On a nonempty sequence wn , $\rho_i(wn)$ is defined iff $\rho_i(w)$ is defined, $u_{\rho_i(w)}$ is $f_j(y_{p_j(1)}, \dots, y_{p_j(a_j)})$ and $1 \leq n \leq a_j$; in which case $\rho_i(wn) = p_j(n)$. Finally, we define a function $\lambda : \{1, \dots, m\} \rightarrow \Sigma^+$ by: if u_i is y_i then $\lambda(i) = -\beta_i$; if u_i is a free variable x then $\lambda(i) = x$; and if u_i is $f_i(y_{p_i(1)}, \dots, y_{p_i(a_i)})$ then $\lambda(i) = f_i$. The desired tree $\llbracket y_i \rrbracket$ is defined as the composition $\llbracket y_i \rrbracket = \lambda \circ \rho_i$.

We define $\llbracket y_i \rrbracket^*$ from $\llbracket y_i \rrbracket$ as follows. Say that w is *open* in $\llbracket y_i \rrbracket$ if there exists w' such that $\llbracket y_i \rrbracket(ww') \in \{x_1^{\alpha_1}, \dots, x_k^{\alpha_k}\}$. Say that w is *closed* in $\llbracket y_i \rrbracket$ if $\llbracket y_i \rrbracket(w)$ is defined and w is not open. Say that w is *minimally closed* if it is closed, but no proper prefix is. Then $\llbracket y_i \rrbracket^*(w)$ is defined if and only if w is either open in $\llbracket y_i \rrbracket$ or minimally closed. In the case that w is open, define $\llbracket y_i \rrbracket^*(w) = \llbracket y_i \rrbracket(w)$. In the case that w is minimally closed, define $\llbracket y_i \rrbracket^*(w) = -\beta_{\rho_i(w)}$.

Given a normal form of base type, $s(x_1^{\alpha_1}, \dots, x_k^{\alpha_k}) : \beta$, we have that $s \equiv \pi_1(u)$ for some u of the form above. Accordingly, we have trees $\llbracket s \rrbracket = \llbracket y_1 \rrbracket$ and $\llbracket s \rrbracket^* = \llbracket y_1 \rrbracket^*$ associated to s itself.

Lemma 4.2 *For normal forms $s, s'(x_1^{\alpha_1}, \dots, x_k^{\alpha_k}) : \beta$:*

1. *If $\llbracket s \rrbracket = \llbracket s' \rrbracket$ then $\mathcal{I} \vdash s = s'$.*
2. *If $\llbracket s \rrbracket^* = \llbracket s' \rrbracket^*$ then $\mathcal{I}^* \vdash s = s'$.*

The lemma is proved by defining, for every regular tree t , a canonical normal form t_t . One then proves that, for every normal form s , $\mathcal{I} \vdash s = t_{\llbracket s \rrbracket}$ and $\mathcal{I}^* \vdash s = t_{\llbracket s \rrbracket^*}$. This requires various consequences of amalgamation. Similar arguments can be found in Sections 5.4 and 6.4–5 of [3] and also in [17]. The lemma follows.

The argument thus far has established the known completeness of the iteration theory axioms relative to regular trees. To prove Theorem 2, we show that distinct regular trees can never be identified in a typically ambiguous way without losing (closed) consistency. The proof adapts the “Böhm-out” method from the λ -calculus [1].

Lemma 4.3 *Suppose \mathcal{T} is a typically-ambiguous Conway Theory, $s, s'(x_1^{\alpha_1}, \dots, x_k^{\alpha_k}) : \beta$ are normal forms and $\mathcal{T} \vdash s = s'$.*

1. *If \mathcal{T} is closed-consistent then $\llbracket s \rrbracket = \llbracket s' \rrbracket$.*
2. *If \mathcal{T} is consistent then $\llbracket s \rrbracket^* = \llbracket s' \rrbracket^*$.*

PROOF For statement 1, suppose $\llbracket s \rrbracket \neq \llbracket s' \rrbracket$. Let $r : \sigma$ be any closed term. We show that $\mathcal{T} \vdash r = \mu y^\sigma. y$. Thus \mathcal{T} is not closed-consistent.

Let $w \in \mathbb{N}^*$ be a sequence of minimum length such that $\llbracket s \rrbracket(w) \neq \llbracket s' \rrbracket(w)$ (both $\llbracket s \rrbracket(w)$ and $\llbracket s' \rrbracket(w)$ are therefore defined). Without loss of generality, we assume that $\llbracket s \rrbracket(w) \neq -$. We shall define a suitable $(\cdot)^*$ and θ (as in the definition of typical ambiguity) allowing us to bring the disagreement between $\llbracket s \rrbracket$ and $\llbracket s' \rrbracket$ up to the top level.

We write w as $n_1 \dots n_d$ (where $d \geq 0$), and w_i for its prefix $n_1 \dots n_i$ (where $0 \leq i \leq d$). For each base type α , define $\alpha^* = \sigma^{d+1}$, where σ is the type of the closed term r . For each function symbol f , of arity $(\alpha_1, \dots, \alpha_n; \beta')$, we define $f\theta(z_1^{\sigma^{d+1}}, \dots, z_n^{\sigma^{d+1}}) : \sigma^{d+1}$ by $f\theta = \langle u_1^f, \dots, u_{d+1}^f \rangle$ where:

$$u_i^f = \begin{cases} \pi_{i+1}(z_{n_i}) & \text{if } 1 \leq i \leq d \text{ and } \llbracket s \rrbracket(w_{i+1}) = f \\ r & \text{if } i = d+1 \text{ and } \llbracket s \rrbracket(w) = f \\ \mu y^\sigma. y & \text{otherwise.} \end{cases}$$

Then, as in the definition of typical ambiguity, θ determines terms $s\theta, s'\theta(x_1^{\sigma^{d+1}}, \dots, x_k^{\sigma^{d+1}})$ of type σ^{d+1} . By typical ambiguity, $\mathcal{T} \vdash s\theta = s'\theta$. Define terms $t_1, \dots, t_k : \sigma^{d+1}$ by: t_i is $\langle \mu y^\sigma. y, \dots, \mu y^\sigma. y \rangle$ if $\llbracket s \rrbracket(w) \neq x_i$, and t_i is $\langle r, \dots, r \rangle$ if $\llbracket s \rrbracket(w) = x_i$ (only the last component of these tuples is important). We write $[\vec{t}/\vec{x}]$ for the substitution $[t_1, \dots, t_k / x_1^{\sigma^{d+1}}, \dots, x_k^{\sigma^{d+1}}]$. By the substitution property of theories, $\mathcal{T} \vdash s\theta[\vec{t}/\vec{x}] = s'\theta[\vec{t}/\vec{x}]$. However, we prove below that $\mathcal{F} \vdash \pi_1(s\theta[\vec{t}/\vec{x}]) = r$ and $\mathcal{F} \vdash \pi_1(s'\theta[\vec{t}/\vec{x}]) = \mu y^\sigma. y$. Thus indeed $\mathcal{T} \vdash r = \mu y^\sigma. y$.

For the proof that $\mathcal{F} \vdash \pi_1(s\theta[\vec{t}/\vec{x}]) = r$, suppose s is $\pi_1(u)$ where u is a term of the form $\mu\langle y_1, \dots, y_m \rangle. \langle u_1, \dots, u_m \rangle$. As earlier, we write y_i for the term $\pi_i(u)$. We write ρ for the partial function ρ_1 from \mathbb{N}^* to $\{1, \dots, m\}$ used in the definition of $\llbracket s \rrbracket$. By working from $i = d+1$ down to $i = 1$, one calculates that, for any i with $1 \leq i \leq d+1$, it holds that

$$\mathcal{F} \vdash \pi_i(y_{\rho(w_{i-1})} \theta[\vec{t}/\vec{x}]) = r.$$

The desired equality is just the special case $i = 1$.

The proof that $\mathcal{F} \vdash \pi_1(s'\theta[\vec{t}/\vec{x}]) = \mu y^\sigma. y$ is very similar, again proving an equality for $i = d+1$ which propagates down to the desired case for $i = 1$, using the normal form expansion of s' . The crucial fact required in the argument is that $\llbracket s' \rrbracket(w_{i+1}) = \llbracket s \rrbracket(w_{i+1})$ if and only if $i \leq d$, which follows from the choice of w .

For statement 2, suppose $\llbracket s \rrbracket^* \neq \llbracket s' \rrbracket^*$. Let x^σ be any variable. We show that $\mathcal{T} \vdash x^\sigma = \mu y^\sigma. y$. Thus \mathcal{T} is not consistent.

Let $w \in \mathbb{N}^*$ be a sequence such that: (i) $\llbracket s \rrbracket^*(w) \in \{x_1^{\alpha_1}, \dots, x_k^{\alpha_k}\}$; and (ii) $\llbracket s' \rrbracket^*(w)$ is undefined or $\llbracket s' \rrbracket^*(w) \neq \llbracket s \rrbracket^*(w)$. (Such a sequence w can always be found, by swapping s and s' if necessary.)

As before, write w as $n_1 \dots n_d$ (where $d \geq 0$), and w_i for $n_1 \dots n_i$ (where $0 \leq i \leq d$). For each base type α , define $\alpha^* = \sigma^{d+1}$. For each function symbol f , of arity $(\alpha_1, \dots, \alpha_n; \beta')$, we define $f\theta(z_1^{\sigma^{d+1}}, \dots, z_n^{\sigma^{d+1}}) : \sigma^{d+1}$ by $f\theta = \langle u_1^f, \dots, u_{d+1}^f \rangle$ where:

$$u_i^f = \begin{cases} \pi_{i+1}(z_{n_i}) & \text{if } \llbracket s \rrbracket^*(w_{i+1}) = f \\ \mu y^\sigma. y & \text{otherwise.} \end{cases}$$

θ determines terms $s\theta, s'\theta(x_1^{\sigma^{d+1}}, \dots, x_k^{\sigma^{d+1}}) : \sigma^{d+1}$. By typical ambiguity, $\mathcal{T} \vdash s\theta = s'\theta$. Define $t_1, \dots, t_k : \sigma^{d+1}$ by: t_i is $\langle \mu y^\sigma. y, \dots, \mu y^\sigma. y \rangle$ if $\llbracket s \rrbracket^*(w) \neq x_i$, and t_i is $\langle \mu y^\sigma. y, \dots, \mu y^\sigma. y, x^\sigma \rangle$ if $\llbracket s \rrbracket^*(w) = x_i$. By substitution, $\mathcal{T} \vdash s\theta[\vec{t}/\vec{x}] = s'\theta[\vec{t}/\vec{x}]$. Then, much as above, $\mathcal{F} \vdash \pi_1(s\theta[\vec{t}/\vec{x}]) = x^\sigma$ and $\mathcal{F} \vdash \pi_1(s'\theta[\vec{t}/\vec{x}]) = \mu y^\sigma. y$. Thus $\mathcal{T} \vdash x^\sigma = \mu y^\sigma. y$ as required. \square

We now complete the proof of Theorem 2. We have already seen that \mathcal{I} and \mathcal{I}^* are typically ambiguous Conway theories. Consistency can be shown easily by giving a non-trivial semantics. To show that \mathcal{I} contains any closed-consistent typically-ambiguous Conway theory, let \mathcal{T} be any such theory. We must show that $\mathcal{T} \vdash t = t'$ implies $\mathcal{I} \vdash t = t'$. As \mathcal{T} is determined by its equations between terms whose only free variables are of base type, it suffices to show the implication for such terms t, t' . Suppose then that $\mathcal{T} \vdash t = t' : \sigma$. By Lemma 4.1, there exist normal forms s, s' such that $\mathcal{F} \vdash t = s$ and $\mathcal{F} \vdash t' = s'$, so also $\mathcal{T} \vdash s = s'$. One shows, by induction on σ , that $\mathcal{I} \vdash s = s'$ hence $\mathcal{I} \vdash t = t'$. If σ is a base type then this follows from Lemmas 4.3 and 4.2. For product types, the induction step is easy. The maximality of \mathcal{I}^* amongst consistent typically-ambiguous Conway theories is established similarly.

5. Initial and bifree algebras

In the remainder of the paper, we show that parametrically uniform Conway operators arise from universal properties in axiomatic approaches to semantics. A principal tool we use is the notion of *bifree algebra*, embodying the fundamental universal property introduced by Freyd in his work on algebraic compactness [13], which combines the properties of initial algebras and final coalgebras. In this section, we briefly review the relevant concepts.

Given an endofunctor F on a category \mathcal{C} , an *F-algebra* is a morphism $a : FA \rightarrow A$. An *F-algebra homomorphism* from $a : FA \rightarrow A$ to $b : FB \rightarrow B$ is a morphism $f : A \rightarrow B$ such that $f \circ a = b \circ Ff$. An *initial F-algebra* is an initial object in the category of *F-algebras* and homomorphisms.

The results below summarise some properties of initial algebras. Propositions 5.3 and 5.4 appear to be new.

Proposition 5.1 (Lambek) *If $a : FA \rightarrow A$ is an initial F-algebra then a is an isomorphism.*

Proposition 5.2 (Freyd [13]) *If F^2 has an initial algebra then F has an initial algebra, $a : FA \rightarrow A$ say, and $a \circ Fa$ is an initial F^2 -algebra. Conversely, if \mathcal{C} has binary products and F has an initial algebra then so does F^2 .*

Proposition 5.3 *For functors $F : \mathcal{C} \rightarrow \mathcal{D}$ and $G : \mathcal{D} \rightarrow \mathcal{C}$ between any two categories \mathcal{C} and \mathcal{D} , if $a : GF A \rightarrow A$ is an initial GF -algebra in \mathcal{C} then Fa is an initial FG -algebra in \mathcal{D} .*

Proposition 5.4 *Suppose idempotents in \mathcal{C} split, F is a retract of G in the category of endofunctors on \mathcal{C} , and G has an initial algebra $b : GB \rightarrow B$. Then F also has an initial algebra $a : FA \rightarrow A$ where A is a retract of B .*

An *F-coalgebra* is a morphism $a' : A \rightarrow FA$, and a *final F-coalgebra* is a terminal object in the evident category of *F-coalgebras* and homomorphisms. A *bifree algebra* for F is a morphism $a : FA \rightarrow A$ such that a is an initial *F-algebra* and $a^{\perp 1}$ is a final *F-coalgebra* (a is an isomorphism by Lambek's result above). All the propositions above have evident analogues applying to final coalgebras and bifree algebras.

One word of warning, in addition to algebras and coalgebras for endofunctors (as discussed above), we shall also make considerable use of algebras for monads and coalgebras for comonads. To avoid ambiguity, we shall always refer to such (co)algebras as (co)monad (co)algebras.

6. Fixed points from bifree algebras

In this section we show how parametrically uniform Conway operators arise in axiomatic domain theory. We work in a very general setting in which the category \mathcal{D} of “domains” arises as the co-Kleisli category of a comonad on the category \mathcal{S} of “strict maps”.

Suppose then that \mathcal{S} is a category with finite products, and (L, ε, δ) is a comonad on \mathcal{S} . We write \mathcal{D} for the co-Kleisli category of the comonad, $J : \mathcal{S} \rightarrow \mathcal{D}$ for the functor which is the right-adjoint part of the pair of adjoint functors determined by (and determining) the comonad, and $K : \mathcal{D} \rightarrow \mathcal{S}$ for the left-adjoint part. Then J is the identity on objects and, as it is a right adjoint, \mathcal{D} has finite products and J preserves them. Thus we are in a situation to apply

the concepts introduced in Section 2. As there, we use the symbol \longrightarrow for morphisms in \mathcal{S} . We shall find conditions on \mathcal{S} and L such that \mathcal{D} has a parametrically uniform Conway operator, indeed a unique one.

We begin with a fundamental proposition relating bifree algebras and fixed-point operators.

Proposition 6.1 *If $L : \mathcal{S} \rightarrow \mathcal{S}$ has a bifree algebra then \mathcal{D} has a unique uniform fixed-point operator. If L^2 has a bifree algebra then this fixed-point operator is dinatural.*

Although a similar result is proved in [13], here, in stating the proposition with respect to an arbitrary comonad on \mathcal{S} , we are placing the result in what we believe to be its natural general setting. The possibilities this provides are thoroughly exploited in the proofs of Proposition 6.5 and Theorem 3 below. That said, the proof of Proposition 6.1, outlined below, is essentially due to Freyd [13].

Let $s : L\Phi \longrightarrow \Phi$ be the bifree algebra for L on \mathcal{S} . The L -algebra $s : L\Phi \longrightarrow \Phi$ corresponds to an endomorphism $s : \Phi \longrightarrow \Phi$ in \mathcal{D} . The next two lemmas give the critical properties of s in \mathcal{D} .

Lemma 6.2 *For any $f : A \longrightarrow A$ there exists a unique $u_f : \Phi \longrightarrow A$ such that $f \circ Ju_f = Ju_f \circ s$.*

Lemma 6.3 *There exists a unique map $\infty : 1 \longrightarrow \Phi$ such that $s \circ \infty = \infty$.*

The proof that \mathcal{D} has a unique uniform fixed-point operator follows swiftly from Lemmas 6.2 and 6.3. The argument is carried out directly in \mathcal{D} . Define $(\cdot)^* : \mathcal{D}(A, A) \rightarrow \mathcal{D}(1, A)$ by $f^* = Ju_f \circ \infty$. Then $(\cdot)^*$ is a fixed-point operator by: $f \circ f^* = f \circ Ju_f \circ \infty = Ju_f \circ s \circ \infty = Ju_f \circ \infty = f^*$. For uniformity, suppose we have h, g with $Jh \circ f = g \circ Jh$, as in Definition 2.7. Then $J(h \circ u_f) \circ s = Jh \circ f \circ Ju_f = g \circ J(h \circ u_f)$, so $h \circ u_f = u_g$ (by the uniqueness of u_g). Thus indeed $Jh \circ f^* = Jh \circ Ju_f \circ \infty = Ju_g \circ \infty = g^*$. For uniqueness, suppose $(\cdot)^*$ is any fixed-point operator. Then $s^* = \infty$ by the uniqueness part of Lemma 6.3. If, further, $(\cdot)^*$ is uniform then, because $f \circ Ju_f = Ju_f \circ s$, we have $f^* = Ju_f \circ s^* = Ju_f \circ \infty = f^*$.

It remains to prove the dinaturality claim of Proposition 6.1. Suppose that L^2 has a bifree algebra. (By Proposition 5.2, this itself implies that L has a bifree algebra.) Again, the argument follows Freyd [13].

Lemma 6.4 *For any $f : A \longrightarrow A$ in \mathcal{D} , it holds that $f^* = (f \circ f)^*$.*

The lemma is first proved for $s : \Phi \longrightarrow \Phi$, using the fact that $s \circ Ls : L^2\Phi \longrightarrow \Phi$ is a bifree L^2 -algebra, as given by Proposition 5.2. It follows for arbitrary f by uniformity.

Dinaturality is an easy consequence of the lemma. Given $f : A \longrightarrow B$ and $g : B \longrightarrow A$, consider the map $q =$

$c \circ (f \times g) : A \times B \longrightarrow A \times B$, where c is the symmetry map for product. Then $q^* = (q \circ q)^* = \langle (g \circ f)^*, (f \circ g)^* \rangle$, where the last equality is obtained by uniformity. It follows that $q \circ \langle (g \circ f)^*, (f \circ g)^* \rangle = \langle (g \circ f)^*, (f \circ g)^* \rangle$, which implies the desired equality.

In the remainder of the section, we obtain conditions that imply that \mathcal{D} has a parametrically uniform Conway operator. The main strategy is to instantiate Proposition 6.1 by varying the comonad. This will allow us to derive all the properties of Conway operators by assuming the existence of enough bifree algebras. The first example of such an application is to obtain a parameterized fixed-point operator.

Proposition 6.5 *If every endofunctor $L(X \times (-))$ on \mathcal{S} has a bifree algebra then \mathcal{D} has a unique parametrically uniform parameterized fixed-point operator.*

The proof is by instantiating Proposition 6.1 using the endofunctors $L(X \times (-))$ as comonads. The comonad structure is most easily seen by considering the composite functor $\mathcal{S} \longrightarrow \mathcal{D} \longrightarrow \mathcal{D}_{X \times (\perp)}$, which has a left adjoint, thereby exhibiting $\mathcal{D}_{X \times (\perp)}$ as the co-Kleisli category of the composite comonad on \mathcal{S} . Indeed, by the remarks after Definitions 2.2 and 2.8, the parameterized fixed-point property, parameterized uniformity, and uniqueness all follow directly from Proposition 6.1 when interpreted in the appropriate co-Kleisli categories. It only remains to verify the naturality of the induced $(\cdot)^\dagger$ operator in its parameter.

A comonad morphism $\nu : (L_1, \varepsilon_1, \delta_1) \Rightarrow (L_2, \varepsilon_2, \delta_2)$, between two comonads on \mathcal{S} , is a natural transformation $\nu : L_1 \Rightarrow L_2$ that preserves the counit and comultiplication in the evident way (see [2, §3.6] for the dual notion of monad morphism). Any such comonad morphism induces a functor $H : \mathcal{D}_2 \rightarrow \mathcal{D}_1$ between the associated co-Kleisli categories (it is the identity on objects, and a morphism from A to B in \mathcal{D}_2 , given by $f : L_2 A \longrightarrow B$ in \mathcal{S} , gets mapped to the morphism from A to B in \mathcal{D}_1 given by $f \circ \nu_A : L_1 A \longrightarrow B$ in \mathcal{S}).

Lemma 6.6 *Suppose $\nu : (L_1, \varepsilon_1, \delta_1) \Rightarrow (L_2, \varepsilon_2, \delta_2)$ is a comonad morphism, where both L_1 and L_2 have bifree algebras in \mathcal{S} . Let \mathcal{D}_1 and \mathcal{D}_2 be the associated co-Kleisli categories, let $H : \mathcal{D}_2 \rightarrow \mathcal{D}_1$ be the ν -induced functor, and let $(\cdot)^*_1$ and $(\cdot)^*_2$ be the unique uniform fixed-point operators in \mathcal{D}_1 and \mathcal{D}_2 . Then, for any $f : A \longrightarrow A$ in \mathcal{D}_2 , it holds that $H(f^{*_2}) = (Hf)^{*_1}$.*

The proof, which uses the construction of the uniform fixed-point operators given after Lemmas 6.2 and 6.3, is routine.

To derive the naturality of $(\cdot)^\dagger$ from lemma 6.6, consider any $g : X \longrightarrow Y$ in \mathcal{D} (the co-Kleisli category of L) as in Definition 2.2. Then $(g \times (-)) : X \times (-) \Rightarrow Y \times (-)$ is a comonad morphism between comonads on \mathcal{D} , and $K(g \times J(-)) : L(X \times (-)) \Rightarrow L(Y \times (-))$ is a corresponding comonad morphism, between comonads on \mathcal{S} ,

that induces the functor $H_g : \mathcal{D}_{Y \times (\perp)} \rightarrow \mathcal{D}_{X \times (\perp)}$ between its co-Kleisli categories. Thus, by the discussion after Definition 2.2, Lemma 6.6 does indeed imply naturality.

In order to obtain that the unique parametrically uniform $(\cdot)^\dagger$ is also a Conway operator, we require yet more bifree algebras. We say that \mathcal{S} has *sufficiently many bifree algebras* if all endofunctors $L(X \times L(X \times (-)))$ and $L(X \times (-) \times (-))$ on \mathcal{S} have bifree algebras.

Theorem 3 *If \mathcal{S} has sufficiently many bifree algebras then \mathcal{D} has a unique parametrically uniform parameterized fixed-point operator, and it is a Conway operator.*

To prove the theorem, suppose that \mathcal{S} has sufficiently many bifree algebras. By Proposition 5.2, all endofunctors $L(X \times (-))$ on \mathcal{S} have bifree algebras. So by Proposition 6.5, \mathcal{D} has a unique parametrically uniform parameterized fixed-point operator. Moreover, parameterized dinaturality follows from ordinary dinaturality given by Proposition 6.1, when the comonad is instantiated to $L(X \times (-))$.

It remains to prove the diagonal property. To this end, observe that on any category \mathcal{C} with finite products, the endofunctor $(-) \times (-)$ can be endowed with the structure of a comonad; in fact this can be done in two inequivalent ways. In both cases the counit is $\pi_1 : A \times A \longrightarrow A$. The two possible comultiplications are $\langle \pi_1, \pi_2, \pi_2, \pi_1 \rangle$ and $\langle \pi_1, \pi_2, \pi_2, \pi_2 \rangle : A \times A \longrightarrow A \times A \times A \times A$. (In the first case the coalgebras of the comonad are involutions in \mathcal{C} , in the second case the coalgebras are idempotents.) Strangely, the proof below works equally well with either choice of comultiplication.

Consider the comonad $(-) \times (-)$ on the category $\mathcal{D}_{X \times (\perp)}$, and write \mathcal{D}_{d_X} for its co-Kleisli category. The chain of right adjoints $\mathcal{S} \rightarrow \mathcal{D} \rightarrow \mathcal{D}_{X \times (\perp)} \rightarrow \mathcal{D}_{d_X}$ shows that \mathcal{D}_{d_X} arises as the co-Kleisli category of a comonad with underlying functor $L(X \times (-) \times (-))$ on \mathcal{S} . Because $L(X \times (-) \times (-))$ has a bifree algebra, Proposition 6.1 relativizes to give that \mathcal{D}_{d_X} has a unique uniform fixed-point operator. In terms of \mathcal{D} this says that there exists a unique family $(\cdot)^\ddagger : \mathcal{D}(X \times A \times A, A) \rightarrow \mathcal{D}(X, A)$ satisfying:

1. For any $f : X \times A \times A \longrightarrow A$,
 $f \circ \langle \text{id}_X, f^\ddagger, f^\ddagger \rangle = f^\ddagger : X \longrightarrow A$.
2. For any $f : X \times A \times A \longrightarrow A$, $g : X \times B \times B \longrightarrow B$ and $h : A \longrightarrow B$, $h \circ f = g \circ (\text{id}_X \times Jh \times Jh)$ implies $g^\ddagger = Jh \circ f^\ddagger$.

The diagonal property is proved by showing that the two operations, mapping $f : X \times A \times A \longrightarrow A$ to $(f^\dagger)^\dagger$ and $(f \circ (\text{id}_X \times \Delta))^\dagger : X \longrightarrow A$ respectively (defined using the parametrically uniform parameterized fixed-point operator on \mathcal{D}), both satisfy the characterising properties of $(\cdot)^\ddagger$ above. Therefore the two operations are equal, hence $(f \circ (\text{id}_X \times \Delta))^\dagger = (f^\dagger)^\dagger$.

We end this section by observing that it is a simple application of Proposition 5.3 to show that the requirement of the existence of sufficiently many bifree algebras in \mathcal{S} is equivalent to requiring the existence of bifree algebras in \mathcal{D} . We write $T : \mathcal{D} \rightarrow \mathcal{D}$ for the induced monad (given by the composite JK) on \mathcal{D} .

Proposition 6.7 *\mathcal{S} has sufficiently many bifree algebras if and only if all endofunctors of the form $T(X \times T(X \times (-)))$ and $T(X \times (-) \times (-))$ on \mathcal{D} have bifree algebras.*

In spite of the above reformulation, we believe that it is usually more appropriate to consider the bifree algebras as living in \mathcal{S} . A common application of the results in this section will involve using a category \mathcal{S} that is *algebraically compact* [13, 14], in which case the existence of sufficiently many bifree algebras in \mathcal{S} is guaranteed. The canonical example of this situation is when \mathcal{S} is \mathbf{Cpo}_\perp , which is algebraically compact with respect to \mathbf{Cpo} -enriched endofunctors [10]. The results in this section thus apply to the co-Kleisli category of any comonad on \mathbf{Cpo}_\perp whose underlying functor is \mathbf{Cpo} -enriched, not just to the lifting comonad. A degenerate case is the identity comonad, showing that it is also possible for \mathcal{D} itself to be algebraically compact (although this implies that $\mathbf{1}$ is a zero object in \mathcal{D}).

7. Fixed points and lifting monads

In Section 6, we took a category of “strict” maps as basic, and derived the relevant properties of fixed points in a category of “domains” determined as the co-Kleisli category of a comonad on \mathcal{S} . In many examples, however, the category \mathcal{S} is itself obtained as the category of algebras for a “lifting” monad on a category of “predomains”. In this section, we investigate such situations in general. Surprisingly, the strong properties of a lifting monad allow all assumptions about the existence of bifree algebras to be dropped. Instead, the mere existence of uniform non-parameterized fixed-points suffices to determine a unique parametrically uniform Conway operator.

Let \mathcal{C} be a category with finite products and a strong monad (T, η, μ, t) (see e.g. [22, 23]). We write \mathcal{K} for the Kleisli category of the monad, and we write $I : \mathcal{C} \rightarrow \mathcal{K}$ for the associated (left-adjoint) functor. We assume that \mathcal{C} has *Kleisli exponentials*, i.e. that, for every X in \mathcal{C} the functor $I(X \times (-)) : \mathcal{C} \rightarrow \mathcal{K}$ has a right adjoint (see e.g. [22, 28]). These assumptions give the structure required to model Moggi’s *computational λ-calculus* [22].

We wish to consider a notion of fixed-point in \mathcal{C} suitable for adding a recursion operator to the computational λ -calculus. Because of the existence of Kleisli exponentials, it suffices to consider a non-parameterized notion.

Definition 7.1 (Uniform T -fixed-pt. op.) A *uniform T -fixed-point operator* is a family of functions

$(\cdot)^* : \mathcal{C}(TA, TA) \rightarrow \mathcal{C}(1, TA)$ such that:

1. For any $f : TA \longrightarrow TA$, $f \circ f^* = f^*$.
2. For any $f : TA \longrightarrow TA$, $g : TA \longrightarrow TA$ and $h : TA \longrightarrow TB$, if $h \circ \mu = \mu \circ Th$ and $g \circ h = h \circ f$ then $g^* = h \circ f^*$.

One familiar setting in which a (unique) uniform T -fixed-point operator exists is when \mathcal{C} has a *fixpoint object* in the sense of Crole and Pitts [5], see [24, 28]. In this paper, we take the weaker notion of uniform T -fixed-point operator as primitive. However, we shall see circumstances below in which the two notions are equivalent.

In this section, our aim is to show how T -fixed-point operators give rise to fixed-point operators as considered earlier in the paper. To this end, we write \mathcal{S} for the category of algebras of the monad T (the Eilenberg-Moore category) and L for the comonad on \mathcal{S} induced by the adjunction with \mathcal{C} . Let \mathcal{D} be the co-Kleisli category of L , and let $J : \mathcal{S} \rightarrow \mathcal{D}$ be the induced functor (as in Section 6). Concretely \mathcal{D} can be described as the category whose objects are Eilenberg-Moore algebras for T , with hom-sets: $\mathcal{D}(TA \xrightarrow{a} A, TB \xrightarrow{b} B) = \mathcal{C}(A, B)$.

Proposition 7.2 *There is a one-to-one correspondence between uniform T -fixed-point operators on \mathcal{C} and parametrically uniform parameterized fixed-point operators on \mathcal{D} .*

Our aim is to show that, under suitable conditions, there is a unique uniform T -fixed-point operator, and that the unique parametrically uniform parameterized fixed-point operator determined is a Conway operator.

One condition is that \mathcal{C} have a *parameterized natural numbers object* $1 \xrightarrow{0} \mathbf{N} \xrightarrow{s} \mathbf{N}$ (see [19, p. 71, Exercise 4] — this is the appropriate notion of natural numbers object for non-cartesian-closed categories). The other assumption is one on the monad.

Definition 7.3 (Equational lifting monad [4]) A strong monad (T, η, μ, t) is said to be an *equational lifting monad* if it is commutative and also satisfies the equation $t \circ \Delta = T\langle \eta, \text{id}_X \rangle : TX \longrightarrow T(TX \times X)$.

In [4], it is shown that equational lifting monads exactly capture the equational properties of partial map classifiers.

Theorem 4 *Suppose \mathcal{C} has a parameterized natural numbers object and T is an equational lifting monad. Then \mathcal{C} has at most one uniform T -fixed-point operator. Moreover, if such an operator exists then the associated unique parameterized fixed-point operator on \mathcal{D} is a Conway operator.*

The bulk of the work in the proof of Theorem 4 goes into proving the proposition below.

Proposition 7.4 *Under the conditions of Theorem 4, if all idempotents in \mathcal{C} split and \mathcal{C} has a uniform T -fixed-point operator then \mathcal{S} has sufficiently many bifree algebras.*

The proof is outlined in Section 8.

To derive Theorem 4 from Proposition 7.4, one first shows that all the structure on \mathcal{C} (parameterized natural number object, equational lifting monad, Kleisli exponentials) extends to its *Karoubi envelope*, $\text{Split}(\mathcal{C})$, (see [19, p. 100, Exercise 2]). Moreover, if \mathcal{C} has a uniform T -fixed-point operator then so does $\text{Split}(\mathcal{C})$. By definition, all idempotents split in $\text{Split}(\mathcal{C})$. Thus, by Proposition 7.4, $\text{Split}(\mathcal{S})$ has sufficiently many bifree algebras ($\text{Split}(\mathcal{S})$ is indeed the category of algebras for the monad on $\text{Split}(\mathcal{C})$). Hence, by Theorem 3, $\text{Split}(\mathcal{D})$ has a unique parametrically uniform parameterized fixed-point operator, and it is a Conway operator. However, it is easily shown that parametrically uniform parameterized fixed-point operators and Conway operators on \mathcal{D} and on $\text{Split}(\mathcal{D})$ are in one-to-one correspondence. Theorem 4 follows by Proposition 7.2.

One other consequence of Proposition 7.4 is that, by Proposition 5.3, the existence of a bifree L -algebra on \mathcal{S} is equivalent to the existence of a bifree T -algebra on \mathcal{C} . Freyd observed that any bifree T -algebra is a *fixpoint object* [5]. We have already mentioned that any fixpoint object determines a uniform T -fixed-point operator. Thus, in the circumstances of Proposition 7.4, the existence of a T -fixed-point operator is equivalent to that of a fixpoint object.

8. Proof of Proposition 7.4

We have a category \mathcal{C} , with finite products, parameterized natural numbers object, equational lifting monad and Kleisli exponentials, in which every idempotent splits. One consequence of idempotents splitting is that, for every Y in \mathcal{C} , the functor $F(Y \times (-)) : \mathcal{C} \rightarrow \mathcal{S}$ (where $F : \mathcal{C} \rightarrow \mathcal{S}$ is the standard “free algebra” functor) has a right adjoint $(-)^Y : \mathcal{S} \rightarrow \mathcal{C}$. Essentially this means that for any object X of \mathcal{C} that lies in the image of the forgetful $U : \mathcal{S} \rightarrow \mathcal{C}$ (we henceforth call such objects *algebra carrying*), and every object Y of \mathcal{C} , the exponential X^Y exists in \mathcal{C} (the object X^Y is constructed as a retract of the Kleisli exponential TX^Y). It also implies that \mathcal{D} is cartesian closed.

Lemma 8.1 *For every algebra-carrying X , the endofunctors $X \times (-) : \mathcal{C} \rightarrow \mathcal{C}$ and $X \times (-) \times (-) : \mathcal{C} \rightarrow \mathcal{C}$ have final coalgebras.*

Lemma 8.2 *For every object X of \mathcal{S} , the endofunctors $X \times (-) : \mathcal{S} \rightarrow \mathcal{S}$ and $X \times (-) \times (-) : \mathcal{S} \rightarrow \mathcal{S}$ have final coalgebras.*

Lemma 8.3 *L is a retract of $(L1) \times (-)$ in the category of endofunctors on \mathcal{S} .*

Lemma 8.4 All endofunctors $L(X \times L(X \times (-)))$ and $L(X \times (-) \times (-))$ have final coalgebras in \mathcal{S} .

Briefly, the final coalgebras of Lemma 8.1 both have carrier $X^{\mathbb{N}}$, which is used to encode infinite sequences and full binary trees. Lemma 8.2 follows by proving that the forgetful from the category of coalgebras for $X \times (-)$ on \mathcal{S} to the category of coalgebras for $X \times (-)$ on \mathcal{C} is monadic and so creates the terminal object (similarly for $X \times (-) \times (-)$). For Lemma 8.3, the retraction is given by the pair of morphisms $\langle T!, a \rangle : TA \longrightarrow (T\mathbf{1}) \times A$ and $(T\pi_2) \circ t' : (T\mathbf{1}) \times A \longrightarrow TA$ in \mathcal{C} (where $t' : T\mathbf{1} \times A \longrightarrow T(\mathbf{1} \times A)$ is the “costrength” of T). The verification that this pair has the required properties is the only place in which the equational lifting monad equation (Definition 7.3) is used. Finally, Lemma 8.4 follows from Lemmas 8.2, 8.3 and Proposition 5.4.

Now that final coalgebras for the desired functors have been constructed in \mathcal{S} , it remains to show that they are bifree. We achieve this by some more manœuvring between categories, using Proposition 5.3 to transfer universal properties from one place to another. In fact, we shall exploit properties of the Kleisli category \mathcal{K} .

As T is a commutative strong monad, \mathcal{K} is a symmetric monoidal category and $I : \mathcal{C} \rightarrow \mathcal{K}$ is monoidal (where cartesian product is taken as the monoidal product on \mathcal{C}). We write \otimes for the monoidal product on \mathcal{K} , and L' for the underlying functor of the comonad on \mathcal{K} induced by T .

Proposition 8.5

1. \mathcal{K} can be construed as a \mathcal{D} -enriched category.
2. \otimes and L' can be construed as \mathcal{D} -functors on \mathcal{K} .
3. For any \mathcal{D} -enriched endofunctor F on \mathcal{K} , an isomorphism $a : FA \longrightarrow A$ in \mathcal{K} is an initial F -algebra if and only if $a^{\perp 1}$ is a final F -coalgebra.

The proof is given in [28]. In outline, 1 is proved using Kleisli exponentials, and 2 is then routine. For 3, the idea is to use the uniform fixed-point operator in \mathcal{D} to establish that the property of being an initial F -algebras is equivalent to a self dual property (called *special F-invariance*), and hence equivalent to the property of being a final F -coalgebra; see Theorem 5.2 of [28].

To finish the proof of Proposition 7.4, consider, for example, the functor $L(X \times L(X \times (-))) : \mathcal{S} \rightarrow \mathcal{S}$. We write $K : \mathcal{K} \rightarrow \mathcal{S}$ for the “comparison” functor from Kleisli category to Eilenberg-Moore category. We write $H : \mathcal{S} \rightarrow \mathcal{S}$ for the composite:

$$\mathcal{S} \xrightarrow{X \times (\perp)} \mathcal{S} \xrightarrow{U} \mathcal{C} \xrightarrow{I} \mathcal{K}$$

Then $L(X \times L(X \times (-))) = KHKH : \mathcal{S} \rightarrow \mathcal{S}$. By Lemma 8.4, $KHKH$ has a final coalgebra in \mathcal{S} . Thus,

by Proposition 5.3, $KHKH$ has a final coalgebra in \mathcal{K} . By Proposition 8.5.2, $KHKH$ is \mathcal{D} -enriched. Hence, by Proposition 8.5.3, \mathcal{K} has a bifree $KHKH$ -algebra. Thus, again by Proposition 5.3, \mathcal{S} has a bifree $KHKH$ -algebra, i.e. there is a bifree algebra for $L(X \times L(X \times (-)))$. The argument for $L(X \times (-) \times (-))$ is similar. This completes the proof of Proposition 7.4.

9. Discussion

Theorem 4 has applications to an axiomatic approach to denotational semantics. The conditions on \mathcal{C} are exactly suited to modelling a call-by-value version, \mathbf{PCF}_v , of \mathbf{PCF} with product types (as considered in e.g. [33]). The monad and Kleisli exponentials interpret Moggi’s computational λ -calculus [22], which is the core of \mathbf{PCF}_v . The natural numbers object is used to interpret the arithmetic operations. Intuitively, the assumption of an equational lifting monad expresses that nontermination is the only computational effect in \mathbf{PCF}_v . We suggest that a uniform T -fixed-point operator is the natural structure for interpreting recursion. By Theorem 4, there is at most one such operator, and so the interpretation of recursion is uniquely determined. Moreover, the interpretation of recursion satisfies all desirable equational properties.

An interesting aspect of the proposed notion of model is that all ingredients in the model correspond to syntactic features of the language. Thus the free category with the identified structure corresponds to a term model constructed out of \mathbf{PCF}_v programs quotiented by the equivalence induced by the categorical structure. Then the interpretation of \mathbf{PCF}_v terms in an arbitrary model is given by the unique structure preserving functor from the free model. Thus the denotational semantics of \mathbf{PCF}_v is recast in the framework of Lawvere’s functorial semantics.

The categorical structure of the models determines a rudimentary equational logic for proving operational equalities between \mathbf{PCF}_v programs. On the one hand, this logic supports a “denotational” form of reasoning, using categorical universal properties. On the other, by interpreting the equalities in the free model, any argument has a direct “operational” reading as following a chain of equalities between \mathbf{PCF}_v programs. Thus one might argue that the notion of model provides a denotational framework for direct operational reasoning. One wonders how powerful the induced proof principles are.

Another question of power is how far our approach of deriving equational properties of recursion from categorical universal properties can be extended to derive properties of higher-order recursion. A natural syntax for higher-order recursion is given by the simply-typed λ -calculus extended with a typed fixed-point combinator. It can be shown that the desired equational theory between such terms is that in-

duced by a suitable notion of η -expanded typed Böhm trees, that this theory is co-r.e. and satisfies a characterisation as a maximally-consistent typically ambiguous theory (cf. [31] and our Theorem 2). A major open question is whether the theory is decidable. The restricted case of equalities between so-called *recursion schemas* has recently been settled in the positive by the long awaited proof of the decidability of language equivalence for DPDAAs [27]. It would be remarkable if the proof rules in Stirling’s tableau approach to decidability [32] could be derived from category-theoretic universal properties.

Another interesting (and less ambitious!) direction for research is to investigate the equational theory induced by Hasegawa’s notion of *uniform trace* [16], which generalises parametrically uniform Conway operators to symmetric monoidal categories. In particular, Hasegawa considers traced *cartesian-center* monoidal categories as models of *cyclic sharing graphs*. Perhaps there is a completeness theorem for uniform traces with respect to an equational theory induced by suitable unfoldings of such graphs.

References

- [1] H. Barendregt. *The Lambda Calculus, its Syntax and Semantics*. North Holland, Second edition, 1984.
- [2] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer-Verlag, 1985.
- [3] S. Bloom and Z. Ésik. *Iteration Theories*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1993.
- [4] A. Bucalo, C. Führmann, and A. Simpson. Equational lifting monads. *Proceedings CTCS ’99, Electronic Notes in Computer Science*, 29, 1999.
- [5] R. Crole and A. Pitts. New foundations for fixpoint computations: FIX-hyperdoctrines and the FIX-logic. *Information and Computation*, 98:171–210, 1992.
- [6] S. Eilenberg. *The Category \mathcal{C}* . Unpublished manuscript.
- [7] M. Escardó and T. Streicher. Induction and recursion on the partial real line with applications to Real PCF. *Theoretical Computer Science*, 210(1):121–157, 1999.
- [8] Z. Ésik. Independence of the equational axioms of iteration theories. *Journal of Computer and System Science*, 36:66–76, 1988.
- [9] Z. Ésik. Group axioms for iteration. *Information and Computation*, 148:131–180, 1999.
- [10] M. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Distinguished Dissertation Series, Cambridge University Press, 1996.
- [11] M. Fiore and G. Plotkin. An extension of models of axiomatic domain theory to models of synthetic domain theory. In *Proceedings of CSL 96*, pages 129–149. Springer LNCS 1258, 1997.
- [12] M. Fiore, G. Plotkin, and A. Power. Complete cuboidal sets in axiomatic domain theory. In *Proceedings of 12th Annual Symposium on Logic in Computer Science*, 1997.
- [13] P. Freyd. Algebraically complete categories. In *Category Theory, Proceedings Como 1990*, Springer LNM 1488, 1991.
- [14] P. Freyd. Remarks on algebraically compact categories. In *Applications of Categories in Computer Science*, pages 95–106. LMS Lecture Notes 177, Cambridge University Press, 1992.
- [15] C. Gunter. *Semantics of Programming Languages*. MIT Press, 1992.
- [16] M. Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. Distinguished Dissertation Series, Springer-Verlag, 1999.
- [17] A. Hurkens, M. McArthur, Y. Moschovakis, L. Moss, and G. Witney. The logic of recursive equations. *Journal of Symbolic Logic*, 63(2):451–478, 1998.
- [18] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447–468, 1996.
- [19] J. Lambek and P. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.
- [20] J. Longley and A. Simpson. A uniform account of domain theory in realizability models. *Math. Struct. in Comp. Sci.*, 7:469–505, 1997.
- [21] G. McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types*. Distinguished Dissertation Series, Springer-Verlag, 1998.
- [22] E. Moggi. Computational lambda-calculus and monads. In *Proceedings of 4th Annual Symposium on Logic in Computer Science*, pages 14–23, 1989.
- [23] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1), 1991.
- [24] P. Mulry. Strong monads, algebras and fixed points. In *Applications of Categories in Computer Science*, pages 202–216. LMS Lecture Notes 177, Cambridge University Press, 1992.
- [25] G. Plotkin. *Domains*. The “Pisa” notes. Available from <http://www.dcs.ed.ac.uk/home/gdp/>, 1983.
- [26] B. Reus and T. Streicher. General synthetic domain theory — a logical approach. *Math. Struct. in Comp. Sci.*, 9:177–223, 1999.
- [27] G. Senizergues. The equivalence problem for deterministic pushdown automata is decidable. In *Automata, Languages and Programming*. Springer LNCS 1256, 1997.
- [28] A. Simpson. *Recursive types in Kleisli categories*. Unpublished manuscript, available from <http://www.dcs.ed.ac.uk/home/als/>, 1992.
- [29] A. Simpson. A characterisation of the least-fixed-point operator by dinaturality. *Theoretical Computer Science*, 118:301–314, 1993.
- [30] A. Simpson. Computational adequacy in an elementary topos. In *Computer Science Logic, Proceedings CSL ’98*, pages 323–342. Springer LNCS 1584, 1999.
- [31] R. Statman. λ -definable functionals and $\beta\eta$ conversion. *Archiv für Math. Logik und Grund.*, 23:21–26, 1983.
- [32] C. Stirling. *Decidability of DPDA Equivalence*. To appear in *Theoretical Computer Science*, 2000. Tech. Report LFCS-99-411, University of Edinburgh.
- [33] G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, 1993.

Theoretical Informatics and Applications

Theoret. Informatics Appl. **36** (2002) 181–194

DOI: 10.1051/ita:2002009

FEEDBACK, TRACE AND FIXED-POINT SEMANTICS

P. KATIS¹, NICOLETTA SABADINI¹ AND
ROBERT F.C. WALTERS^{1, 2}

Abstract. We introduce a notion of category with feedback-with-delay, closely related to the notion of traced monoidal category, and show that the Circ construction of [15] is the free category with feedback on a symmetric monoidal category. Combining with the Int construction of Joyal *et al.* [12] we obtain a description of the free compact closed category on a symmetric monoidal category. We thus obtain a categorical analogue of the classical localization of a ring with respect to a multiplicative subset. In this context we define a notion of fixed-point semantics of a category with feedback which is seen to include a variety of classical semantics in computer science.

Mathematics Subject Classification. 68Q55, 68Q70, 18D10.

1. INTRODUCTION

There has recently been an increasing interest in algebras of systems (or behaviours) in which there is a feedback or fixed-point operation. One source of this work is Kleene’s theory of regular languages, the algebra of Conway [10], and process algebras. Another starting point is the work of Elgot [11,12] on single-sorted algebraic theories with a fixed point operation, developed by Bloom and Ésik, many of their results on the equational properties of fixed points being summarized in the book [5]. There are also the network algebras of Stefanescu [34] with the emphasis on feedback.

What has emerged recently are connections with other areas of algebra and geometry. It has become clear that the basic algebra involved is that of monoidal categories \mathbf{C} , either symmetric or braided [16]; the principal operations are tensor and composition. Our concern in this paper is with symmetric, even strict,

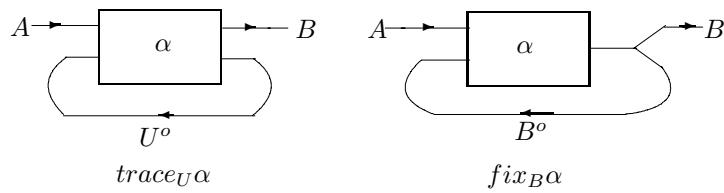
¹ Dipartimento di Scienze CC. FF.MM., Università degli Studi dell’Insubria, Como, Italy;
e-mail: nicoletta.sabadini@uninsubria.it, robert.walters@uninsubria.it

² School of Mathematics and Statistics, University of Sydney, NSW 2006, Australia.

monoidal categories – for this paper we shall refer to them simply as monoidal categories. Similarly when we speak of rings or monoids we will in each case intend commutative. Geometrically, following [27], arrows from a tensor of m objects to a tensor of n objects in a monoidal category may be represented by *circuit components* with m input wires and n output wires; then the operations become series and parallel operations of components; the symmetry is a twist of the wires. Fixed-point or feedback involves an operation which joins an input wire to an output wire of a single component – however it is not quite clear what operation to take as primitive. In Elgot–Bloom–Ésik the tensor product was taken to be a product (or dually a sum), with associated diagonal “splitting” (or codiagonal “merging”) operations on wires. For them fixed-point is an operation $fix_B : \mathbf{C}(A \times B, B) \rightarrow \mathbf{C}(A, B)$, geometrically splitting the output line B and joining the second of the B output wires to the B input wire. The fundamental paper of Joyal *et al.* [15] introduced an axiomatization of a different operation on a monoidal category which they called a *trace*. Trace is an operation $trace_U : \mathbf{C}(A \otimes U, B \otimes U) \rightarrow \mathbf{C}(A, B)$, and geometrically joins the output wire U to the input wire U . They show that a traced monoidal category \mathbf{C} may be embedded monoidally in a *compact closed* category $Int(\mathbf{C})$. Each object A in a compact closed category [23, 24], has an adjoint A° , with unit $\eta : I \rightarrow A^\circ \otimes A$ and counit $\varepsilon : A \otimes A^\circ \rightarrow I$ which may be represented geometrically as curved wires, with shapes \subset and \supset respectively. (Note: wires may be oriented, the identity on A° being a wire from right to left.) In terms of the unit and counit *fix* and *trace* may then be expressed as follows:

$$\begin{aligned} trace_U(\alpha : A \otimes U \rightarrow B \otimes U) &= (1_B \otimes \varepsilon_U)(\alpha \otimes 1_{U^\circ})(1_A \otimes \eta_{U^\circ}), \\ fix_B(\alpha : A \times B \rightarrow B) &= (1_B \times \varepsilon_B)(\Delta_B \times 1_{B^\circ})(\alpha \times 1_{B^\circ})(1_A \times \eta_{B^\circ}), \end{aligned}$$

both formulae corresponding exactly to the respective geometries:



A precise connection between the equational theory of fixed points and the notion of trace was made by Hyland and Hasegawa [14] who proved that a traced monoidal category in which the tensor is a product is the same as a Conway theory. In other language this result was already known to Bloom and Ésik [3, 4]. It is clear geometrically that if the tensor is either a product or a sum that *trace* and *fix* have the same expressivity. Further recent papers in the area include [1, 8, 32].

An important algebraic connection made by the paper of Joyal *et al.* is revealed by a special case of their construction *Int*. Notice that a monoidal category in which the only arrows are identities is, ignoring the arrows, exactly a monoid structure on the objects, whereas a compact closed category with only identity arrows is an Abelian group. The notion of traced monoidal category in this special case turns out to be cancellative monoid. Again in this special case *Int(C)* is the universal Abelian group constructed from **C**, which in the even more special case that **C** is the additive monoid of natural numbers yields *Int(C)* as the usual construction of (the additive group of) the integers.

The purpose of this paper is to show that there is more to this algebraic connection. A fundamental construction in the theory of commutative rings is the *localization of a ring* [26] relative to a multiplicative subset. Restricting to the multiplicative monoid this construction may be called the localization of monoid relative to a multiplicative subset. The construction may be made in two steps, first the construction of a monoid in which elements of the multiplicative set are cancellable (make a in the monoid equivalent to b if there is an u in the multiplicative set such that $au = bu$), followed by the *Int* construction, which provides multiplicative inverses for the elements of the multiplicative set. We will describe a categorical generalization of this process, but for simplicity restrict to the case where the multiplicative set is the whole monoid.

The categorical analogue of localization we have in mind is the universal construction of a compact closed category from a monoidal category. This construction can also be factored as a sequence of algebraic left adjoints:

$$\text{MONOIDAL} \xrightarrow{\text{Circ}} \text{FEEDBACK} \xrightarrow{Q} \text{TRACED} \xrightarrow{\text{Int}} \text{COMPACT}.$$

Here *MONOIDAL* is the category of monoidal categories and strict monoidal functors. *FEEDBACK* is the category of monoidal categories with a trace-like feedback operation which however fails to satisfy two of the axioms of trace, one of the naturality axioms of trace and the yanking axiom (the failure of these axioms is due to delay in the feedback operation). Morphisms preserve tensor and feedback. *TRACED* is the category of traced monoidal categories and trace preserving strict monoidal functors. *COMPACT* is the category of compact closed categories and functors preserving the compact closed structure. The first process in the sequence is the *Circ* construction of [18], the second step is a quotient Q killing delay, and the third is the Joyal–Street–Verity *Int* construction. Notice that applying Π_0 , the connected components functor, to a monoidal category yields a monoid, to a category with feedback or a traced monoidal category yields a cancellative monoid, and to a compact closed category yields an Abelian group. Under the effect of Π_0 , *Circ* is the free cancellative monoid on a monoid, Q is trivial, and *Int* is the free Abelian group on a cancellative monoid.

The *Circ* construction is defined as follows. From a monoidal category **C** form *Circ(C)* the monoidal category whose objects are the same as **C** and whose arrows from A to B (called *circuits*) are (isomorphism classes of) pairs (α, U) where $\alpha : A \otimes U \rightarrow B \otimes U$ is an arrow of **C**; composition of (α, U) and $(\beta, V) : B \rightarrow C$

is $(\pi_1(\beta \otimes U)\pi_2(\alpha \otimes V), U \otimes V)$ for suitable permutations π_1, π_2 . In [18, 20] the authors described a wide range of examples of the *Circ* construction, including non-deterministic automata, Mealy automata, circuits, sequential algorithms, linear systems; an example not included there is the category of systems of recursive equations in an algebra. *Circ(C)* has a “feedback” operation, which in the examples is feedback with *delay*, and introduces new state to a system. There are precise delay elements $\delta_A = (\text{symmetry}, A) : A \rightarrow A$.

The *Circ* construction has its origins in the cascade product of automata in [25], and in the seminal paper of Elgot [11]. Variations of the construction have been used later by several authors including Bloom and Ésik [4], Stefanescu [33] and Bartha [2]. This last paper [2] is the most closely related to the present work. Bartha has a notion of *strong feedback theory*, a one-sorted algebraic theory with an operation of *feedback* with delay, a notion more special than our notion of category with feedback. A precise correspondence between the axioms is difficult because general feedback properties are mixed in [2] with properties special to the case that the tensor in a theory is a sum. In most of the examples of categories with feedback, or compact closed categories, of interest to us the tensor is not a product or a sum. To give one example, he has an axiom (S5) that $\text{feedback}(1_X : X \rightarrow X) = 1_I : I \rightarrow I$; whereas in the classical example of finite dimensional spaces the trace of the identity function of a space V is the dimension of V . The principal construction of his paper is the free strong feedback theory on a one-sorted theory. As a part of this construction Bartha defines *Circ(T)* for a one-sorted theory \mathbf{T} (he calls it $\text{Aut}_{\mathbf{T}}$). The free strong feedback theory on \mathbf{T} is then $\text{Aut}_{\overline{\mathbf{T}}}/\sim$ where $\overline{\mathbf{T}}$ is a modification of \mathbf{T} and \sim is a certain congruence. The definitions of $\overline{\mathbf{T}}$ and \sim as given depend strongly on the fact that the theories are one-sorted.

Circuits have a variety of semantics. For example linear systems [20, 28] have both discrete (unit-delay) and continuous (infinitesimal-delay) semantics, and in fact such delay semantics provide an important justification for the introduction of categories with delayed feedback as a separate concept. However in this paper we will discuss fixed point (zero-delay) semantics. We define a fixed point semantics for a category with feedback \mathbf{D} to be a strong monoidal functor from \mathbf{D} to a (not-necessarily-strict monoidal) compact closed category which takes feedback to trace. In the case that \mathbf{D} is *Circ(C)* for some monoidal \mathbf{C} this amounts to a strong monoidal functor from \mathbf{C} to a compact closed category. A fixed point semantics kills the delay elements. Examples of fixed point semantics are (i) the (matrix of) languages recognized by a non-deterministic automaton, (ii) the partial function semantics of sequential programs (in the compact closed environment of **Cospan(Sets)**), (iii) the equilibrium behaviour of circuits (in compact closed **Rel**), and (iv) the set of fixed points of equations in an algebra (in compact closed **Rel** or **Span**).

Notice that we do not address the difficult question of the existence of fixed-points which has been one of the central concerns of semantics, addressed for example in the work of Tarski and of Scott, though clearly there are connections between the trace semantics and the Lefschetz fixed point theorem.

This paper is an extension of the first part of a talk [22] given by the third author at FICS 2001 (Firenze). The second part of that talk, which regarded the existence of solutions of fixed point equations in the cospan-span model of concurrency [13, 21] will be written up as a separate paper. Various results in this paper were announced in [20]. There categories with feedback were defined as algebras for a particular monad, while in this paper they are given a finite equational presentation.

2. CATEGORIES WITH FEEDBACK

As mentioned earlier, in this section of the paper *monoidal category* will mean *strict symmetric monoidal category*, and *monoidal functor* will mean *strict monoidal functor*. To simplify notation we will use the symbol π as a generic symbol for a permutation in a monoidal category, where the particular permutation is clear from the context.

Remark 2.1. The definition of category with feedback is similar to part of the definition of trace monoidal category [15]; the *yanking* axiom is lacking, and one of the naturality axioms holds only in a weak form.

Definition 2.2. A category with feedback is a monoidal category \mathbf{C} with for each triple of objects A, B, U an operation

$$fbk_U^{A,B} : \mathbf{C}(A \otimes U, B \otimes U) \rightarrow \mathbf{C}(A, B)$$

satisfying the following axioms:

(i) (naturality)

$$fbk_U((\beta : B \otimes U \rightarrow C \otimes U) \cdot ((\alpha : A \rightarrow B) \otimes 1_U)) = fbk_U(\beta) \cdot \alpha$$

(ii) (naturality)

$$fbk_U(((\gamma : C \rightarrow D) \otimes 1_U) \cdot (\beta : B \otimes U \rightarrow C \otimes U)) = \gamma \cdot fbk_U(\beta)$$

(iii) (weak naturality) If $\gamma : U \rightarrow V$ is an isomorphism in \mathbf{C} then

$$fbk_U((B \otimes \gamma) \cdot (\alpha : A \otimes U \rightarrow B \otimes U) \cdot (A \otimes \gamma^{-1})) = fbk_U(\alpha),$$

(iv) (vanishing)

$$fbk_I(\alpha : A \rightarrow B) = \alpha,$$

(v) (vanishing)

$$fbk_{U \otimes V}(\alpha : A \otimes U \otimes V \rightarrow B \otimes U \otimes V) = fbk_U(fbk_V(\alpha))$$

(vi) (superposing)

$$fbk_U(\pi \cdot ((\alpha : A \otimes U \rightarrow B \otimes U) \otimes (\beta : C \rightarrow D)) \cdot \pi) = fbk_U(\alpha) \otimes \beta.$$

Clearly any traced monoidal category, and hence any compact closed category, is a category with feedback.

Remark 2.3. The axioms also bear a close relation with axioms of Bloom and Ésik [5]. Axioms (i) and (ii) are analogues of the parameter identity. Axiom (iii) is the feedback permutation identity. Axiom (iv) is an analogue of a zero identity. Axiom (v) is an analogue of the pairing identity.

2.1. THE CIRC CONSTRUCTION

Definition 2.4. Let \mathbf{C} be a monoidal category. Then $Circ(\mathbf{C})$ is the monoidal category defined as follows: objects of $Circ(\mathbf{C})$ are those of \mathbf{C} ; arrows from A to B are isomorphism classes of pairs (α, U) where α is an arrow in \mathbf{C} from $A \otimes U$ to $B \otimes U$, and where an *isomorphism* from $(\alpha, U) : A \rightarrow B$ to $(\beta, V) : A \rightarrow B$ is an isomorphism $\gamma : U \rightarrow V$ in \mathbf{C} such that $(B \otimes \gamma)\alpha = \beta(A \otimes \gamma)$. For simplicity, we will usually work in terms of representatives rather than equivalence classes of pairs. Tensor of objects is as in \mathbf{C} . Identity, composition and tensor of arrows is as follows:

$$1_A = (1_A, I),$$

$$(\beta, V) \cdot (\alpha, U) = (\pi(\beta \otimes U)\pi(\alpha \otimes V), U \otimes V),$$

$$(\alpha, U) \otimes (\beta, V) = (\pi \cdot (\alpha \otimes \beta) \cdot \pi, U \otimes V).$$

Proposition 2.5. $Circ(\mathbf{C})$ is a category with feedback where the feedback operation is defined as follows:

$$fbk_U((\alpha, V) : A \otimes U \rightarrow B \otimes U) = (\alpha, U \otimes V).$$

Proof. Axiom(i)

$$\begin{aligned} & f bk_U(((\beta, W) : B \otimes U \rightarrow C \otimes U) \cdot (((\alpha, V) : A \rightarrow B) \otimes 1_U)) \\ &= (\pi \cdot (\beta \otimes V) \cdot \pi \cdot (\alpha \otimes U \otimes W) \cdot \pi, U \otimes V \otimes W) : A \rightarrow C \\ &\simeq f bk_U(\beta, W) \cdot (\alpha, V). \end{aligned}$$

Axiom(ii)

$$\begin{aligned} fbk_U(((\gamma, V) : C \rightarrow D) \otimes 1_U) \cdot ((\beta, W) : B \otimes U \rightarrow C \otimes U)) \\ = (\pi \cdot \gamma \otimes U \otimes W \cdot \pi \cdot \beta \otimes V, U \otimes W \otimes V) : B \rightarrow D \\ = (\gamma, V) \cdot fbk_U((\beta, W)). \end{aligned}$$

Axiom(iii)

Suppose $(\gamma, P) : U \rightarrow V$ is an isomorphism in $Circ(\mathbf{C})$ with inverse (δ, Q) . Then $(\gamma, P)(\delta, Q) = (\pi(\gamma \otimes Q)\pi(\delta \otimes P), Q \otimes P) = (1_V, I)$. Hence in \mathbf{C} the arrow $\pi \cdot (\gamma \otimes Q) : U \rightarrow V$ is the inverse of $\pi \cdot (\delta \otimes P) : V \rightarrow U$.

$$\begin{aligned} fbk_U((B \otimes (\gamma, P)) \cdot ((\alpha, W) : A \otimes U \rightarrow B \otimes U) \cdot (A \otimes (\delta, Q))) \\ = \pi(B \otimes \gamma \otimes Q \otimes W)\pi(\alpha \otimes Q \otimes P)\pi(A \otimes \delta \otimes P \otimes W)\pi \\ \cong fbk_U(\alpha). \end{aligned}$$

Axiom(iv)

$$fbk_I((\alpha, U) : A \otimes I \rightarrow B \otimes I) = (\alpha, I \otimes U) = (\alpha, U) : A \rightarrow B.$$

Axiom(v)

$$\begin{aligned} fbk_{U \otimes V}((\alpha, W) : A \otimes U \otimes V \rightarrow B \otimes U \otimes V) &= (\alpha, U \otimes V \otimes W) \\ &= fbk_U(\alpha, V \otimes W) = fbk_U(fbk_V(\alpha, W)). \end{aligned}$$

Axiom(vi)

$$\begin{aligned} fbk_U(\pi \cdot (((\alpha, V) : A \otimes U \rightarrow B \otimes U) \otimes ((\beta, W) : C \rightarrow D)) \cdot \pi) \\ = \pi \cdot (\alpha \otimes \beta) \cdot \pi = fbk_U(\alpha) \otimes \beta. \end{aligned}$$

□

2.2. UNIVERSALITY OF THE CIRC CONSTRUCTION

Notice that \mathbf{C} is contained in $Circ(\mathbf{C})$ via the functor $A \mapsto A$, $\alpha \mapsto (\alpha, I)$.

Proposition 2.6. *Given any strict monoidal functor from $F : \mathbf{C} \rightarrow \mathbf{D}$ from a monoidal category to a category with feedback, there is a unique strict monoidal*

feedback-preserving functor

$$G : \text{Circ}(\mathbf{C}) \rightarrow \mathbf{D}$$

extending F .

Proof. On objects G is defined by $G(A) = F(A)$. On arrows $G((\alpha, U) : A \rightarrow B) = fbk_{FU}(F\alpha : FA \otimes FU \rightarrow FB \otimes FU)$. Noting that $(\alpha, U) : A \rightarrow B = fbk_U(\alpha : A \otimes U \rightarrow B \otimes U, I)$ in $\text{Circ}(\mathbf{C})$, these definitions are clearly forced by the requirement that G is feedback preserving and extends F .

First note that G is well-defined on arrows. If $(\alpha, U) \simeq (\beta, V) : A \rightarrow B$ then there is an isomorphism $\gamma : U \rightarrow V$ such that $(B \otimes \gamma) \cdot \alpha = \beta \cdot (A \otimes \gamma)$ or $(B \otimes \gamma) \cdot \alpha \cdot (A \otimes \gamma^{-1}) = \beta : A \otimes V \rightarrow B \otimes V$ in \mathbf{C} , so

$$\begin{aligned} G(\beta, V) &= fbk_{FV}(F\beta) \\ &= fbk_{FV}((FB \otimes F\gamma) \cdot F\alpha \cdot (FA \otimes F\gamma^{-1})) \\ &= fbk_{FV}(F\alpha) = G(\alpha, U). \end{aligned} \quad (\text{weak naturality})$$

We need only to check that G is a feedback-preserving strict monoidal functor.

(i) G is a functor: consider $\alpha : A \otimes U \rightarrow B \otimes U$, $\beta : B \otimes V \rightarrow C \otimes V$. Then $(\beta, V)(\alpha, U) = (\pi(\beta \otimes U)\pi(\alpha \otimes V), U \otimes V)$, and hence

$$\begin{aligned} G((\beta, V)(\alpha, U)) &= G((\pi(\beta \otimes U)\pi(\alpha \otimes V), U \otimes V)) && \text{(definition)} \\ &= fbk_{FU \otimes FV}(\pi(F\beta \otimes FU)\pi(F\alpha \otimes FV)) && \text{(definition)} \\ &= fbk_{FU}(fbk_{FV}(\pi(F\beta \otimes FU)\pi(F\alpha \otimes FV))) && \text{(vanishing)} \\ &= fbk_{FU}(fbk_{FV}(\pi(F\beta \otimes FU)\pi(F\alpha \otimes FV))) && \text{(naturality)} \\ &= fbk_{FU}((fbk_{FV}(F\beta) \otimes FU)(F\alpha \otimes FV)) && \text{(superposing)} \\ &= fbk_{FV}(F\beta)fbk_{FU}(F\alpha) && \text{(naturality)} \\ &= G((\beta, V))G((\alpha, U)). && \text{(definition)} \end{aligned}$$

(ii) G is strict monoidal. Consider $\alpha : A \otimes U \rightarrow B \otimes U$, $\beta : C \otimes V \rightarrow D \otimes V$. Then

$$(\alpha, U) \otimes (\beta, V) = \pi(\alpha \otimes \beta)\pi,$$

and hence

$$\begin{aligned}
& G((\alpha, U) \otimes (\beta, V)) \\
&= G(\pi(\alpha \otimes \beta)\pi) && \text{(definition)} \\
&= fbk_{FU \otimes FV}(\pi(F\alpha \otimes F\beta)\pi) && \text{(definition)} \\
&= fbk_{FU}(fbk_{FV}(\pi(F\alpha \otimes F\beta)\pi)) && \text{(vanishing)} \\
&= fbk_{FU}(\pi f bk_{FV}(F\alpha \otimes F\beta)\pi) && \text{(naturality)} \\
&= fbk_{FU}(\pi(F\alpha \otimes f bk_{FV}(F\beta))\pi) && \text{(superposing)} \\
&= fbk_{FU}(F\alpha) \otimes f bk_{FV}(F\beta) && \text{(superposing)} \\
&= G(\alpha, U) \otimes G(\beta, V). && \text{(definition)}
\end{aligned}$$

(iii) G is feedback-preserving. Consider $\alpha : A \otimes U \otimes V \rightarrow B \otimes U \otimes V$. Then

$$fbk_U((\alpha, V) : A \otimes U \rightarrow B \otimes U) = (\alpha, U \otimes V),$$

and hence

$$\begin{aligned}
G(fbk_U((\alpha, V))) &= f bk_{FU \otimes FV}(F\alpha) && \text{(definition)} \\
&= f bk_{FU}(fbk_{FV}(F\alpha)) && \text{(vanishing)} \\
&= f bk_{FU}(G(\alpha)). && \text{(definition)}
\end{aligned}$$

□

2.3. TRACED MONOIDAL CATEGORIES FROM CATEGORIES WITH FEEDBACK

Proposition 2.7. *The free traced monoidal category on a category with feedback \mathbf{D} is the category-with-feedback quotient $Q(\mathbf{D})$ of \mathbf{D} obtained by adding the following axiom
(yanking)*

$$fbk_A(\text{symmetry} : A \otimes A \rightarrow A \otimes A) = 1_A.$$

Proof. The only axiom of traced monoidal categories missing when yanking is added to categories with feedback is

(naturality)

$$fbk_U((1_B \otimes (\beta : V \rightarrow U)) \cdot (\alpha : A \otimes U \rightarrow B \otimes V)) = f bk_V(\alpha(1_A \otimes \beta)).$$

But Lemma 2.1 of [15] shows that naturality follows from weak naturality in the presence of the other axioms together with yanking. □

It follows that the free compact closed category on a monoidal category \mathbf{C} is $\text{Int}(Q(\text{Circ}(\mathbf{C})))$.

2.4. FIXED-POINT SEMANTICS

Definition 2.8. A *fixed-point semantics* of a category with feedback \mathbf{D} is a monoidal functor to a compact closed category, which takes feedback to trace.

Notice that if \mathbf{D} is $Circ(\mathbf{C})$ then a fixed point semantics of \mathbf{D} is the same as a monoidal functor from \mathbf{C} to a compact closed category.

3. EXAMPLES

All of the examples except the last are discussed in [20], though not explicitly their fixed point semantics in the sense of this paper. We sketch briefly the examples adding details not discussed in that paper. Throughout this section we relax our restriction that monoidal categories and functors are strict; clearly although we have only proved results in the strict case the results may be suitably generalized to non-necessarily-strict monoidal categories and strong monoidal functors.

We will use two compact closed categories as codomains for semantics. One is the category **Rel** [9], with objects sets, and arrows binary relations, tensor product being product of sets, and product of relations. The adjoint of A is A ; the unit $\eta : I \rightarrow A \times A$ is the relation $\{(*, a, a) ; a \in A\}$; the counit $\varepsilon : A \times A \rightarrow I$ is $\{(a, a, *) ; a \in A\}$. The other compact closed category is **Cospan(Sets)** [19], the category with objects sets, and arrows from A to B cospans $A \xrightarrow{f} R \xleftarrow{g} B$ of functions, with composition computed using pushouts. The tensor is binary coproduct $+$, and the unit $\eta : 0 \rightarrow A + A$ is the cospan $0 \xrightarrow{!} A \xleftarrow{\nabla} A + A$; the counit $\varepsilon : A + A \rightarrow 0$ is the copspan $A + A \xrightarrow{\nabla} A \xleftarrow{!} 0$.

3.1. DETERMINISTIC MEALY AUTOMATA

Consider the monoidal category (\mathbf{Sets}, \times) . A circuit from A to B in **Sets**, with product, is a pair (α, U) where α is a function $A \times U \rightarrow B \times U$. But this is the same thing as a deterministic Mealy automaton [31] with input alphabet A , output alphabet B , and state space U (though lacking a specified initial state). Composition is essentially cascade product. One fixed point semantics is the functor $Equilibrium : Circ(\mathbf{Sets}, \times) \rightarrow \mathbf{Rel}$ defined by $Equilibrium(\alpha, U) = \{(a, b) : \exists u \text{ such that } \alpha(a, u) = (b, u)\}$. That is, $Equilibrium(\alpha, U)$ consists of those input/output pairs for which the system has an equilibrium state.

3.2. ELGOT AUTOMATA

Consider the monoidal category $(\mathbf{Sets}, +)$. A circuit from A to B in **Sets**, with direct sum, is a pair (α, U) where α is a function $A + U \rightarrow B + U$. Such were called Elgot automata in [17, 35] and model sequential algorithms. (Notice that Rutten in [29] has used the name Elgot automaton for the different notion of a partial function $A \rightarrow A + B$.) The set A may be thought of as initial states, the set B as terminal states, and U as internal states of an algorithm; then α is the

next-state function. The usual semantics for such an automaton is the partial function $\phi : A \rightharpoonup B$ computed by α in the following sense: $\phi(a)$ is defined and equal to b if there exists an n such that for $0 < i < n$, $\alpha^i(a)$ is defined and belongs to U while $\alpha^n(a) = b \in B$. (In [30] it was shown that the recursive functions are precisely the semantics of Elgot automata constructible from the predecessor and successor functions using the operations of a distributive category.)

This usual semantics is *not* however an example of the fixed point semantics of this paper since the category of partial functions is not compact closed. A slight modification however does work. We take the compact closed domain of the semantics to be **Cospan(Sets)** and then a fixed point semantics for Elgot automata is:

$$I/O : ((\alpha, U) : A \rightarrow B) \longmapsto \text{trace}_{U, \text{Cospan}}(A + U \xrightarrow{\alpha} B + U \xleftarrow{1_B + U} B + U).$$

$I/O(\alpha, U)$ may be seen, by a simple calculation, to be $A \xrightarrow{\alpha \cdot \text{inj}_A} (B + U) / \sim^{\text{inj}_B} B$ where $u_1 \sim u_2$ if $\alpha(u_1) = u_2$, and $u \sim b$ if $\alpha(u) = b$. That is, all the steps of the calculation are equated, but it is not the case that all non-terminating computations are equated. The usual partial function semantics is just the pullback of this pair of functions.

3.3. NON-DETERMINISTIC AUTOMATA

Consider the monoidal category $\text{Matr}(\wp(\Sigma^*))$ with objects natural numbers, and arrows matrices of subsets of the free monoid Σ^* . Composition is matrix composition using the fact that $\wp(\Sigma^*)$ is a semiring, with union as addition and complex concatenation of sets of words as multiplication. Tensor product is $+$. In [7] non-deterministic automata on the alphabet Σ with several initial and several terminal states were identified with certain arrows in $\text{Circ}(\text{Matr}(\wp(\Sigma^*)))$. The category $\text{Matr}(\wp(\Sigma^*))$ is itself traced monoidal and hence the identity functor induces a semantics

$$\text{Circ}(\text{Matr}(\wp(\Sigma^*))) \rightarrow \text{Matr}(\wp(\Sigma^*)) \rightarrow \text{Int}(\text{Matr}(\wp(\Sigma^*))).$$

A calculation shows that the semantics of an automaton is the matrix of languages recognized by the automaton.

3.4. SYSTEMS OF RECURSIVE EQUATIONS IN AN ALGEBRAIC THEORY

Consider an equational theory T . For concreteness let us take the theory T_{Rings} of commutative rings, though any theory, including an algebraic theory of computational processes would do. Objects are powers R^n of a symbol R ; arrows from R^m to R^n are n tuples of polynomials in the m variables x_1, x_2, \dots, x_m . The tensor product is \times . Arrows of $\text{Circ}(T_{\text{Rings}})$ are pairs $(\alpha, R^p) : R^m \times R^p \rightarrow R^n \times R^p$. Such an arrow consists of $n + p$ polynomials in $m + p$; denote the polynomials

$f_1, f_2, \dots, f_n, g_1, \dots, g_p$. There is a corresponding system of recursive polynomial equations, which has the same content, namely,

$$\begin{aligned} y_1 &= f_1(x_1, \dots, x_m, u_1, \dots, u_p) \\ y_2 &= f_2(x_1, \dots, x_m, u_1, \dots, u_p) \\ &\dots \\ y_n &= f_n(x_1, \dots, x_m, u_1, \dots, u_p) \\ u_1 &= g_1(x_1, \dots, x_m, u_1, \dots, u_p) \\ &\dots \\ u_p &= g_p(x_1, \dots, x_m, u_1, \dots, u_p). \end{aligned} \tag{*}$$

From this point of view $Circ(T_{Rings})$ is an algebra of systems of equations, composition being substitution, feedback being the operation of equating a variable on the left with one on the right.

We will give, not one semantics, but one corresponding to each model of the theory, that is, to each ring \mathbf{R} . Following Lawvere, a ring \mathbf{R} is a product-preserving functor from T_{Rings} to **Sets**. The compact closed category we take is (\mathbf{Rel}, \times) . To give a semantics is to give a monoidal functor from (T_{Rings}, \times) to (\mathbf{Rel}, \times) . We take that functor to be

$$T_{Rings} \xrightarrow{\mathbf{R}} \mathbf{Sets} \xrightarrow{\text{inclusion}} \mathbf{Rel}.$$

A calculation shows that the semantics of system $(*)$ is the subset of $\mathbf{R}^m \times \mathbf{R}^n$ consisting of $m + n$ tuples $y_1, \dots, y_n, x_1, \dots, x_m$ of elements of the ring \mathbf{R} such that there exist $u_1, \dots, u_p \in \mathbf{R}$ satisfying the system of equations $(*)$. Note that an alternative semantics in **Span(Sets)** instead of **Rel** would contain all the information of the solutions of the system, not just their existence.

Acknowledgements. The authors gratefully acknowledge stimulating conversations over many years with Stephen Bloom. We thank a referee for pointing out the work of Bartha. The research has been financially supported by the Dipartimento di Scienze Chimiche, Fisiche e Matematiche of the University of Insubria, Como, Italy, and by the Italian Progetto Cofinanziato MURST *Tipi di Ordine Superiore e Concorrenza* (TOSCA).

REFERENCES

- [1] S. Abramsky, *Retracing some paths in process algebras*, *Concur '96*, edited by U. Montanari and V. Sassone. Springer-Verlag, *Lecture Notes in Comput. Sci.* **1119** (1996) 1-17
- [2] M. Bartha, An algebraic model of synchronous systems. *Inform. and Comput.* **97** (1992) 97-131.
- [3] L. Bernatsky and Z. Ésik, Semantics of flowchart programs and the free Conway theories. *RAIRO: Theoret. Informatics Applic.* **32** (1998) 35-78.

- [4] S.L. Bloom and Z. Ésik, Axiomatising schemes and their behaviours. *J. Comput. System Sci.* **31** (1985) 375-393.
- [5] S.L. Bloom and Z. Ésik, *Iteration Theories: The Equational Logic of Iterative Processes*. Springer-Verlag, EATCS Monogr. Theoret. Comput. Sci. (1993).
- [6] S.L. Bloom and Z. Ésik, Matrix and matricial iteration theories, Part I. *J. Comput. System Sci.* **46** (1993) 381-408.
- [7] S.L. Bloom, N. Sabadini and R.F.C. Walters, Matrices, machines and behaviors. *Appl. Categorical Structures* **4** (1996) 343-360.
- [8] R.F. Blute, J.R.B. Cockett and R.A.G. Seely, Feedback for linearly distributive categories: Traces and fixpoints. *J. Pure Appl. Algebra* **154** (2000) 27-69.
- [9] A. Carboni and R.F.C. Walters, Cartesian Bicategories I. *J. Pure Appl. Algebra* **49** (1987) 11-32.
- [10] J. Conway, *Regular Algebra and Finite Machines*. Chapman and Hall, London (1971).
- [11] C.C. Elgot, *Monadic computation and iterative algebraic theories*, edited by J.C. Shepherdson. North Holland, Amsterdam, Logic Colloquium 1973, *Studies in Logic* **80** (1975).
- [12] C.C. Elgot, Matricial Theories. *J. Algebra* **42** (1976) 391-421.
- [13] F. Gadducci, U. Montanari, P. Katis, N. Sabadini and R.F.C. Walters, Comparing Cospan-spans and Tiles via a Hoare-style process calculus. TOSCA Udine, *Electron. Notes Theoret. Comput. Sci.* **62** (2001) 152-171.
- [14] M. Hasegawa, *Models of Sharing Graphs: A categorical semantics of let and letrec*, Ph.D. Thesis. Edinburgh (1997), Springer (1999).
- [15] A. Joyal, R. Street and D. Verity, Traced monoidal categories. *Math. Proc. Camb. Phil. Soc.* **119** (1996) 447-468.
- [16] A. Joyal and R. Street, Braided tensor categories. *Adv. in Math.* **102** (1993) 20-78.
- [17] W. Khalil and R.F.C. Walters, An imperative language based on distributive categories II. *RAIRO: Theoret. Informatics Appl.* **27** (1993) 503-522.
- [18] P. Katis, N. Sabadini and R.F.C. Walters, Bicategories of processes. *J. Pure Appl. Algebra* **115** (1997) 141-178.
- [19] P. Katis, N. Sabadini and R.F.C. Walters, Span(Graph): A categorical algebra of transition systems, in *Proc. Algebraic Methodology and Software Technology*. Springer-Verlag, *Lecture Notes in Comput. Sci.* **1349** (1997) 307-321.
- [20] P. Katis, N. Sabadini and R.F.C. Walters, On the algebra of systems with feedback and boundary. *Rend. Circ. Mat. Palermo (2) Suppl.* **63** (2000) 123-156.
- [21] P. Katis, N. Sabadini and R.F.C. Walters, A formalization of the IWIM Model, in *Proc. COORDINATION 2000*, edited by A. Porto and G.-C. Roman. Springer-Verlag, *Lecture Notes in Comput. Sci.* **1906** (2000) 267-283.
- [22] P. Katis, N. Sabadini and R.F.C. Walters, *Recursion and concurrency*, Invited talk, FICS 2001. Florence (2001).
- [23] P. Katis and R.F.C. Walters, The compact closed bicategory of left adjoints. *Math. Proc. Camb. Phil. Soc.* **130** (2001) 77-87.
- [24] G.M. Kelly and M. Laplaza, Coherence for compact closed categories. *J. Pure Appl. Algebra* **19** (1980) 193-213.
- [25] K. Krohn and J. Rhodes, Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Trans. Amer. Math. Soc.* **116** (1965) 450-464.
- [26] M. Nagata, *Local rings*. Interscience (1962).
- [27] R. Penrose, *Applications of negative dimensional torsors*, edited by D.J.A. Welsh. Academic Press, New York, *Comb. Math. Appl.* (1971) 221-244.
- [28] W.J. Rugh, *Linear System Theory*, Second Edition. Prentice Hall (1996).
- [29] J.J.M.M. Rutten, A calculus of transition systems (towards universal coalgebra), in *Modal Logic and Process Algebra, a bisimulation perspective*, edited by A. Ponse, M. de Rijke and Y. Venema. CSLI Publications, Standford, *CSLI Lecture Notes* **53** (1995) 231-256.
- [30] N. Sabadini, S. Vigna and R.F.C. Walters, A note on recursive functions. *Math. Struct. Comput. Sci.* **6** (1996) 127-139.

- [31] M.W. Shields, *An introduction to Automata Theory*. Blackwell Scientific Publications, Oxford (1987).
- [32] A. Simpson and G. Plotkin, Complete axioms for categorical fixed-point operators, in *Proc. 15th LICS* (2000) 30-41.
- [33] Gh. Stefanescu, On flowchart theories I: The deterministic case. *J. Comput. System Sci.* **35** (1985) 163-191.
- [34] G. Stefanescu, *Network Algebra*. Springer-Verlag (2000).
- [35] R.F.C. Walters, *Categories and Computer Science*. Carslaw Publications (1991), Cambridge University Press (1992).

Recursion from Cyclic Sharing: Traced Monoidal Categories and Models of Cyclic Lambda Calculi

Masahito Hasegawa

LFCS, Department of Computer Science, University of Edinburgh
 JCMB, King's Buildings, Edinburgh EH9 3JZ, Scotland
 Email: mhas@dcs.ed.ac.uk

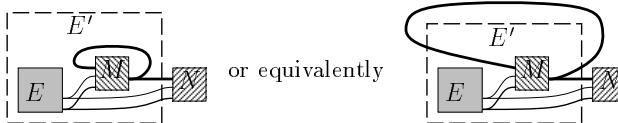
Abstract. *Cyclic sharing* (cyclic graph rewriting) has been used as a practical technique for implementing recursive computation efficiently. To capture its semantic nature, we introduce categorical models for *lambda calculi with cyclic sharing* (cyclic lambda graphs), using *notions of computation* by Moggi / Power and Robinson and *traced monoidal categories* by Joyal, Street and Verity. The former is used for representing the notion of sharing, whereas the latter for cyclic data structures. Our new models provide a semantic framework for understanding recursion created from cyclic sharing, which includes traditional models for recursion created from fixed points as special cases. Our cyclic lambda calculus serves as a uniform language for this wider range of models of recursive computation.

1 Introduction

One of the traditional methods of interpreting a recursive program in a semantic domain is to use the least fixed-point of continuous functions. However, in the real implementations of programming languages, we often use some kind of shared cyclic structure for expressing recursive environments efficiently. For instance, the following is a call-by-name operational semantics of the recursive call, in which free x may appear in M and N . We write $E \vdash M \Downarrow V$ for saying that evaluating a program M under an environment E results a value V ; in a call-by-name strategy an environment assigns a free variable to a pair consisting of an environment and a program.

$$\frac{E' \vdash N \Downarrow V \text{ where } E' = E \cup \{x \mapsto (E', M)\}}{E \vdash \text{letrec } x = M \text{ in } N \Downarrow V}$$

That is, evaluating a recursive program `letrec x = M in N` under an environment E amounts to evaluating the subprogram N under a cyclic environment E' which references itself. One may see that it is reasonable and efficient to implement the recursive (self-referential) environment E' as a cyclic data structure as below.



Also it is known that if we implement a programming language using the technique of sharing, the use of the fixed point combinator causes some unexpected duplication of resources; it is more efficient to get recursion by cycles than by the fixed point combinator in such a setting. This fact suggests that there is a gap between the traditional approach based on fixed points and cyclically created recursion.

The aim of this paper is to introduce semantic models for understanding recursive computation created by such a cyclic data structure, especially cyclic lambda graphs as studied in [AK94]. Our task is to deal with the notion of values/non-values (which provides the notion of sharing) and the notion of cycles at the semantic level. This is done by combining Moggi's *notions of computation* [Mog88] and the notion of *traced monoidal categories* recently introduced by Joyal, Street and Verity [JSV96]. The former has been used for explaining computation and values systematically, which we apply for interpreting the notion of sharing. The latter has originally been invented for analyzing cyclic structures arising from mathematics and physics, notably knot theory (e.g. [RT90]); it is then natural to use this concept for modeling cyclic graph structure. We claim that our new models are natural objects for the study of recursive computation because they unify several aspects on recursion in just one semantic framework. The leading examples are

- the *graphical (syntactical) interpretation* of recursive programs by cyclic data structures motivated as above,
- the *domain-theoretic interpretation* in which the meaning of a recursive program `letrec` $x = F(x)$ in x is given by the least fixed point $\bigcup_n F^n(\perp)$, and
- the *non-deterministic interpretation* where the program `letrec` $x = F(x)$ in x is interpreted by $\{x \mid x = F(x)\}$, the set of all possible solutions of the equation $x = F(x)$.

Each of them has its own strong tradition in computer science. However, to our knowledge, this is the first attempt to give a uniform account on these well-known, but less-related, interpretations. Moreover, our cyclic lambda calculus serves as a uniform language for them.

Construction of this paper

We recall the definition of traced monoidal categories in Section 2. In Section 3 we observe that traces and fixed point operators are closely related in two practically interesting situations - in cartesian categories, and in a special form of monoidal adjunction known as *notions of computation*. The motivating examples above are shown to be instances of our setting. Armed with these semantic observations, in Section 4 we give the models for two simply typed lambda calculi with cyclic sharing - one with unrestricted substitution, and the other with restricted substitution on values. The two settings studied in the previous section correspond to the models of these calculi respectively; the soundness and completeness results are stated. As an application, we analyze fixed point operators definable in our calculi (Section 5).

Related work

On fixed point operators. Axiomatizations of feedback operators similar to Theorem 3.1 have been given by Bloom and Ésik in [BÉ93] where they study the dual situation (categories with coproducts). Also the same authors have considered a similar axiomatization of fixed point operators in cartesian closed categories [BÉ96]. Ignoring the difference of presentations, their “Conway cartesian categories” exactly correspond to traced cartesian categories (see the remark after Theorem 3.1). Their “Conway cartesian closed categories” are then traced cartesian closed categories with an additional condition called “abstraction identity”.

On cyclic lambda calculi. Our source of cyclic lambda calculi is the version presented in [AK94]. The use of the letrec-syntax for representing cyclic sharing is not new; our presentation is inspired by a graph rewriting system in [AA95] and the call-by-need λ_{letrec} -calculus in [AF96]. In this paper we concentrate on the equational characterization of the calculi; the connection between rewriting-theoretic aspects and our work remains as an important future issue. We think the relation to operational semantics should be established in this direction, especially in the connection with the call-by-need strategy [Lau93, AF96]. Also we note that our approach is applicable not only to cyclic lambda calculi but also to general cyclic graph rewriting systems.

On action calculi. The syntax and models in this paper have arisen from the study of Milner’s *action calculi* [Mil96], a proposed framework for interactive computation. The use of notions of computation as models of *higher-order action calculi* [Mil94a] is developed in [GH96], whereas the relation between traced categories and *reflexive action calculi* [Mil94b] is studied by Mifsud [Mif96] and the author – axioms for reflexion are proved to be equivalent to those of trace. In fact, our cyclic lambda calculus can be seen as a fragment of a higher-order reflexive action calculus. A further study of action calculi in this paper’s direction will appear in the author’s forthcoming thesis (also see Example 3.10).

On Geometry of Interaction. It has been pointed out that several models of *Geometry of Interaction* [Gir89] can be regarded as traced monoidal categories (see Abramsky’s survey [Abr96]). We expect that there are potential applications of our results in this direction.

2 Traced Monoidal Categories

The notion of trace we give here for symmetric monoidal categories is adopted from the original definition of traces for balanced monoidal categories [JS93] in [JSV96].

For ease of presentation, in this section we write as if our monoidal categories are strict (i.e. monoidal products are strictly associative and coherence isomorphisms are identities).

Definition 2.1. (Traced symmetric monoidal categories [JSV96])

A symmetric monoidal category $(\mathcal{T}, \otimes, I, c)$ (where c is the symmetry; $c_{X,Y} : X \otimes Y \perp\!\!\!\rightarrow Y \otimes X$) is said to be *traced* if it is equipped with a natural family of functions, called a *trace*,

$$Tr_{A,B}^X : \mathcal{T}(A \otimes X, B \otimes X) \perp\!\!\!\rightarrow \mathcal{T}(A, B)$$

subject to the following three conditions.

– **Vanishing:**

$$Tr_{A,B}^I(f) = f : A \rightarrow B$$

where $f : A \perp\!\!\!\rightarrow B$, and

$$Tr_{A,B}^{X \otimes Y}(f) = Tr_{A,B}^X(Tr_{A \otimes X, B \otimes X}^Y(f)) : A \perp\!\!\!\rightarrow B$$

where $f : A \otimes X \otimes Y \perp\!\!\!\rightarrow B \otimes X \otimes Y$

– **Superposing:**

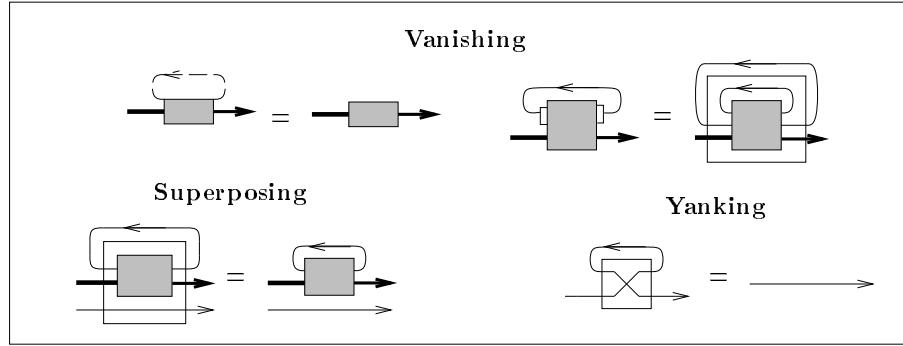
$$Tr_{C \otimes A, C \otimes B}^X(id_C \otimes f) = id_C \otimes Tr_{A,B}^X(f) : C \otimes A \perp\!\!\!\rightarrow C \otimes B$$

where $f : A \otimes X \perp\!\!\!\rightarrow B \otimes X$

– **Yanking:**

$$Tr_{X,X}^X(c_{X,X}) = id_X : X \perp\!\!\!\rightarrow X \quad \blacksquare$$

We present the graphical version of these axioms to help with the intuition of traced categories as categories with cycles (or feedback, reflexion). Such graphical languages for various monoidal categories have been developed in [JS91].



Note that naturality of a trace can be axiomatized as follows.

– Naturality in A (**Left Tightening**)

$$Tr_{A,B}^X((g \otimes id_X); f) = g; Tr_{A',B}^X(f) : A \perp\!\!\!\rightarrow B$$

where $f : A' \otimes X \perp\!\!\!\rightarrow B \otimes X$, $g : A \perp\!\!\!\rightarrow A'$

- Naturality in B (**Right Tightening**)

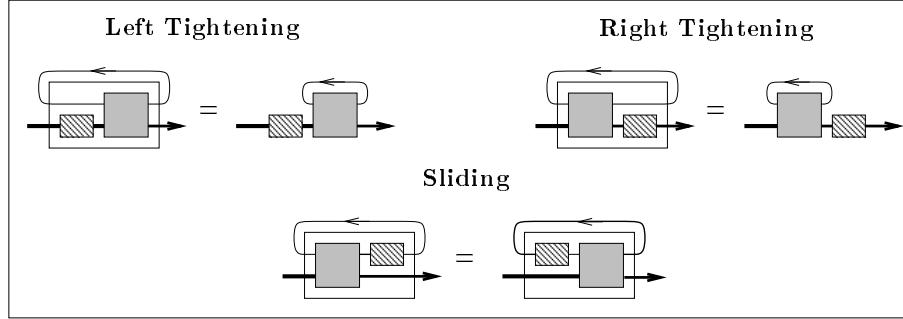
$$Tr_{A,B}^X(f; (g \otimes id_X)) = Tr_{A,B'}^X(f); g : A \perp\!\!\!\rightarrow B$$

where $f : A \otimes X \perp\!\!\!\rightarrow B' \otimes X$, $g : B' \perp\!\!\!\rightarrow B$

- Naturality in X (**Sliding**)

$$Tr_{A,B}^X(f; (id_B \otimes g)) = Tr_{A,B'}^{X'}((id_A \otimes g); f) : A \perp\!\!\!\rightarrow B$$

where $f : A \otimes X \perp\!\!\!\rightarrow B \otimes X'$, $g : X' \perp\!\!\!\rightarrow X$



Remark 2.2. The axiom **Superposing** is slightly simplified from the original version in [JSV96]

$$Tr_{A \otimes C, B \otimes D}^X((id_A \otimes c_{C,X}); (f \otimes g); (id_B \otimes c_{X,D})) = Tr_{A,B}^X(f) \otimes g$$

where $f : A \otimes X \perp\!\!\!\rightarrow B \otimes X$, $g : C \perp\!\!\!\rightarrow D$. Assuming axioms **Left** & **Right Tightenings**, ours is derivable from this original one, and vice versa. ■

Any compact closed category [KL80] is traced, for instance the category of sets and binary relations, and the category of finite dimensional vector spaces (see examples in next section). Moreover, the structure theorem in [JSV96] tells us that any traced symmetric monoidal category can be fully and faithfully embedded into a compact closed category (which can be obtained by a simple fraction construction). This fact, however, does not imply that the usage of traced categories is the same as that of compact closed categories. For the study of cyclic data structures, we find traced categories more useful than compact closed categories, as the latter seems to be too strong for our purpose.

3 Recursion from Traces

In this section we observe that traced categories can support recursive computation under certain circumstances. These results form the basis of our semantic analysis of “recursion created by cyclic structures” where we regard traced categories as the models of cyclic sharing.

3.1 Fixed Point Operators in Traced Cartesian Categories

Compact closed categories whose monoidal product is cartesian are trivial. This is not the case for traced categories. In fact, in [JSV96] it is shown that the category of sets and binary relations with its biproduct as the monoidal product is traced. Actually we find traced cartesian categories interesting in the context of semantics for recursive computation:

Theorem 3.1. *A cartesian category \mathcal{C} is traced if and only if it has a family of functions*

$$(\perp)^{\dagger_{A,X}} : \mathcal{C}(A \times X, X) \xrightarrow{\perp} \mathcal{C}(A, X)$$

(in below, parameters A, X may be omitted) such that

1. $(\perp)^\dagger$ is a parametrized fixed point operator; for $f : A \times X \xrightarrow{\perp} X$, $f^\dagger : A \xrightarrow{\perp} X$ satisfies $f^\dagger = \langle id_A, f^\dagger \rangle; f$.
2. $(\perp)^\dagger$ is natural in A ; for $f : A \times X \xrightarrow{\perp} X$ and $g : B \xrightarrow{\perp} A$, $((g \times id_X); f)^\dagger = g; f^\dagger : B \xrightarrow{\perp} X$.
3. $(\perp)^\dagger$ is natural in X ; for $f : A \times X \xrightarrow{\perp} Y$ and $g : Y \xrightarrow{\perp} X$, $(f; g)^\dagger = ((id_A \times g); f)^\dagger; g : A \xrightarrow{\perp} X$.
4. $(\perp)^\dagger$ satisfies Bekič's lemma; for $f : A \times X \times Y \xrightarrow{\perp} X$ and $g : A \times X \times Y \xrightarrow{\perp} Y$, $\langle f, g \rangle^\dagger = \langle id_A, (\langle id_{A \times X}, g^\dagger \rangle; f)^\dagger \rangle; \langle \pi'_{A,X}, g^\dagger \rangle : A \xrightarrow{\perp} X \times Y$.

Sketch of the proof: From a trace operator Tr , we define a fixed point operator $(\perp)^\dagger$ by

$$f^\dagger = Tr^X(\langle f, f \rangle) : A \xrightarrow{\perp} X$$

for $f : A \times X \xrightarrow{\perp} X$. Conversely, from a fixed point operator $(\perp)^\dagger$ we define a trace Tr by

$$Tr^X(f) = \langle id_A, (f; \pi'_{B,X})^\dagger \rangle; f; \pi_{B,X} : A \xrightarrow{\perp} B$$

(equivalently $((id_A \times \pi'_{B,X}); f)^\dagger; \pi_{B,X}$) for $f : A \times X \xrightarrow{\perp} B \times X$. We note that these constructions are mutually inverse. \square

This theorem was proved by Martin Hyland and the author independently. There are several equivalent formulations of this result. For instance, in the presence of other conditions, we can restrict 3 to the case that g is a symmetry (c.f. Lemma 1.1. of [JSV96]). For another – practically useful – example, Hyland has shown that axioms 1~4 are equivalent to 2 and

- (parametrized) dinaturality: $(\langle \pi_{A,X}, f \rangle; g)^\dagger = \langle id_A, (\langle \pi_{A,Y}, g \rangle; f)^\dagger \rangle; g : A \xrightarrow{\perp} X$ for $f : A \times X \xrightarrow{\perp} Y$ and $g : A \times Y \xrightarrow{\perp} X$
- diagonal property: $(f^\dagger)^\dagger = ((id_A \times \langle id_X, id_X \rangle); f)^\dagger$ for $f : A \times X \times X \xrightarrow{\perp} X$.

This axiomatization is the same as that of “Conway cartesian categories” in [BÉ96]. Further variations are: 2,4 with dinaturality; and 1,2,4 with the symmetric form of 4.

Perhaps the simplest example is the opposite of the category of sets and partial functions with coproduct as the monoidal product; the trace is given by the *feedback* operator which maps a partial function $f : X \rightarrow A + X$ to $f^\dagger : X \rightarrow A$, determined by iterating f until we get an answer in A if exists. Such a setting is studied in [BÉ93].

An immediate consequence of Theorem 3.1 is the close relationship between traces and the least fixed point operators in traditional domain theory.

Example 3.2. (the least fixed point operator on domains)

Consider the cartesian closed category **Dom** of Scott domains and continuous functions. The least fixed point operator satisfies conditions 1~4, thus determines a trace operator given by $Tr^X(f) = \lambda a^A.\pi(f(a, \bigcup^n(\lambda x^X.\pi'(f(a, x)))^n(\perp_X))) : A \rightarrow B$ for $f : A \times X \rightarrow B \times X$. Since the least fixed point operator is the unique dinatural fixed point operator on **Dom**, the trace above is the unique one on **Dom**. ■

The same is true for several cartesian closed categories arising from domain theory. In fact, a systematic account is possible. Simpson [Sim93] has shown that, under a mild condition, in cartesian closed full subcategories of algebraic cpos, the least fixed point operator is characterized as the unique dinatural fixed point operator. On the other hand, it is easy to see that the least fixed point operators satisfy the conditions in Theorem 3.1. Therefore, in many such categories, a trace uniquely exists and is determined by the least fixed point operator. However, we note that there are at least two traces in the category of continuous lattices, an instance which does not satisfy Simpson's condition; this category has two fixed point operators which satisfy our conditions – the least one and the greatest continuous one.

Further justification of our axiomatization of fixed point operator comes from recent work on *axiomatic domain theory* which provides a more abstract and systematic treatment of domains and covers a wider range of models of domain theory than the traditional order-theoretic approach. For this, we assume some working knowledge of this topic as found in [Sim92]. Readers who are not familiar with this topic may skip over to next subsection.

Example 3.3. (axiomatic domain theory)

Consider a cartesian closed category \mathcal{C} (category of “predomains”) equipped with a commutative monad L (the “lift”) such that the Kleisli category \mathcal{C}_L (category of predomains and “partial maps”) is *algebraically compact* [Fre91]. This setting provides a canonical fixed point operator (derived from the *fixpoint object* [CP92]) on the category of “domains” (obtained as the co-Kleisli category of the induced comonad on the Eilenberg-Moore category \mathcal{C}^L) which satisfies our axioms – Bekič’s lemma is proved from the algebraic compactness of \mathcal{C}_L [Mog95] (this idea is due to Plotkin). Thus the requirement for solving recursive domain equations (algebraic compactness) implies that the resulting category of domains is traced. ■

Regarding these facts, we believe that traces provide a good characterization of fixed point operators in traditional denotational semantics.

We conclude this subsection by observing an attractive fact which suggests how natural our trace-fixpoint correspondence is (this is rather a digression in this paper, however). Our correspondence preserves a fundamental concept on fixed point operators called *uniformity*, also known as Plotkin’s condition. This is important because fixed point operators are often canonically and uniquely characterized by this property.

Proposition 3.4. *In a traced cartesian category, the following two conditions are equivalent for any $h : X \perp\!\!\!\rightarrow Y$.*

- (Uniformity of the trace operator) For any f and g ,

$$\begin{array}{ccc} A \times X & \xrightarrow{f} & B \times X \\ \text{if } A \times h \downarrow & & \downarrow B \times h \quad \text{commutes then } Tr^X(f) = Tr^Y(g). \end{array}$$

$$A \times Y \xrightarrow{g} B \times Y$$

- (Uniformity of the fixed point operator) For any f and g ,

$$\begin{array}{ccc} A \times X & \xrightarrow{f} & X \\ \text{if } A \times h \downarrow & & \downarrow h \quad \text{commutes then } f^\dagger; h = g^\dagger. \quad \square \\ A \times Y & \xrightarrow{g} & Y \end{array}$$

In the case of domain-theoretic categories, the second condition is equivalent to saying that h is a strict map (\perp -preserving map). This fact suggests the possibility of studying the notion of strict maps and uniformity of fixed points in more general settings as in the following subsection. In particular, the first condition makes sense in any traced monoidal category.

3.2 Trace and Notions of Computation

Our observation so far says that to have an abstract trace is to have a fixed point operator in the traditional sense, provided the monoidal product is cartesian. However, regarding our motivation to model cyclic sharing, this setting is somewhat restrictive – in a cartesian category (regarded as an algebraic theory) arbitrary substitution is justified, thus there is no non-trivial notion of sharing.

To overcome this, we consider a mild generalization. Now our traced category may not be cartesian, but it is assumed to have a sub-cartesian category such that the inclusion functor preserves symmetric monoidal structure and has a right adjoint (examples will be given below). Intuitively, the sub-cartesian category is the category of “values” which can be substituted freely whereas the symmetric monoidal category part is the category of all cyclic structures which cannot be copied in general because they may contain shared resources. In this weaker setting, there still exists a fixed point operator.

Let $F : \mathcal{C} \perp\!\!\!\rightarrow \mathcal{T}$ be a faithful, identity-on-objects strict symmetric monoidal functor from a cartesian category \mathcal{C} to a traced symmetric monoidal category \mathcal{T} , with a right adjoint. Thus we identify the objects in \mathcal{C} and in \mathcal{T} , and F is identity as a function on objects. However, for readability, we write $A \times B$ and $A \otimes B$ for cartesian product in \mathcal{C} and tensor product in \mathcal{T} respectively though they are identical as F is strict symmetric monoidal. We assume a similar convention for the terminal object 1 and the unit object I .

Theorem 3.5. *Given $F : \mathcal{C} \perp\!\!\!\rightarrow \mathcal{T}$ as above, there is a family of functions*

$$(\perp)_{-}^{\dagger_{A,X}} : \mathcal{T}(A \otimes X, X) \perp\!\!\!\rightarrow \mathcal{T}(A, X)$$

such that

1. $(\perp)_{-}^{\dagger}$ is a parametrized fixed point operator in the sense that, for $f : A \otimes X \perp\!\!\!\rightarrow X$ in \mathcal{T} , $f^{\dagger} : A \perp\!\!\!\rightarrow X$ satisfies $f^{\dagger} = \Delta_A; (id_A \otimes f^{\dagger}); f$ where $\Delta_A = F(\langle id_A, id_A \rangle) : A \perp\!\!\!\rightarrow A \otimes A$.
2. $(\perp)_{-}^{\dagger}$ is natural in A in \mathcal{C} ; for $f : A \otimes X \perp\!\!\!\rightarrow X$ in \mathcal{T} and $g : B \perp\!\!\!\rightarrow A$ in \mathcal{C} , $((F(g) \otimes id_X); f)^{\dagger} = F(g); f^{\dagger} : B \perp\!\!\!\rightarrow X$.
3. $(\perp)_{-}^{\dagger}$ is natural in X in \mathcal{T} ; for $f : A \otimes X \perp\!\!\!\rightarrow Y$ in \mathcal{T} and $g : Y \perp\!\!\!\rightarrow X$ in \mathcal{T} , $(f; g)^{\dagger} = ((id_A \otimes g); f)^{\dagger}; g : A \perp\!\!\!\rightarrow X$.

Sketch of the proof: Let us write $U : \mathcal{T} \perp\!\!\!\rightarrow \mathcal{C}$ for the right adjoint of F , and $\epsilon_X : UX \perp\!\!\!\rightarrow X$ (in \mathcal{T}) for the counit. By definition, we have a natural isomorphism $(\perp)^* : \mathcal{T}(A, B) \xrightarrow{\sim} \mathcal{C}(A, UB)$. We also define $\theta_{A,X} : A \times UX \perp\!\!\!\rightarrow U(A \otimes X)$ in \mathcal{C} by $\theta_{A,X} = (id_A \otimes \epsilon_X)^*$. Now we define $(\perp)_{-}^{\dagger}$ by

$$f^{\dagger} = Tr^{UX}(F(\theta_{A,X}; UF); \Delta_{UX}); \epsilon_X : A \perp\!\!\!\rightarrow X \quad \text{in } \mathcal{T}$$

for $f : A \otimes X \perp\!\!\!\rightarrow X$ in \mathcal{T} . \square

Observe that an easier construction (c.f. Theorem 3.1) $Tr^X(f; \Delta_X) : A \perp\!\!\!\rightarrow X$ from $f : A \otimes X \perp\!\!\!\rightarrow X$ in \mathcal{T} does not work as a fixed point operator – the construction in Theorem 3.5 uses the adjunction in a crucial manner.

It is in general impossible to recover a trace operator from a fixed point operator which satisfies the conditions of Theorem 3.5; for instance, if \mathcal{T} has a zero object 0 such that $0 \otimes A \simeq 0$ (e.g. **Rel** below), the zero map satisfies these conditions. It is an interesting question to ask if we can strengthen the conditions so that we can recover a trace operator.

A careful inspection of our construction reveals that we need the trace operator just on objects of the form UX (equivalently $F(UX)$ as F is identity-on-objects); actually it is sufficient if the full subcategory of \mathcal{T} whose objects are of the form of $UX_1 \otimes \dots \otimes UX_n$ is traced. Thus such a fixed point operator exists even in a weaker setting. It would be interesting to see if this fixed point operator determines this sub-trace structure. It would be more interesting to see if there is a good connection between such a fixed point operator and fixed point operators in models of intuitionistic linear logic as studied in [Bra95].

An observation corresponding to Proposition 3.4 is as follows: for any h in \mathcal{T} , if $F(U(h))$ satisfies the uniformity condition for the trace operator then h satisfies the uniformity condition for the fixed point operator.

Note that our setting is equivalent to saying that we have a cartesian category \mathcal{C} with a monad $U \circ F$ on it, which satisfies the mono-requirement and has a commutative tensorial strength θ , such that the Kleisli category \mathcal{T} is traced. In other words, we are dealing with some *notions of computation* in the sense of Moggi [Mog88] with extra structure (trace). Our definition is inspired by a recent reformulation of notions of computation by Power and Robinson [PR96].

Definition 3.6. A *traced computational model* is a faithful, identity-on-object strict symmetric monoidal functor $F : \mathcal{C} \perp\!\!\!\rightarrow \mathcal{T}$ where \mathcal{C} is a cartesian category and \mathcal{T} a traced symmetric monoidal category, such that the functor $F(\perp) \otimes X : \mathcal{C} \perp\!\!\!\rightarrow \mathcal{T}$ has a right adjoint $X \Rightarrow (\perp) : \mathcal{T} \perp\!\!\!\rightarrow \mathcal{C}$: thus $\mathcal{T}(F(\perp) \otimes X, \perp) \simeq \mathcal{C}(\perp, X \Rightarrow \perp)$. ■

$X \Rightarrow Y$ is the so-called *Kleisli exponent*; if \mathcal{C} is cartesian closed, $X \Rightarrow Y$ is obtained as $(UY)^X$. As a traced computational model satisfies the assumption in Theorem 3.5 (a right adjoint of F is given by $I \Rightarrow (\perp)$), there is a fixed point operator in its traced category. The right adjoint $X \Rightarrow (\perp)$ can be used to interpret higher-order (higher-type) computation. Thus traced computational models have enough structure to interpret higher-order recursive computation; later we see how they can be used as the models of a simply typed lambda calculus with cyclic sharing.

To help with the intuition, we shall give a selection of traced computational models below. Most of them have already been mentioned in Section 1.

Example 3.7. (traced cartesian closed categories)

A traced cartesian closed category is a traced computational model in which the cartesian category part and the traced category part are identical. Examples include many domain-theoretic categories such as Example 3.2. ■

Example 3.8. (non-deterministic model)

The inclusion from the category **Set** of sets and functions to the category **Rel** of sets and binary relations (with the direct product of sets as the symmetric monoidal product) forms a traced computational model: $\mathbf{Rel}(A \otimes X, B) \simeq \mathbf{Set}(A, \mathbf{Rel}(X, B))$. The trace operator on **Rel**, induced by the compact closed structure of **Rel**, is given as follows: for a relation $R : A \otimes X \perp\!\!\!\rightarrow B \otimes X$, we define a relation $Tr^X(R) : A \perp\!\!\!\rightarrow B$ by $(a, b) \in Tr^X(R)$ iff $((a, x), (b, x)) \in R$ for an $x \in X$ (here a relation from A to B is given as a subobject of $A \times B$). The parametrized fixed point operator $(\perp)^\dagger$ on **Rel** is given by

$$R^\dagger = \{(a, x) \mid \exists S \subseteq X \ S = \{y \mid \exists x \in S \ ((a, x), y) \in R\} \ \& \ x \in S\} : A \longrightarrow X$$

for $R : A \otimes X \perp\!\!\!\rightarrow X$ (and R^\dagger is not the zero map!). Note that we can use an elementary topos instead of **Set**, which may provide a computationally more sophisticated model. ■

Example 3.9. (finite dimensional vector spaces over a finite field)

Let F_2 be the field with just two elements (thus its characteristic is 2), and $\mathbf{Vect}_{F_2}^{\text{fin}}$ be the category of finite dimensional vector spaces (with chosen bases) over F_2 . There is a strict symmetric monoidal functor from the category of finite sets to $\mathbf{Vect}_{F_2}^{\text{fin}}$ which maps a set S to a vector space with the basis S , and this functor has a right adjoint (the underlying functor). Since $\mathbf{Vect}_{F_2}^{\text{fin}}$ is traced (in the very classical sense), this is an instance of traced computational models. Note that this example is similar to the previous one – compare the matrix representation of binary relations and that of linear maps. ■

Example 3.10. (higher-order reflexive action calculi)

Recent work [GH96, Mif96] on action calculi [Mil96] shows that the higher-order reflexive extension of an action calculus [Mil94a, Mil94b] forms a traced computational model. In this calculus the fixed point operator $(\perp)^\dagger$ is given by

$$t^\dagger = \uparrow_{\epsilon \Rightarrow n} ((x^{\epsilon \Rightarrow n})^\Gamma (\mathbf{id}_m \otimes \langle x \rangle \cdot \mathbf{ap}_{\epsilon,n}) \cdot t^\rhd \cdot \mathbf{copy}_{\epsilon \Rightarrow n}) \cdot \mathbf{ap}_{\epsilon,n} : m \rightarrow n$$

for $t : m \otimes n \perp \rightarrow n$. Mifsud gives essentially the same operator $\mathbf{iter}_f(t)$ in his thesis [Mif96]. Using this, we can present recursion operators in various process calculi, typically the replication operator. ■

4 Semantics of Lambda Calculi with Cyclic Sharing

We introduce two simply typed lambda calculi enriched with the notion of cyclic sharing, the *simply typed λ_{letrec} -calculus* and $\lambda_{\text{letrec}}^v$ -calculus in which cyclically shared resources are represented in terms of the letrec syntax. It is shown that traced cartesian closed categories and traced computational models are sound and complete models of these calculi respectively.

4.1 The Syntax and Axioms

As the semantic observation we have seen suggests, the simply typed $\lambda_{\text{letrec}}^v$ -calculus is designed as a modification of Moggi's computational lambda calculus [Mog88]; we replace the let-syntax by the letrec-syntax which allows cyclic bindings.

In this section, we fix a set of *base types*.

Types

$$\sigma, \tau, \dots ::= b \mid \sigma \Rightarrow \tau \quad (\text{where } b \text{ is a base type})$$

Syntactic Domains

Variables	$x, y, z \dots$
Raw Terms	$M, N \dots ::= x \mid \lambda x.M \mid MN \mid \text{letrec } D \text{ in } N$
Values	$V, W \dots ::= x \mid \lambda x.M$
Declarations	$D \dots ::= x = M \mid x = M, D$

In a declaration, binding variables are assumed to be disjoint.

Typing

$$\begin{array}{c}
 \frac{}{\Gamma, x : \sigma \vdash x : \sigma} \text{ Variable} \quad \frac{\Gamma, x : \sigma, y : \sigma', \Gamma' \vdash M : \tau}{\Gamma, y : \sigma', x : \sigma, \Gamma' \vdash M : \tau} \text{ Exchange} \\
 \\
 \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x. M : \sigma \Rightarrow \tau} \text{ Abstraction} \quad \frac{\Gamma \vdash M : \sigma \Rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \text{ Application} \\
 \\
 \frac{\Gamma, x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash M_i : \sigma_i \ (i = 1, \dots, n) \quad \Gamma, x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash N : \tau}{\Gamma \vdash \text{letrec } x_1 = M_1, \dots, x_n = M_n \text{ in } N : \tau} \text{ letrec}
 \end{array}$$

Axioms

$$\begin{array}{lll}
 \text{Identity} & \text{letrec } x = M \text{ in } x & = M \quad (x \notin FV(M)) \\
 \text{Associativity} & \text{letrec } y = (\text{letrec } D_1 \text{ in } M), D_2 \text{ in } N = \text{letrec } D_1, y = M, D_2 \text{ in } N & = \text{letrec } D_1, D_2 \text{ in } M \\
 & \text{letrec } D_1 \text{ in } \text{letrec } D_2 \text{ in } M & = \text{letrec } D_2, D_1, D \text{ in } N \\
 \text{Permutation} & \text{letrec } D_1, D_2, D \text{ in } N & = \text{letrec } D_2, D_1, D \text{ in } N \\
 \text{Commutativity} & (\text{letrec } D \text{ in } M)N & = \text{letrec } D \text{ in } MN \\
 & M(\text{letrec } D \text{ in } N) & = \text{letrec } D \text{ in } MN \\
 \beta & (\lambda x. M)N & = \text{letrec } x = N \text{ in } M \\
 \sigma_v & \text{letrec } x = V, D[x] \text{ in } M & = \text{letrec } x = V, D[V] \text{ in } M \\
 & \text{letrec } x = V, D \text{ in } M[x] & = \text{letrec } x = V, D \text{ in } M[V] \\
 & \text{letrec } x = V \text{ in } M & = M \quad (x \notin FV(V) \cup FV(M)) \\
 \eta_0 & \lambda x. yx & = y
 \end{array}$$

Both sides of equations must have the same type under the same typing context; we will work just on well-typed terms. We assume the usual conventions on variables.

We remark that axioms Identity, Associativity, Permutation and Commutativity ensure that two $\lambda_{\text{letrec}}^v$ -terms are identified if they correspond to the same cyclic directed graph; thus they are a sort of structural congruence, rather than representing actual computation. β creates a sharing from a function application. σ_v describes the substitution of values (the first two for the dereference, the last one for the garbage collection). In $M[x]$ and $D[x]$, $[x]$ denotes a free occurrence of x . From β , σ_v and η_0 , we have the “call-by-value” $\beta\eta$ -equations:

Lemma 4.1. *In $\lambda_{\text{letrec}}^v$ -calculus, the following are derivable.*

$$\begin{array}{ll}
 \beta_v & (\lambda x. M)V = M\{V/x\} \\
 \eta_v & (\lambda x. Vx) = V \quad (x \notin FV(V)) \quad \square
 \end{array}$$

We think it is misleading to relate this calculus to the call-by-value operational semantics; restricting substitutions on values does not mean that this calculus is for call-by-value. Rather, our equational theory is fairly close to the *call-by-need* calculus proposed in [AF96], which corresponds to a version of lazy implementations of the call-by-name operational semantics. We expect that this connection is the right direction to relate our calculus to an operational semantics.

Also we define a “strengthened” version in which arbitrary substitution and η -reduction are allowed (thus any term is a value):

$$\begin{array}{ll} \sigma & \text{letrec } x = N, D[x] \text{ in } M = \text{letrec } x = N, D[N] \text{ in } M \\ & \text{letrec } x = N, D \text{ in } M[x] = \text{letrec } x = N, D \text{ in } M[N] \\ & \text{letrec } x = N \text{ in } M = M \quad (x \notin FV(M)) \\ \eta & \lambda x. Mx = M \quad (x \notin FV(M)) \end{array}$$

We shall call this version the *simply typed λ_{letrec} -calculus* – this corresponds to the calculus in [AK94] ignoring the typing and the extensionality (η -axiom).

4.2 Interpretation into Traced Computational Models

We just present the case of the $\lambda_{\text{letrec}}^v$ -calculus; the case of the λ_{letrec} -calculus is obtained just by replacing a traced computational model by a traced cartesian closed category.

Let us fix a traced computational model $F : \mathcal{C} \perp\!\!\!\rightarrow \mathcal{T}$, and choose an object $\llbracket b \rrbracket$ for each base type b . The interpretation of arrow types is then defined by $\llbracket \sigma \Rightarrow \tau \rrbracket = \llbracket \sigma \rrbracket \Rightarrow \llbracket \tau \rrbracket$. We interpret a $\lambda_{\text{letrec}}^v$ -term (with its typing environment) $x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash M : \tau$ to an arrow $\llbracket x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash M : \tau \rrbracket : \llbracket \sigma_1 \rrbracket \otimes \dots \otimes \llbracket \sigma_n \rrbracket \perp\!\!\!\rightarrow \llbracket \tau \rrbracket$ in \mathcal{T} as follows.

$$\begin{array}{ll} \llbracket x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash x_i : \sigma_i \rrbracket & = F(\pi_i) \quad \text{where } \pi_i \text{ is the } i\text{-th projection} \\ \llbracket \Gamma \vdash \lambda x. M : \sigma \Rightarrow \tau \rrbracket & = F(\mathbf{cur}(\llbracket \Gamma, x : \sigma \vdash M : \tau \rrbracket)) \\ \llbracket \Gamma \vdash M^{\sigma \Rightarrow \tau} N^\sigma : \tau \rrbracket & = \Delta; (\llbracket \Gamma \vdash M : \sigma \Rightarrow \tau \rrbracket \otimes \llbracket \Gamma \vdash N : \tau \rrbracket); \mathbf{ap} \\ \llbracket \Gamma \vdash \text{letrec } x_1 = M_1^{\sigma_1}, \dots, x_k = M_k^{\sigma_k} \text{ in } N : \tau \rrbracket = & \\ \Delta; (id \otimes Tr[\sigma_1] \otimes \dots \otimes [\sigma_k])(\Delta_k; (\llbracket \Gamma' \vdash M_1 : \sigma_1 \rrbracket \otimes \dots \otimes \llbracket \Gamma' \vdash M_k : \sigma_k \rrbracket); \Delta); \llbracket \Gamma' \vdash N : \tau \rrbracket & \end{array}$$

where $\mathbf{ap}_{A,B} : (A \Rightarrow B) \otimes A \perp\!\!\!\rightarrow B$ is the counit of the adjoint $F(\perp) \otimes A \dashv A \Rightarrow (\perp)$, and $\mathbf{cur} : \mathcal{T}(FA \otimes B, C) \perp\!\!\!\rightarrow \mathcal{C}(A, B \Rightarrow C)$ the associated natural bijection. In the last case, Γ' is $\Gamma, x_1 : \sigma_1, \dots, x_k : \sigma_k$ and Δ_k is the k -times copy ($\Delta_k A = F(\underbrace{id, \dots, id}_{k \text{ times}}) : A \perp\!\!\!\rightarrow \underbrace{A \otimes \dots \otimes A}_{k \text{ times}}$). Note that values are first interpreted in \mathcal{C} (following Moggi’s account, \mathcal{C} is the category of values) and then lifted to \mathcal{T} via F .

A straightforward calculation shows that traced computational models are sound for the $\lambda_{\text{letrec}}^v$ -calculus (and the same for traced cartesian closed categories and the λ_{letrec} -calculus):

Theorem 4.2. (*Soundness*)

- For any traced computational model with chosen object $\llbracket b \rrbracket$ for each base type b , this interpretation is sound; if $\Gamma \vdash M : \sigma$, $\Gamma \vdash N : \sigma$ and $M = N$ in the $\lambda_{\text{letrec}}^v$ -calculus then $\llbracket \Gamma \vdash M : \sigma \rrbracket = \llbracket \Gamma \vdash N : \sigma \rrbracket$.
- For any traced cartesian closed category with chosen object $\llbracket b \rrbracket$ for each base type b , this interpretation is sound; if $\Gamma \vdash M : \sigma$, $\Gamma \vdash N : \sigma$ and $M = N$ in the λ_{letrec} -calculus then $\llbracket \Gamma \vdash M : \sigma \rrbracket = \llbracket \Gamma \vdash N : \sigma \rrbracket$. \square

Example 4.3. (domain-theoretic model)

As we already noted, **Dom** is a traced cartesian closed category (hence also a traced computational model). The interpretation of a λ_{letrec} -term $\vdash \text{letrec } x = M \text{ in } x : \sigma$ in **Dom** is just the least fixed point $\bigcup_n F^n(\perp)$ where $F : [\![\sigma]\!] \perp \rightarrow [\![\sigma]\!]$ is the interpretation of $x : \sigma \vdash M : \sigma$. ■

Example 4.4. (non-deterministic model)

In **Rel** (Example 3.8), a $\lambda_{\text{letrec}}^v$ -term is interpreted as the set of “all possible solutions of the recursive equation”. The interpretation of $\vdash \text{letrec } x = M \text{ in } x : \sigma$ is just the set $\{x \in [\![\sigma]\!] \mid (x, x) \in [x : \sigma \vdash M : \sigma]\}$ (a subobject of $[\![\sigma]\!] = 1 \times [\![\sigma]\!]$). For instance,

$$\begin{array}{lll} \llbracket \vdash \text{letrec } x = x \text{ in } x : \sigma \rrbracket & = [\![\sigma]\!] & : 1 \rightarrow [\![\sigma]\!] \\ \llbracket \vdash \text{letrec } x = x^2 \text{ in } x : \text{nat} \rrbracket & = \{0, 1\} & : 1 \rightarrow \mathbb{N} \\ \llbracket \vdash \text{letrec } x = x + 1 \text{ in } x : \text{nat} \rrbracket & = \emptyset & : 1 \rightarrow \mathbb{N} \end{array}$$

(for the latter two cases we enrich the calculus with natural numbers). Note that this model is sound for the $\lambda_{\text{letrec}}^v$ -calculus, but not for the λ_{letrec} -calculus – since we cannot copy non-deterministic computation, this model is “resource-sensitive”. ■

Moreover, we can construct a term model (enriched with the unit and product types) to which the $\lambda_{\text{letrec}}^v$ -calculus (or λ_{letrec} -calculus) is faithfully interpreted. Actually it is possible to show that the $\lambda_{\text{letrec}}^v$ -calculus is faithfully embedded into the higher-order reflexive action calculus (Example 3.10) which is an instance of traced computational models. Thus we also have completeness:

Theorem 4.5. (*Completeness*)

- If $[\![\Gamma \vdash M : \sigma]\!] = [\![\Gamma \vdash N : \sigma]\!]$ for every traced computational model, then $M = N$ in the $\lambda_{\text{letrec}}^v$ -calculus.
- If $[\![\Gamma \vdash M : \sigma]\!] = [\![\Gamma \vdash N : \sigma]\!]$ for every traced cartesian closed category, then $M = N$ in the λ_{letrec} -calculus. □

Remark 4.6. To represent the parametrized fixed point operator given in Theorem 3.5 we have to extend the $\lambda_{\text{letrec}}^v$ -calculus with a *unit type* `unit` which has a unique value $*$:

$$\overline{\Gamma \vdash * : \text{unit}} \quad \text{Unit} \quad V = * \quad (V : \text{unit})$$

The interpretation of the unit type in a traced computational model is just the terminal object (unit object). The type constructor `unit` $\Rightarrow (\perp)$ then plays the role of the right adjoint of the inclusion from the category of values to the category of terms. We define the parametrized fixed point operator by

$$\frac{\Gamma, x : \sigma \vdash M : \sigma}{\Gamma \vdash \mu x^\sigma. M \equiv \text{letrec } f^{\text{unit} \Rightarrow \sigma} = \lambda y^{\text{unit}}. ((\lambda x^\sigma. M)(f*)) \text{ in } f * : \sigma}$$

which satisfies $\mu x. M = (\lambda x. M)(\mu x. M)$, but may not satisfy $\mu x. M = M\{\mu x. M/x\}$ in the $\lambda_{\text{letrec}}^v$ -calculus because $\mu x. M$ may not be a value in general. The operator Y_3 in the next section is essentially same as this fixed point operator, except for avoiding to use `unit`. ■

We could give the untyped version and its semantic models – by a reflexive object in a traced computational model (or a traced cartesian closed category). Regarding the results in Section 3, we can establish the connection between the dinatural diagonal fixed point operator in a model of the untyped λ_{letrec} -calculus and the trace operator of the cartesian closed category. It would be interesting to compare recursion created by untypedness and recursion created by trace (cyclic sharing) in such models.

5 Analyzing Fixed Points

In the $\lambda_{\text{letrec}}^v$ -calculus, several (weak) fixed point operators are definable – this is not surprising, because there are several known encodings of fixed point operators in terms of cyclic sharing. However, it is difficult to see that they are not identified by our equational theory – syntactic reasoning for cyclic graph structures is not an easy task, as the non-confluence result in [AK94] suggests. On the other hand, in many traditional models for recursive computation, all of them have the same denotational meaning mainly because we cannot distinguish values from non-values in such models.

One purpose of developing the traced computational models is to give a clear semantic account for these several recursive computations created from cyclic sharing. Though this topic has not yet been fully developed, we shall give some elementary analysis using the $\lambda_{\text{letrec}}^v$ -calculus and a traced computational model **(Rel)**.

We define $\lambda_{\text{letrec}}^v$ -terms $\Gamma \vdash Y_i(M) : \sigma$ ($i = 1, 2, 3$) for given term $\Gamma \vdash M : \sigma \Rightarrow \sigma$ as follows.

$$\begin{aligned} Y_1 &= \text{letrec fix}^{(\sigma \Rightarrow \sigma) \Rightarrow \sigma} = \lambda f^{\sigma \Rightarrow \sigma}. f(\text{fix } f) \text{ in fix} \\ Y_2 &= \lambda f^{\sigma \Rightarrow \sigma}. \text{letrec } x^\sigma = fx \text{ in } x \\ Y_3(M) &= \text{letrec } g^{\tau \Rightarrow \sigma} = \lambda y^\tau. M(gy) \text{ in } gN \\ &\quad (N \text{ is a closed term of type } \tau, \text{ e.g. letrec } x = x \text{ in } x : \tau) \end{aligned}$$

Each of them can be used as a fixed point operator, but their behaviours are not the same. For instance, it is known that Y_2 is more efficient than others, under the call-by-need evaluation strategy [Lau93]. Y_1 satisfies the fixed point equation $YV = V(YV)$ for any value $V : \sigma \Rightarrow \sigma$.

$$\begin{aligned} Y_1M &= \text{letrec fix} = \lambda f. f(\text{fix } f) \text{ in fix } M && \text{Commutativity} \\ &= \text{letrec fix} = \lambda f. f(\text{fix } f) \text{ in } (\lambda f. f(\text{fix } f))M && \sigma_v \\ &= \text{letrec fix} = \lambda f. f(\text{fix } f) \text{ in letrec } f' = M \text{ in } f'(\text{fix } f') && \beta \\ &= \text{letrec } f' = M \text{ in letrec fix} = \lambda f. f(\text{fix } f) \text{ in } f'(\text{fix } f') && \text{Associativity, Permutation} \\ &= \text{letrec } f' = M \text{ in } f'((\text{letrec fix} = \lambda f. f(\text{fix } f) \text{ in fix } f')f') && \text{Commutativity} \\ &= \text{letrec } f' = M \text{ in } f'(Y_1f') \\ &= M(Y_1M) \quad \text{if } M \text{ is a value} \end{aligned}$$

Y_2 satisfies $Y_2M = M(Y_2M)$ only when Mx is equal to a value (hence M is

supposed to be a higher-order value). If $M = \lambda y.V$ for some value V ,

$$\begin{aligned}
Y_2 M &= \text{letrec } x = (\lambda y.V)x \text{ in } x & \beta_v \\
&= \text{letrec } x = V\{x/y\} \text{ in } x & \beta_v \\
&= \text{letrec } x = V\{x/y\} \text{ in } V\{x/y\} & \sigma_v \\
&= \text{letrec } x = (\lambda y.V)x \text{ in } (\lambda y.V)x & \beta_v \\
&= (\lambda y.V)(\text{letrec } x = (\lambda y.V)x \text{ in } x) \text{ Commutativity} \\
&= M(Y_2 M)
\end{aligned}$$

Y_3 satisfies $Y_3(M) = M(Y_3(M))$ for any term $M : \sigma \Rightarrow \sigma$ (thus is a “true” fixed point operator).

$$\begin{aligned}
Y_3(M) &= \text{letrec } g = \lambda y.M(gy) \text{ in } (\lambda y.M(gy))N & \sigma_v \\
&= \text{letrec } g = \lambda y.M(gy) \text{ in } \text{letrec } y' = N \text{ in } M(gy') & \beta \\
&= \text{letrec } g = \lambda y.M(gy) \text{ in } M(g(\text{letrec } y' = N \text{ in } y')) \text{ Commutativity} \\
&= M(\text{letrec } g = \lambda y.M(gy) \text{ in } g(\text{letrec } y' = N \text{ in } y')) \text{ Commutativity} \\
&= M(\text{letrec } g = \lambda y.M(gy) \text{ in } gN) & \text{Identity} \\
&= M(Y_3(M))
\end{aligned}$$

The interpretation of these operators in a traced computational model is as follows.

$$\begin{aligned}
\llbracket \vdash Y_1 \rrbracket &= Tr^{(A \Rightarrow A) \Rightarrow A}(F(\mathbf{cur}((id \otimes \Delta); (\mathbf{ap} \otimes id); c; \mathbf{ap})); \Delta) \\
\llbracket \vdash Y_2 \rrbracket &= F(\mathbf{cur}(Tr^A(\mathbf{ap}); \Delta)) \\
\llbracket \Gamma \vdash Y_3(M) \rrbracket &= (Tr^{B \Rightarrow A}(F(\mathbf{cur}((\llbracket \Gamma \vdash M : \sigma \Rightarrow \sigma \rrbracket \otimes \mathbf{ap}); \mathbf{ap})); \Delta) \otimes \llbracket \vdash N : \tau \rrbracket); \mathbf{ap}
\end{aligned}$$

where $A = \llbracket \sigma \rrbracket$ and $B = \llbracket \tau \rrbracket$. They have the different interpretations in **Rel**, hence are not identified in the $\lambda_{\text{letrec}}^v$ -calculus. Assume that $S = \llbracket \vdash M : \sigma \Rightarrow \sigma \rrbracket \subseteq \mathbf{Rel}(A, A)$. Then

$$\llbracket \vdash Y_1(M) : \sigma \rrbracket = \bigcup_{f \in S} \bigcup_{(A'; f) = A' \subseteq A} A' \quad \llbracket \vdash Y_2(M) : \sigma \rrbracket = \bigcup_{f \in S} \{x \mid (x, x) \in f\}$$

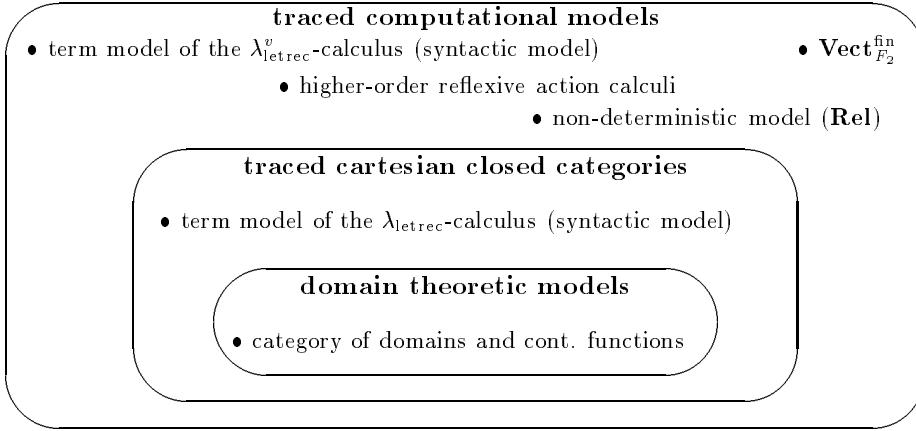
whereas

$$\llbracket \vdash Y_3(M) : \sigma \rrbracket = \bigcup_{(A'; \bigcup S) = A' \subseteq A} A'$$

(In the definition of Y_3 , we take $N : \tau$ as $\text{letrec } x = x \text{ in } x : \tau$.)

6 Conclusion

We have presented new semantic models for interpreting cyclic sharing in terms of traced monoidal categories and notions of computation, and shown the connections with cyclic lambda calculi and with traditional semantics for recursive computation. We have also demonstrated that our framework covers a wider range of models of recursion than the traditional approach. We summarize this situation, together with examples in this paper, in the diagram below.



Acknowledgements

I am deeply grateful to Martin Hyland and John Power for helpful discussions, suggestions and encouragement. I also thank Philippa Gardner, Alex Mifsud, Marcelo Fiore, Alex Simpson and Gordon Plotkin for their comments and encouragement.

References

- Abr96. S. Abramsky, Retracing some paths in process algebra. In *Proc. 7th Int. Conf. Concurrency Theory (CONCUR'96)*, Springer LNCS 1119, pages 1-17, 1996.
- AA95. Z. Ariola and Arvind, Properties of a first-order functional language with sharing. *Theoretical Computer Science* 146, pages 69-108, 1995.
- AF96. Z. Ariola and M. Felleisen, A call-by-need lambda calculus. Technical report CIS-TR-96-97, 1996. To appear in *Journal of Functional Programming*.
- AK94. Z. Ariola and J. Klop, Cyclic lambda graph rewriting. In *Proc. 9th Symposium on Logic in Computer Science (LICS'94)*, pages 416-425, 1994.
- BÉ93. S. L. Bloom and Z. Ésik, *Iteration Theories*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1993.
- BÉ96. S. L. Bloom and Z. Ésik, Fixed point operators on ccc's. Part I. *Theoretical Computer Science* 155, pages 1-38, 1996.
- Bra95. T. Braüner, The Girard translation extended with recursion. In *Proc. Computer Science Logic 1994 (CSL'94)*, Springer LNCS 933, pages 31-45, 1995.
- CP92. R. L. Crole and A. M. Pitts, New foundations for fixpoint computations: Fix hyperdoctrines and the fix logic. *Information and Computation* 98, pages 171-210, 1992.
- Fre91. P. Freyd, Algebraically complete categories. In *Proc. 1990 Como Category Theory Conference*, Springer LNM 1144, pages 95-104, 1991.
- GH96. P. Gardner and M. Hasegawa, On higher-order action calculi and notions of computation. Draft, LFCS, University of Edinburgh, 1996.
- Gir89. J. -Y. Girard, Geometry of interaction I: interpretation of system F. In *Logic Colloquium '88*, pages 221-260, North-Holland, 1989.

- JS91. A. Joyal and R. Street, The geometry of tensor calculus I. *Advances in Mathematics* 88, pages 55-113, 1991.
- JS93. A. Joyal and R. Street, Braided tensor categories. *Advances in Mathematics* 102, pages 20-78, 1993.
- JSV96. A. Joyal, R. Street and D. Verity, Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society* 119(3), pages 447-468, 1996.
- KL80. M. Kelly and M. L. Laplaza, Coherence for compact closed categories. *Journal of Pure and Applied Algebra* 19, pages 193-213, 1980.
- Lau93. J. Launchbury, A natural semantics for lazy evaluation. In *Proc. 21st ACM Symp. Principles of Programming Languages (POPL'93)*, pages 144-154, 1993.
- Mif96. A. Mifsud, *Control structures*. PhD thesis, LFCS, University of Edinburgh, 1996.
- Mil94a. R. Milner, Higher-order action calculi. In *Proc. Computer Science Logic 1992 (CSL'92)*, Springer LNCS 832, pages 238-260, 1994.
- Mil94b. R. Milner, Action calculi V: reflexive molecular forms (with Appendix by O. Jensen). Third draft, July 1994.
- Mil96. R. Milner, Calculi for interaction. *Acta Informatica* 33(8), pages 707-737, 1996.
- Mog88. E. Moggi, Computational lambda-calculus and monads. Technical report ECS-LFCS-88-66, LFCS, University of Edinburgh, 1988.
- Mog95. E. Moggi, Metalanguages and applications. Draft, 1995.
- PR96. A. J. Power and E. P. Robinson, Premonoidal categories and notions of computation. 1996. To appear in *Mathematical Structures in Computer Science*.
- RT90. N. Yu. Reshetikhin and V. G. Turaev, Ribbon graphs and their invariants derived from quantum groups. *Communications in Mathematical Physics* 127, pages 1-26, 1990.
- Sim92. A. Simpson, Recursive types in Kleisli categories. Manuscript, LFCS, University of Edinburgh, 1992.
- Sim93. A. Simpson, A characterisation of the least-fixed-point operator by dinaturality. *Theoretical Computer Science* 118, pages 301-314, 1993.

Dependent Inductive and Coinductive Types are Fibrational Dialgebras

Henning Basold

Radboud University, iCIS, Intelligent Systems

CWI, Amsterdam, The Netherlands

h.basold@cs.ru.nl

In this paper, I establish the categorical structure necessary to interpret dependent inductive and coinductive types. It is well-known that dependent type theories à la Martin-Löf can be interpreted using fibrations. Modern theorem provers, however, are based on more sophisticated type systems that allow the definition of powerful inductive dependent types (known as inductive families) and, somewhat limited, coinductive dependent types. I define a class of functors on fibrations and show how data type definitions correspond to initial and final dialgebras for these functors. This description is also a proposal of how coinductive types should be treated in type theories, as they appear here simply as dual of inductive types. Finally, I show how dependent data types correspond to algebras and coalgebras, and give the correspondence to dependent polynomial functors.

1 Introduction

It is a well-established fact that the semantics of inductive data types without term dependencies can be given by initial algebras, whereas the semantics of coinductive types can be given by final coalgebras. However, for types that depend on terms, the situation is not as clear-cut.

Partial answers for inductive types can be found in [3, 8, 9, 11, 14, 19, 20], where semantics have been given for inductive types through polynomial functors in the category of set families or in locally Cartesian closed categories. Similarly, semantics for non-dependent coinductive types have been given in [1, 2, 6] by using polynomial functors on locally Cartesian closed categories. Finally, an interpretation for Martin-Löf type theory (without recursive type definitions) has been given in [21] and corrected in [16].

So far, we are, however, lacking a full picture of dependent coinductive types that arise as duals of dependent inductive types. To actually get such a picture, I extend in the present work Hagino's idea [13], of using dialgebras to describe data types, to dependent types. This emphasises the actual structure behind (co)inductive types as their are used in systems like Agda.¹ Moreover, dialgebras allow for a direct interpretation of types in this categorical setup, without going through translations into, for example, polynomial functors.

Having defined the structures we need to interpret dependent data types, it is natural to ask whether this structure is actually sensible. The idea, pursued here, is that we want to obtain initial and final dialgebras from initial algebras and final coalgebras for polynomial functors. This is achieved by showing that the dialgebras in this work correspond to algebras and coalgebras, and that their fixed points can be constructed from fixed points of polynomial functors (in the sense of [12]).

¹It should be noted that, for example, Coq treats coinductive types differently. In fact, the route taken in Agda with copatterns and in this work is much better behaved.

To summarise, this paper makes the following contributions. First, we get a precise description of the categorical structure necessary to interpret inductive and coinductive data types, which can be seen as categorical semantics for an extension of the inductive and (copattern-based) coinductive types of Agda. The second contribution is a reduction to fixed points of polynomial functors.

What has been left out, because of space constraints, is an analysis of the structures needed to obtain induction and coinduction principles. Moreover, to be able to get a sound interpretation, with respect to type equality of dependent types, we need to require a Beck-Chevalley condition. This condition can be formulated for general (co)inductive types, but is also not given here.

Related work As already mentioned, there is an enormous body of work on obtaining semantics for (dependent) inductive, and to some extent, coinductive types, see [3, 11, 14, 20]. In the present work, we will mostly draw from [2] and [12]. Categorical semantics for basic Martin-Löf type theory have been developed, for example, in [16]. An interpretation, closer to the present work, is given in terms of fibrations by Jacobs [17]. In the first part of the paper, we develop everything on rather arbitrary fibrations, which makes the involved structure more apparent. Only in the second part, where we reduce data types to polynomial functors, we will work with slice categories, since most of the work on polynomial functors in that setting [2, 12]. Last, but not least, the starting idea of this paper is of course inspired by the dialgebras of Hagino [13]. These have also been applied to give semantics to induction-induction [4] schemes.

Outline The rest of the paper is structured as follows. In Section 2, we analyse a typical example of a dependent inductive type, namely vectors, that is, lists indexed by their length. We develop from this example a description of inductive and coinductive dependent data types in terms of dialgebras in fibrations. This leads to the requirements on a fibration, given in Section 3, that allow the interpretation of data types. In the same section, we show how dependent and fibre-wise (co)products arise canonically in such a structure, and we give an example of a coinductive type (partial streams) that can only be treated in Agda through a cumbersome encoding. The reduction of dependent data types to polynomial functors is carried out in Section 4, and finish with concluding remarks in Section 5.

Acknowledgement I would like to thank the anonymous reviewers, who gave very valuable feedback and pointed me to some more literature.

2 Fibrations and Dependent Data Types

In this section we introduce *dependent data types* as initial and final dialgebras of certain functors on fibres of fibrations. We go through this setup step by step.

Let us start with dialgebras and their homomorphisms.

Definition 2.1. Let \mathbf{C} and \mathbf{D} be categories and $F, G : \mathbf{C} \rightarrow \mathbf{D}$ functors. An (F, G) -*dialgebra* is a morphism $c : FA \rightarrow GA$ in \mathbf{D} , where A is an object in \mathbf{C} . Given dialgebras $c : FA \rightarrow GA$ and $d : FB \rightarrow GB$, a morphism $h : A \rightarrow B$ is said to be a (dialgebra) *homomorphism* from c to d , if $Gh \circ c = d \circ Fh$. This allows us to form a category $\text{DiAlg}(F, G)$, in which objects are pairs (A, c) with $A \in \mathbf{C}$ and $c : FA \rightarrow GA$, and morphisms are dialgebra homomorphisms.

The following example shows that dialgebras arise naturally from data types.

Example 2.2. Let A be a set, we denote by A^n the n -fold product of A , that is, lists of length n . Vectors over A are given by the set family $\text{Vec } A = \{A^n\}_{n \in \mathbb{N}}$, which is an object in the category $\mathbf{Set}^{\mathbb{N}}$ of families

indexed by \mathbb{N} . In general, this category is given for a set I by

$$\mathbf{Set}^I = \begin{cases} \text{objects} & X = \{X_i\}_{i \in I} \\ \text{morphisms} & f = \{f_i : X_i \rightarrow Y_i\}_{i \in I} \end{cases}.$$

Vectors come with two constructors: $\text{nil} : \mathbf{1} \rightarrow A^0$ for the empty vector and prefixing $\text{cons}_n : A \times A^n \rightarrow A^{n+1}$ of vectors with elements of A . We note that $\text{nil} : \{\mathbf{1}\} \rightarrow \{A^0\}$ is a morphism in the category \mathbf{Set}^1 of families indexed by the one-element set $\mathbf{1}$, whereas $\text{cons} = \{\text{cons}_n\} : \{A \times A^n\}_{n \in \mathbb{N}} \rightarrow \{A^{n+1}\}_{n \in \mathbb{N}}$ is a morphism in $\mathbf{Set}^{\mathbb{N}}$.

Let $F, G : \mathbf{Set}^{\mathbb{N}} \rightarrow \mathbf{Set}^1 \times \mathbf{Set}^{\mathbb{N}}$ be the functors into the product of \mathbf{Set}^1 and $\mathbf{Set}^{\mathbb{N}}$ with

$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n \in \mathbb{N}}) \quad G(X) = (\{X_0\}, \{X_{n+1}\}_{n \in \mathbb{N}}).$$

Using these, we find that $(\text{nil}, \text{cons}) : F(\text{Vec } A) \rightarrow G(\text{Vec } A)$ is an (F, G) -dialgebra, in fact, it is the *initial* (F, G) -dialgebra.

Definition 2.3. An (F, G) -dialgebra $c : FA \rightarrow GA$ is called *initial*, if for every (F, G) -dialgebra $d : FB \rightarrow GB$ there is a unique homomorphism h from c to d , the *inductive extension* of d . Dually, (A, c) is *final*, provided there is a unique homomorphism h from any other dialgebra (B, d) into c . Here, h is the *coinductive extension* of d .

Having found the algebraic structure underlying vectors, we continue by exploring how we can handle the change of indices in the constructors. It turns out that this is most conveniently done by using fibrations.

Definition 2.4. Let $P : \mathbf{E} \rightarrow \mathbf{B}$ be a functor, where the \mathbf{E} is called the *total* category and \mathbf{B} the *base* category. A morphism $f : A \rightarrow B$ in \mathbf{E} is said to be *cartesian over* $u : I \rightarrow J$, provided that i) $Pf = u$, and ii) for all $g : C \rightarrow B$ in \mathbf{E} and $v : PC \rightarrow I$ with $Pg = u \circ v$ there is a unique $h : C \rightarrow A$ such that $f \circ h = g$. For P to be a *fibration*, we require that for every $B \in \mathbf{E}$ and $u : I \rightarrow PB$ in \mathbf{B} , there is a cartesian morphism $f : A \rightarrow B$ over u . Finally, a fibration is *cloven*, if it comes with a unique choice for A and f , in which case we denote A by $u^* B$ and f by $\bar{u} B$, as displayed in the diagram on the right.

At first sight, this definition is arguably intimidating to someone who has never been exposed to fibrations. The idea is that the base category \mathbf{B} contains as objects the indices of objects in \mathbf{E} , and as morphisms substitutions. The result of carrying out a substitution on indices, is captured by the Cartesian lifting property. Let us illustrate this on set families. We define $\text{Fam}(\mathbf{Set})$ to be the category

$$\text{Fam}(\mathbf{Set}) = \begin{cases} \text{objects} & (I, X : I \rightarrow \mathbf{Set}), I \text{ a set} \\ \text{morphisms} & (u, f) : (I, X) \rightarrow (J, Y) \text{ with } u : I \rightarrow J \text{ and } \{f_i : X_i \rightarrow Y_{u(i)}\}_{i \in I} \end{cases}$$

in which composition is defined by

$$(v, g) \circ (u, f) = \left(v \circ u, \{X_i \xrightarrow{f_i} Y_{u(i)} \xrightarrow{g_{u(i)}} Z_{v(u(i))}\}_{i \in I} \right).$$

A concrete object is the pair $(\mathbb{N}, \text{Vec } A)$, where $\text{Vec } A$ is the family of vectors from Ex. 2.2.

We define a cloven fibration on set families. Let $P : \text{Fam}(\mathbf{Set}) \rightarrow \mathbf{Set}$ be the projection on the first component, that is, $P(I, X) = I$ and $P(u, f) = u$. For a family (J, Y) and a function $u : I \rightarrow J$, we define

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ \dashv \downarrow h & \nearrow u^* B & \xrightarrow{\bar{u} B} \\ PC & \xrightarrow{Pg} & B \\ \dashv \downarrow v & \nearrow I & \xrightarrow{u} \\ I & \xrightarrow{u} & PB \end{array} \quad \begin{array}{c} \mathbf{E} \\ \downarrow P \\ \mathbf{B} \end{array}$$

$u^* Y = \{Y_{u(i)}\}_{i \in I}$ and $\bar{u} Y = (u, \{\text{id} : Y_{u(i)} \rightarrow Y_{u(i)}\}_{i \in I})$. Then, for each $(w, g) : (K, Z) \rightarrow (J, Y)$ and $v : K \rightarrow I$ with $w = u \circ v$, we can define the morphism $(K, Z) \rightarrow (I, u^* Y)$ to be (v, h) with $h_k : Z_k \rightarrow Y_{u(v(k))}$ and $h_k = g_k$, since $u(v(k)) = w(k)$.

An important concept is the *fibre above* an object $I \in \mathbf{B}$, given by the category

$$\mathbf{P}_I = \begin{cases} \text{objects} & A \in \mathbf{E} \text{ with } P(A) = I \\ \text{morphisms} & f : A \rightarrow B \text{ with } P(f) = \text{id}_I \end{cases}.$$

In a cloven fibration, we can use the Cartesian lifting to define for each $u : I \rightarrow J$ in \mathbf{B} a functor $u^* : \mathbf{P}_J \rightarrow \mathbf{P}_I$, together with natural isomorphisms $\text{Id}_{\mathbf{P}_I} \cong \text{id}_I^*$ and $u^* \circ v^* \cong (v \circ u)^*$, see [17, Sec. 1.4]. The functor u^* is called *reindexing* along u .

Assumption 2.5. We assume all fibrations to be cloven in this work.

We are now in the position to take a more abstract look at our initial example.

Example 2.6. First, we note that the fibre of $\text{Fam}(\mathbf{Set})$ above I is isomorphic to \mathbf{Set}^I . Let then $z : \mathbf{1} \rightarrow \mathbb{N}$ and $s : \mathbb{N} \rightarrow \mathbb{N}$ be $z(*) = 0$ and $s(n) = n + 1$, giving us reindexing functors $z^* : \mathbf{Set}^{\mathbb{N}} \rightarrow \mathbf{Set}^1$ and $s^* : \mathbf{Set}^{\mathbb{N}} \rightarrow \mathbf{Set}^{\mathbb{N}}$. By their definition, $z^*(X) = \{X_0\}$ and $s^*(X) = \{X_{n+1}\}_{n \in \mathbb{N}}$, hence the functor G , we used to describe vectors as dialgebra, is $G = \langle z^*, s^* \rangle$. In Sec. 3, we address the structure of F .

We generalise this situation to account for arbitrary data types.

Definition 2.7. Let $P : \mathbf{E} \rightarrow \mathbf{B}$ be a fibration. A *(dependent) data type signature*, parameterised by a category \mathbf{C} , is a pair (F, u) consisting of

- a functor $F : \mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{D}$ with $\mathbf{D} = \prod_{k=1}^n \mathbf{P}_{J_k}$ for some $n \in \mathbb{N}$ and $J_k, I \in \mathbf{B}$, and
- a family u of n morphisms in \mathbf{B} with $u_k : J_k \rightarrow I$ for $k = 1, \dots, n$.

A family u as above induces a functor $\langle u_1^*, \dots, u_n^* \rangle : \mathbf{P}_I \rightarrow \mathbf{D}$, which we will often denote by G_u . This will enable us to define data types for such signatures, but let us first look at an example for the case $\mathbf{C} = \mathbf{1}$, that is, if $F : \mathbf{P}_I \rightarrow \mathbf{D}$ is not parameterised.

Example 2.8. A fibration $P : \mathbf{E} \rightarrow \mathbf{B}$ is said to have dependent coproducts and products, if for each $f : I \rightarrow J$ in \mathbf{B} there are functors \coprod_f and \prod_f from \mathbf{P}_I to \mathbf{P}_J that are respectively left and right adjoint to f^* . For each $X \in \mathbf{P}_I$, we can define a signature, such that $\coprod_f(X)$ and $\prod_f(X)$ arise as data types for these signatures, as follows. Define the constant functor

$$K_X : \mathbf{P}_J \rightarrow \mathbf{P}_I \quad K_X(Y) = X \quad K_X(g) = \text{id}_X.$$

Then (K_X, f) is the signature for coproducts and products. For example, the unit η of the adjunction $\coprod_f \dashv f^*$ will be the initial (K_X, f^*) -dialgebra $\eta_X : K_X(\coprod_f(X)) \rightarrow f^*(\coprod_f(X))$, using that $K_X(\coprod_f(X)) = X$. We come back to this in Ex. 2.10. \square

To define data types in general, we allow them to have additional parameters, that is, we allow signatures (F, u) , where $F : \mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{D}$ and \mathbf{C} is a non-trivial category. Let us first fix some notation. We put $F(V, -)(X) = F(V, X)$ for $V \in \mathbf{C}$, which is a functor $\mathbf{P}_I \rightarrow \mathbf{D}$. Assume that the initial $(F(V, -), G_u)$ -dialgebra $\alpha_V : F(V, \Phi_V) \rightarrow G_u(\Phi_V)$ and final $(G_u, F(V, -))$ -dialgebra $\xi_V : G_u(\Omega_V) \rightarrow F(V, \Omega_V)$ exist. Then we can define functors $\mu(\widehat{F}, \widehat{G}_u) : \mathbf{C} \rightarrow \mathbf{P}_I$ and $\nu(\widehat{G}_u, \widehat{F}) : \mathbf{C} \rightarrow \mathbf{P}_I$, analogous to [18], by

$$\begin{aligned} \mu(\widehat{F}, \widehat{G}_u)(V) &= \Phi_V & \mu(\widehat{F}, \widehat{G}_u)(f : V \rightarrow W) &= (\alpha_W \circ F(f, \text{id}_{\Phi_W}))^- \\ \nu(\widehat{G}_u, \widehat{F})(V) &= \Omega_V & \nu(\widehat{G}_u, \widehat{F})(f : V \rightarrow W) &= (F(f, \text{id}_{\Omega_V}) \circ \xi_V)^\sim, \end{aligned}$$

where the bar and tilde superscripts denote the inductive and coinductive extensions, that is, the unique homomorphism given by initiality and finality, respectively. The reason for the notation $\mu(\widehat{F}, \widehat{G}_u)$ and $v(\widehat{G}_u, \widehat{F})$ is that these are initial and final dialgebras for the functors

$$\widehat{F}, \widehat{G}_u : [\mathbf{C}, \mathbf{P}_I] \rightarrow [\mathbf{C}, \mathbf{D}] \quad \widehat{F}(H) = F \circ \langle \text{Id}_{\mathbf{C}}, H \rangle \quad \widehat{G}_u(H) = G_u \circ H$$

on functor categories. That the families α_V and ξ_V are natural in V follows directly from the definition of the functorial action as (co)inductive extensions. Hence, they give rise to dialgebras $\alpha : \widehat{F}(\mu(\widehat{F}, \widehat{G}_u)) \Rightarrow \widehat{G}_u(\mu(\widehat{F}, \widehat{G}_u))$ and $\xi : \widehat{G}_u(v(\widehat{G}_u, \widehat{F})) \Rightarrow \widehat{F}(v(\widehat{G}_u, \widehat{F}))$.

Definition 2.9. Let (F, u) be a data type signature. An *inductive data type* (IDT) for (F, u) is an initial $(\widehat{F}, \widehat{G}_u)$ -dialgebra with carrier $\mu(\widehat{F}, \widehat{G}_u)$. Dually, a coinductive data type (CDT) for (F, u) is a final $(\widehat{G}_u, \widehat{F})$ -dialgebra, note the order, with the carrier being denoted by $v(\widehat{G}_u, \widehat{F})$. If $\mathbf{C} = \mathbf{1}$, we drop the hats from the notation.

Example 2.10. We turn the definition of the product and coproduct from Ex. 2.8 into actual functors. The observation we use is that the projection functor $\pi_1 : \mathbf{P}_I \times \mathbf{P}_J \rightarrow \mathbf{P}_I$ gives us a ‘‘parameterised’’ constant functor: $K_A^J = \pi_1(A, -)$. If we are given $f : I \rightarrow J$ in \mathbf{B} , then we use the signature (π_1, f) , and define $\coprod_f = \mu(\widehat{\pi}_1, \widehat{f}^*)$ and $\prod_f = v(\widehat{f}^*, \widehat{\pi}_1)$. We check the details of this definition in Thm. 3.2.

3 Data Type Completeness

We now define a class of signatures and functors that should be seen as categorical language for, what is usually called, strictly positive types [3], positive generalised abstract data types [14] or descriptions [8, 9]. Note, however, that none of these treat coinductive types. A *non-dependent* version of strictly positive types that include coinductive types are given in [2].

Let us first introduce some notation. Given categories \mathbf{C}_1 and \mathbf{C}_2 and an object $A \in \mathbf{C}_1$, we denote by $K_A^{\mathbf{C}_1} : \mathbf{C}_1 \rightarrow \mathbf{C}_2$ the functor mapping constantly to A . The projections on product categories are denoted, as usual, by $\pi_k : \mathbf{C}_1 \times \mathbf{C}_2 \rightarrow \mathbf{C}_k$. Using these notations, we can define what we understand to be a data type by mutual induction.

Definition 3.1. A fibration $P : \mathbf{E} \rightarrow \mathbf{B}$ is *data type complete*, if all IDTs and CDTs for *strictly positive signatures* $(F, u) \in \mathcal{S}$ exist, where \mathcal{S} is given by the following rule.

$$\frac{\mathbf{D} = \prod_{i=1}^n \mathbf{P}_{J_i} \quad F \in \mathcal{D}_{\mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{D}} \quad u = (u_1 : J_1 \rightarrow I, \dots, u_n : J_n \rightarrow I)}{(F, u) \in \mathcal{S}_{\mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{D}}}$$

The functors in \mathcal{D} are given by the following rules, assuming that P is data type complete.

$$\begin{array}{c} \frac{A \in \mathbf{P}_J}{K_A^{\mathbf{P}_I} \in \mathcal{D}_{\mathbf{P}_I \rightarrow \mathbf{P}_J}} \quad \frac{\mathbf{C} = \prod_{i=1}^n \mathbf{P}_{J_i}}{\pi_k \in \mathcal{D}_{\mathbf{C} \rightarrow \mathbf{P}_k}} \quad \frac{f : J \rightarrow I \text{ in } \mathbf{B}}{f^* \in \mathcal{D}_{\mathbf{P}_I \rightarrow \mathbf{P}_J}} \quad \frac{F_1 \in \mathcal{D}_{\mathbf{P}_I \rightarrow \mathbf{P}_K}}{F_2 \circ F_1 \in \mathcal{D}_{\mathbf{P}_I \rightarrow \mathbf{P}_J}} \quad \frac{F_2 \in \mathcal{D}_{\mathbf{P}_K \rightarrow \mathbf{P}_J}}{} \\ \frac{F_i \in \mathcal{D}_{\mathbf{P}_I \rightarrow \mathbf{P}_{J_i}} \quad i = 1, 2}{\langle F_1, F_2 \rangle \in \mathcal{D}_{\mathbf{P}_I \rightarrow \mathbf{P}_{J_1} \times \mathbf{P}_{J_2}}} \quad \frac{(F, u) \in \mathcal{S}_{\mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{D}}}{\mu(\widehat{F}, \widehat{G}_u) \in \mathcal{D}_{\mathbf{C} \rightarrow \mathbf{P}_I}} \quad \frac{(F, u) \in \mathcal{S}_{\mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{D}}}{v(\widehat{G}_u, \widehat{F}) \in \mathcal{D}_{\mathbf{C} \rightarrow \mathbf{P}_I}} \end{array}$$

This mutual induction is well-defined, as it can be stratified in the nesting of fixed points.

As a first sanity check, we show that a data type complete fibration has, both, fibrewise and dependent (co)products. These are instances of the following, more general, result.

Theorem 3.2. Suppose $P : \mathbf{E} \rightarrow \mathbf{B}$ is a data type complete fibration. Let $\mathbf{C} = \prod_{i=1}^m \mathbf{P}_{K_i}$ and $\pi_1 : \mathbf{C} \times \mathbf{P}_I \rightarrow \mathbf{C}$ be the first projection. If $G_u : \mathbf{P}_I \rightarrow \mathbf{C}$ is such that (π_1, u) is a signature, then we have the following adjoint situation:

$$\mu(\widehat{\pi}_1, \widehat{G}_u) \dashv G_u \dashv v(\widehat{G}_u, \widehat{\pi}_1).$$

Proof. We only show how the adjoint transposes are obtained in the case of inductive types. Concretely, for a tuple $V \in \mathbf{C}$ and an object $A \in \mathbf{P}_I$, we need to prove the correspondence

$$\frac{f : \mu(\widehat{\pi}_1, \widehat{G}_u)(V) \longrightarrow A \quad \text{in } \mathbf{P}_I}{g : V \longrightarrow G_u A \quad \text{in } \mathbf{C}}$$

Let us use the notation $H = \mu(\widehat{\pi}_1, \widehat{G}_u)$, then the choice of π_1 implies that the initial $(\widehat{\pi}_1, \widehat{G}_u)$ -dialgebra is of type $\alpha : \text{Id}_{\mathbf{C}} \Rightarrow G_u \circ H$, since $\widehat{\pi}_1(H) = \pi_1 \circ (\text{Id}_{\mathbf{C}}, H) = \text{Id}_{\mathbf{C}}$ and $\widehat{G}_u(H) = G_u \circ H$. This allows us to use as transpose of f the morphism $V \xrightarrow{\alpha_V} G_u(H(V)) \xrightarrow{G_u f} G_u A$. As transpose of g , we use the inductive extension of $\widehat{\pi}_1(K_A^{\mathbf{C}})(V) = V \xrightarrow{g} G_u A = \widehat{G}_u(K_A^{\mathbf{C}})(V)$. The proof that this correspondence is natural and bijective follows straightforwardly from initiality. For coinductive types, the result is given by duality. \square

This gives fibrewise coproducts by $+_I = \mu(\widehat{\pi}_1, \widehat{G}_u)$ and products by $\times_I = v(\widehat{G}_u, \widehat{\pi}_1)$, using $u = (\text{id}_I, \text{id}_I)$. Dependent (co)products along $f : I \rightarrow J$ use $u = f$, see Ex. 2.10.

There are many more examples of data types that exist in a data type complete fibration. We describe three fundamental ones.

Example 3.3. 1. The first example are initial and final objects inside the fibres \mathbf{P}_I . Since an initial object is characterised by having a unique morphism to every other object, we define it as an initial dialgebra, namely $\mathbf{0}_I = \mu(\text{Id}, \text{id}_I^*)$. Then there is, for each $A \in \mathbf{P}_I$, a unique morphism $!^A : \mathbf{0}_I \rightarrow A$ given as inductive extension of id_A . Dually, we define the terminal object $\mathbf{1}_I$ in \mathbf{P}_I to be $v(\text{id}^*, \text{Id})$ and for each A the corresponding unique morphism $!_A : A \rightarrow \mathbf{1}_I$ as the coinductive extension of id_A .

Note that this also follows from Thm. 3.2, if we require that (co)inductive data types also exist if $\mathbf{C} = \mathbf{1}$ (the empty product) and $u = \{\}$ (empty family of morphisms). This allows us to define the initial and final object as functors $\mathbf{1} \rightarrow \mathbf{P}_I$.

2. There are several definable notions of equality, provided that \mathbf{B} has binary products. A generic one is propositional equality $\text{Eq} : \mathbf{P}_I \rightarrow \mathbf{P}_{I \times I}$, the left adjoint to the contraction functor $\delta^* : \mathbf{P}_{I \times I} \rightarrow \mathbf{P}_I$, which is induced by the diagonal $\delta : I \rightarrow I \times I$. Thus it is given by the dependent coproduct $\text{Eq} = \coprod_{\delta}$ and the constructor $\text{refl}_X : X \rightarrow \delta^*(\text{Eq} X)$.
3. Assume that there is an object A^ω in \mathbf{B} of streams over A , together with projections to head and tail. Then we can define bisimilarity between streams as CDT for the signature

$$F, G_u : \mathbf{P}_{(A^\omega)^2} \rightarrow \mathbf{P}_{(A^\omega)^2} \times \mathbf{P}_{(A^\omega)^2}$$

$$F = \langle (\text{hd} \times \text{hd})^* \circ K_{\text{Eq}(A)}, (\text{tl} \times \text{tl})^* \rangle \quad \text{and} \quad u = (\text{id}_{A^\omega \times A^\omega}, \text{id}_{A^\omega \times A^\omega}).$$

Note that there is a category $\text{Rel}(\mathbf{E})$ of binary relations in \mathbf{E} by forming the pullback of P along $\Delta : \mathbf{B} \rightarrow \mathbf{B}$ with $\Delta(I) = I \times I$, see [15]. Then we can reinterpret F and G_u by

$$F, G_u : \text{Rel}(\mathbf{E})_{A^\omega} \rightarrow \text{Rel}(\mathbf{E})_{A^\omega} \times \text{Rel}(\mathbf{E})_{A^\omega}$$

$$F = \langle \text{hd}^\# \circ K_{\text{Eq}(A)}, \text{tl}^\# \rangle \quad \text{and} \quad G_u = \langle \text{id}_{A^\omega}^\#, \text{id}_{A^\omega}^\# \rangle,$$

where $(-)^{\#}$ is reindexing in $\text{Rel}(\mathbf{E})$. The final (G_u, F) -dialgebra is a pair of morphisms

$$(\text{hd}_A^{\sim} : \text{Bisim}_A \rightarrow \text{hd}^{\#}(\text{Eq}(A)), \text{tl}_A^{\sim} : \text{Bisim}_A \rightarrow \text{tl}^{\#}(\text{Bisim}_A)).$$

Bisim_A should be thought of to consist of all bisimilarity proofs. Coinductive extensions yield the usual coinduction proof principle, allowing us to prove bisimilarity by establishing a bisimulation relation $R \in \text{Rel}(\mathbf{E})_{A^{\omega}}$ together with $h : R \rightarrow \text{hd}^{\#}(\text{Eq}(A))$ and $t : R \rightarrow \text{tl}^{\#}(R)$, saying that the heads of related streams are equal and that the tails of related streams are again related.

The last example, we give, shall illustrate the additional capabilities of CDTs in the present setup over those currently available in Agda. However, one should note that coinductive types in Agda provide extra power in the sense that destructors can refer to each other. This is equivalent to having a strong coproduct [17, Sec. 10.1 and Def. 10.5.2], which we do not require in the setup of this work and thus A proof of this equivalence is left out because of space constraints.

Example 3.4. A partial stream is a stream together with a, possibly infinite, depth up to which it is defined. Assume that there is an object \mathbb{N}^{∞} of natural numbers extended with infinity and a successor map $s_{\infty} : \mathbb{N}^{\infty} \rightarrow \mathbb{N}^{\infty}$ in \mathbf{B} , we will see how these can be defined below. Then partial streams correspond to the following type declaration.

```
codata PStr (A : Set) : N∞ → Set where
  hd : (n : N∞) → PStr(s∞ n) → A
  tl : (n : N∞) → PStr(s∞ n) → PStr n
```

In an explicit, set-theoretic notation, we can define them as a family indexed by $n \in \mathbb{N}^{\infty}$:

$$\text{PStr}(A)_n = \{s : \mathbb{N} \rightarrow A \mid \forall k < n. k \in \text{dom } s \wedge \forall k \geq n. k \notin \text{dom } s\},$$

where the order on \mathbb{N}^{∞} is given by extending that of the natural numbers with ∞ as strict top element, i.e., such that $k < \infty$ for all $k \in \mathbb{N}$.

The interpretation of $\text{PStr}(A)$ for $A \in \mathbf{P}_1$ in a data type complete fibration is given, similarly to vectors, as the carrier of the final (G_u, F) -dialgebra, where

$$G_u, F : \mathbf{P}_{\mathbb{N}^{\infty}} \rightarrow \mathbf{P}_{\mathbb{N}^{\infty}} \times \mathbf{P}_{\mathbb{N}^{\infty}} \quad G_u = \langle s_{\infty}^*, s_{\infty}^* \rangle \quad F = \left\langle K_{\bar{A}}^{\mathbb{N}^{\infty}}, \text{Id} \right\rangle$$

and $\bar{A} = !_{\mathbb{N}^{\infty}}^*(A) \in \mathbf{P}_{\mathbb{N}^{\infty}}$ is the weakening of A using $!_{\mathbb{N}^{\infty}} : \mathbb{N}^{\infty} \rightarrow \mathbf{1}$. The idea of this signature is that the head and tail of partial streams are defined only on those partial streams that are defined in, at least, the first position. On set families, partial streams are given by the dialgebra $\xi = (\text{hd}, \text{tl})$ with $\text{hd}_n : \text{PStr}(A)_{(s_{\infty} n)} \rightarrow A$ and $\text{tl}_n : \text{PStr}(A)_{(s_{\infty} n)} \rightarrow \text{PStr}(A)_n$ for every $n \in \mathbb{N}^{\infty}$.

We can make this construction functorial in A , using the same “trick” as for sums and products. To this end, we define the functor $H : \mathbf{P}_1 \times \mathbf{P}_{\mathbb{N}^{\infty}} \rightarrow \mathbf{P}_{\mathbb{N}^{\infty}} \times \mathbf{P}_{\mathbb{N}^{\infty}}$ with $H = \langle !_{\mathbb{N}^{\infty}} \circ \pi_1, \pi_2 \rangle$, where π_1 and π_2 are corresponding projection functors, so that $H(A, X) = F(X)$. This gives, by data type completeness, rise to a functor $v(\widehat{G}_u, \widehat{F}) : \mathbf{P}_{\mathbb{N}^{\infty}} \rightarrow \mathbf{P}_{\mathbb{N}^{\infty}}$, which we denote by PStr , together with a pair (hd, tl) of natural transformations. \square

We have seen in the examples above that we would often like to use a data type again as index, which means that we need a mechanism to turn a data type in \mathbf{E} into an index in \mathbf{B} . This is provided by, so called, *comprehension*.

Definition 3.5 (See [17, Lem. 1.8.8, Def. 10.4.7] and [10]). Let $P : \mathbf{E} \rightarrow \mathbf{B}$ be a fibration. If each fibre \mathbf{P}_I has a final object $\mathbf{1}_I$ and these are preserved by reindexing, then there is a fibred *final object functor* $\mathbf{1}_{(-)} : \mathbf{B} \rightarrow \mathbf{E}$. (Note that then $P(\mathbf{1}_I) = I$.) P is a *comprehension category with unit* (CCU), if $\mathbf{1}_{(-)}$ has a right adjoint $\{-\} : \mathbf{E} \rightarrow \mathbf{B}$, the *comprehension*. This gives rise to a functor $\mathcal{P} : \mathbf{E} \rightarrow \mathbf{B}^\rightarrow$ into the arrow category over \mathbf{B} , by mapping $A \mapsto P(\varepsilon_A) : \{A\} \rightarrow P(A)$, where $\varepsilon : \mathbf{1}_{\{-\}} \Rightarrow \text{Id}$ is the counit of $\mathbf{1}_{(-)} \dashv \{-\}$. We often denote $\mathcal{P}(A)$ by π_A and call it the *projection* of A . Finally, P is said to be a *full CCU*, if \mathcal{P} is full.

Note that, in a data type complete category, we can define final objects in each fibre, the preservation of them needs to be required separately.

Example 3.6. In $\text{Fam}(\mathbf{Set})$, the final object functor is given by $\mathbf{1}_I = (I, \{\mathbf{1}\}_{i \in I})$, where $\mathbf{1}$ is the singleton set. Comprehension is defined to be $\{(I, X)\} = \coprod_{i \in I} X_i$ and the projections π_I map then an element of $\coprod_{i \in I} X_i$ to its component $i \in I$.

Using comprehension, we can give a general account to dependent data types.

Definition 3.7. We say that a fibration $P : \mathbf{E} \rightarrow \mathbf{B}$ is a *data type closed category* (DTCC), if it is a CCU, has a terminal object in \mathbf{B} and is data type complete.

As already mentioned, the purpose of introducing comprehension is that it allows us to use data types defined in \mathbf{E} again as index. The terminal object in \mathbf{B} is used to introduce data types without dependencies, like the natural numbers. Let us reiterate on Ex. 3.4.

Example 3.8. Recall that we assumed the existence of extended naturals \mathbb{N}^∞ and the successor map s_∞ on them to define partial streams. We are now in the position to define, in a data type closed category, everything from scratch as follows.

Having defined $+ : \mathbf{P}_1 \times \mathbf{P}_1 \rightarrow \mathbf{P}_1$, see Thm. 3.2, we put $\mathbb{N}^\infty = v(\text{Id}, \mathbf{1} + \text{Id})$ and find the predecessor pred as the final dialgebra on \mathbb{N}^∞ . The successor s_∞ arises as the coinductive extension $(\mathbb{N}^\infty, \kappa_2) \rightarrow (\mathbb{N}^\infty, \text{pred})$, where κ_2 is the coproduct inclusion. Partial streams $\text{PStr} : \mathbf{P}_{\{\mathbb{N}^\infty\}} \rightarrow \mathbf{P}_{\{\mathbb{N}^\infty\}}$ are then given, as in Ex. 3.4, by the final $(\widehat{G}, \widehat{F})$ -dialgebra with $G = \langle \{s_\infty\}^*, \{s_\infty\}^* \rangle$ and $F = \langle !_{\mathbb{N}^\infty} \circ \pi_1, \pi_2 \rangle$. \square

4 Constructing Data Types

In this section, we show how some data types can be constructed through polynomial functors, where I draw from the vast amount of work on polynomial functors that exists in the literature, see [2, 12]. The construction works by, first, reducing dialgebras to (co)algebras and, second, constructing the necessary initial algebras and final coalgebras as fixed points of polynomial functors analogously to the construction of strictly positive types in [2]. This result works thus far only for data types that, if at all, only use dependent coinductive types at the top-level. Nesting of dependent inductive and non-dependent coinductive types works, however, in full generality.

Before we come to polynomial functors and their fixed points, we show that inductive and coinductive data types actually correspond to initial algebras and final coalgebras, respectively.

Theorem 4.1. *Let $P : \mathbf{E} \rightarrow \mathbf{B}$ be a fibration with fibrewise coproducts and dependent sums. If (F, u) with $F : \mathbf{P}_I \rightarrow \mathbf{P}_{J_1} \times \cdots \times \mathbf{P}_{J_n}$ is a signature, then there is an isomorphism*

$$\text{DiAlg}(F, G_u) \cong \text{Alg} \left(\coprod_{u_1} \circ F_1 +_I \cdots +_I \coprod_{u_n} \circ F_n \right)$$

where $F_k = \pi_k \circ F$ is the k th component of F . In particular, existence of inductive data types and initial algebras coincide. Dually, if P has fibrewise and dependent products, then

$$\text{DiAlg}(G_u, F) \cong \text{CoAlg}\left(\prod_{u_1} \circ F_1 \times_I \cdots \times_I \prod_{u_n} \circ F_n\right).$$

In particular, existence of coinductive data types and final coalgebras coincide.

Proof. The first result is given by a simple application of the adjunctions $\coprod_{k=1}^n \dashv \Delta_n$ between the (fibrewise) coproduct and the diagonal, and $\coprod_{u_k} \dashv u_k^*$:

$$\begin{array}{c} FX \longrightarrow G_u X \quad (\text{in } \mathbf{P}_{J_1} \times \cdots \times \mathbf{P}_{J_n}) \\ \hline \hline (\coprod_{u_1} (F_1 X), \dots, \coprod_{u_n} (F_n X)) \longrightarrow \Delta_n X \quad (\text{in } \mathbf{P}_I^n) \\ \hline \hline \coprod_{k=1}^n \coprod_{u_k} (F_k X) \longrightarrow X \quad (\text{in } \mathbf{P}_I) \end{array}$$

That (di)algebra homomorphisms are preserved follows at once from naturality of the used Hom-set isomorphisms. The correspondence for coinductive types follows by duality. \square

To be able to reuse existing work, we work in the following with the codomain fibration $\text{cod} : \mathbf{B}^\rightarrow \rightarrow \mathbf{B}$ for a category \mathbf{B} with pullbacks. Moreover, we assume that \mathbf{B} is locally Cartesian closed, which is equivalent to say that $\text{cod} : \mathbf{B}^\rightarrow \rightarrow \mathbf{B}$ is a closed comprehension category, that is, it is a full CCU with products and coproducts, and \mathbf{B} has a final object, see [17, Thm 10.5.5]. Finally, we need disjoint coproducts in \mathbf{B} , which gives us an equivalence $\mathbf{B}/I+J \simeq \mathbf{B}/I \times \mathbf{B}/J$, see [17, Prop. 1.5.4].

Definition 4.2. A *dependent polynomial* P indexed by I on variables indexed by J is given by a triple of morphisms

$$\begin{array}{ccccc} & & B & & \\ & s \swarrow & \xrightarrow{f} & \searrow t & \\ J & & & & I \end{array}$$

If $J = I = \mathbf{1}$, f is said to be a *(non-dependent) polynomial*. The extension of P is given by the composite

$$[\![P]\!] = \mathbf{B}/J \xrightarrow{s^*} \mathbf{B}/B \xrightarrow{\prod_f} \mathbf{B}/A \xrightarrow{\coprod_I} \mathbf{B}/I,$$

which we denote by $[\![f]\!]$ if f is non-dependent. A functor $F : \mathbf{B}/J \rightarrow \mathbf{B}/I$ is a *dependent polynomial functor*, if there is a dependent polynomial P such that $F \cong [\![P]\!]$.

Remark 4.3. Note that polynomials are called *containers* by Abbott et al. [2, 1], and a polynomial $P = 1 \overset{!}{\leftarrow} B \overset{f}{\rightarrow} A \overset{!}{\rightarrow} 1$ would be written as $A \triangleright f$. Container morphisms, however, are different from those of dependent polynomials, as the latter correspond strong natural transformations [12, Prop. 2.9], whereas the former are in exact correspondence with all natural transformations between extensions [2, Thm. 3.4].

Because of this relation, we will apply results for containers that do not involve morphisms to polynomials. In particular, [2, Prop. 4.1] gives us that we can construct final coalgebras for polynomial functors from initial algebras for polynomial functors. The former are called *M-types* and are denoted by M_f for $f : A \rightarrow B$, whereas the latter are *W-types* and denoted by W_f .

Assumption 4.4. We assume that \mathbf{B} is closed under the formation of W-types, thus is a *Martin-Löf category* in the terminology of [2].

By the above remark, \mathbf{B} then also has all M-types.

Analogously to how [11, Thm. 12] extends [20, Prop. 3.8], we extend here [6, Thm 3.3]. As it was pointed out by one reviewer, this result is actually in [5], the published version of [6].

Theorem 4.5. *If \mathbf{B} has finite limits, then every dependent polynomial functor has a final coalgebra in \mathbf{B}/I .*

Proof. Let $P = I \xleftarrow{s} B \xrightarrow{f} A \xrightarrow{t} I$ be a dependent polynomial, we construct, analogously to [11] the final coalgebra V of $\llbracket P \rrbracket$ as an equaliser as in the following diagram, in which $f \times I$ is a shorthand for $B \times I \xrightarrow{f \times \text{id}_I} A \times I$ and $M_{f \times I}$ is the carrier of the final $\llbracket f \times I \rrbracket$ -coalgebra.

$$\begin{array}{ccccc} & & u_1 & & \\ V & \xrightarrow{g} & M_f & \xrightarrow{\quad} & M_{f \times I} \\ & & u_2 & & \end{array}$$

First, we give u_1 and u_2 , whose definitions are summarised in the following diagrams.

$$\begin{array}{ccc} M_f & \xrightarrow{u_1} & M_{f \times I} \\ \downarrow \xi_f & & \downarrow \xi_{f \times I} \\ \llbracket f \rrbracket(M_f) & & \llbracket f \times I \rrbracket(M_{f \times I}) \\ \downarrow p_{M_f} & & \downarrow \\ \llbracket f \times I \rrbracket(M_f) & \xrightarrow{\llbracket f \times I \rrbracket(u_1)} & \llbracket f \times I \rrbracket(M_{f \times I}) \\ & & \\ M_f & \xrightarrow{u_1} & M_{f \times I} & \xrightarrow{\psi} & M_{f \times I} \\ & & \downarrow \xi_{f \times I} & & \downarrow \xi_{f \times I} \\ & & \llbracket f \times I \rrbracket(M_{f \times I}) & & \llbracket f \times I \rrbracket(M_{f \times I}) \\ & & \downarrow \Sigma_{A \times I} K & & \downarrow \\ & & \llbracket f \times I \rrbracket(M_{f \times I} \times B) & \xrightarrow{\llbracket f \times I \rrbracket(\phi)} & \llbracket f \times I \rrbracket(M_{f \times I}) \end{array}$$

These diagrams shall indicate that u_1 is given as coinductive extensions and ψ as one-step definition (which can be defined using coproducts), using that $M_{f \times I}$ is a final coalgebra. The maps involved in the diagram are given as follows, which we sometimes spell out in the internal language of cod, see for example [1], as this is sometimes more readable.

- $p : \Sigma_A \Pi_f \Rightarrow \Sigma_{A \times I} \Pi_{f \times I}$ is the natural transformation that maps (a, v) to $(a, t(a), v)$. It is given by the extension $\llbracket \alpha, \beta \rrbracket : \llbracket f \rrbracket \Rightarrow \llbracket f \times I \rrbracket$ of the morphism of polynomials [12]

$$\begin{array}{ccc} B & \xrightarrow{f} & A \\ \beta \downarrow \lrcorner & & \downarrow \alpha \\ B \times I & \xrightarrow{f \times I} & A \times I \end{array}$$

where $\alpha = \langle \text{id}, t \rangle$ and $\beta = \langle \text{id}, t \circ f \rangle$.

- The map $K : \Pi_{f \times I}(M_{f \times I}) \rightarrow \Pi_{f \times I}(M_{f \times I} \times B)$ is given as transpose of $\langle \varepsilon_{M_{f \times I}}, \pi_1 \circ \pi \rangle : (f \times I)^*(\Pi_{f \times I}(M_{f \times I})) \rightarrow M_{f \times I} \times B$, where ε is the counit of the product (evaluation) and π is the context projection. In the internal language K is given by $K v = \lambda(b, i). (v(b, i), b)$.

- $\phi : M_{f \times I} \times B \rightarrow M_{f \times I}$ is constructed as coinductive extension as in the following diagram

$$\begin{array}{ccc}
M_{f \times I} \times B & \xrightarrow{\phi} & M_{f \times I} \\
\downarrow \xi_{f \times I} \times \text{id} & & \downarrow \xi_{f \times I} \\
[\![f \times I]\!](M_{f \times I}) \times B & & \\
\downarrow e & & \downarrow \xi_{f \times I} \\
[\![f \times I]\!](M_{f \times I} \times B) & \xrightarrow{[\![f \times I]\!](\phi)} & [\![f \times I]\!](M_{f \times I})
\end{array}$$

Here e is given by $e((a, i, v), b) = (a, sb, \lambda(b', sb).(v(b', i), b'))$.

The important property, which allows us to prove that $\xi_f : M_f \rightarrow [\![f]\!](M_f)$ restricts to $\xi' : V \rightarrow [\![P]\!](V)$ and that ξ' is a final coalgebra, is that $x : V_i \iff \xi_f x = (a : A, v : \Pi_f M_f), t a = i$ and $(\forall b : B. f b = a \Rightarrow v b : V_{sb})$. The direction from left to right is given by simple a calculation, whereas the other direction can be proved by establishing a bisimulation and between $u_1 x$ and $u_2 x$.

Hence V , given as a subobject of M_f , is indeed the final $[\![P]\!]$ -coalgebra in \mathbf{B}/I . \square

Combining this with [2, Prop. 4.1], we have that the existence of final coalgebras for dependent polynomial functors follows from the existence of initial algebras of (non-dependent) polynomial functors. This gives us the possibility of interpreting non-nested fixed points in any Martin-Löf category as follows.

First, we observe that the equivalence $\mathbf{B}/I+J \simeq \mathbf{B}/I \times \mathbf{B}/J$ allows us to rewrite the functors from Thm. 4.1 to a form that is closer to polynomial functors:

$$\begin{aligned}
\coprod_{u_1} \circ F_1 +_I \cdots +_I \coprod_{u_n} \circ F_n &\cong \coprod_u F' \\
\prod_{u_1} \circ F_1 \times_I \cdots \times_I \prod_{u_n} \circ F_n &\cong \prod_u F',
\end{aligned}$$

where $J = J_1 + \cdots + J_n$, $u : J \rightarrow I$ is given by the cotupling $[u_1, \dots, u_n]$ and $F' : \mathbf{B}/I \rightarrow \mathbf{B}/J$ is given by $F' = \langle F_1, \dots, F_n \rangle : \mathbf{B}/I \rightarrow \prod_{i=1}^n \mathbf{B}/J_i \simeq \mathbf{B}/J$. Thus, if we establish that F' is a polynomial functor, we get that $\coprod_u F'$ and $\prod_u F'$ are polynomial functors, see [1]. For non-nested fixed points, that is, F_k is either a constant functor, given by composition or reindexing, this is immediate, as dependent polynomials can be composed and are closed under constant functors and reindexing, see [12].

We say that a dependent polynomial is *parametric*, if it is of the following form.

$$K + I \xleftarrow{s} B \xrightarrow{f} A \xrightarrow{t} I$$

Such polynomials represent polynomial functors $\mathbf{B}/K \times \mathbf{B}/I \rightarrow \mathbf{B}/I$ and allow us speak about nested fixed points just as we have done in Sec. 2. What thus remains is that fixed points of parametric dependent polynomial functors, in the sense of Sec. 2, are again dependent polynomial functors.

The proof of this is literally the same as that for containers [1, Sec. 5.3-5.5] or non-dependent polynomials [11], except that we need to check some extra conditions regarding the indexing.

Theorem 4.6. *Initial algebras and final coalgebras of parametric, dependent polynomial functors are again dependent polynomial functors.*

Proof. Let

$$\begin{aligned} F = J &\xleftarrow{s} B \xrightarrow{f} A \xrightarrow{t} I \\ G = I &\xleftarrow{u} D \xrightarrow{g} C \xrightarrow{v} I \end{aligned}$$

be dependent polynomials and $H(X, Y) = \llbracket F \rrbracket \times_I \llbracket G \rrbracket$ be the parametric dependent polynomial functor in question. Assuming that there is a polynomial

$$J \xleftarrow{x} Q \xrightarrow{h} P \xrightarrow{y} I$$

so that for $K = \coprod_y \prod_h x^*$ we have $K(X) \cong H(X, K(X))$, we can calculate, as in [1], that we need to have isomorphisms

$$\begin{aligned} \psi : A \times_I \llbracket G \rrbracket(P) &\cong P \\ \varphi : B + \coprod_g \varepsilon^* Q &\cong \psi^*(Q) \end{aligned}$$

where $B + \coprod_g \varepsilon^* Q$ is, as in loc. cit., is an abbreviation for $B_a + \coprod_{d:D_c} Q(r d)$ in the context $(a, (c, r)) : A \times_I \llbracket G \rrbracket(P)$. If $K(X)$ shall be an initial algebra, ψ must be an initial algebra as well, whereas if $K(X)$ shall be a final coalgebra, ψ must be one. The isomorphism φ is given as the initial $(\psi^{-1})^*(B + \coprod_g \varepsilon^*)$ -algebra in both cases, see [1]. This we use to define $x : Q \rightarrow J$ as the inductive extension of the map $[s, \pi_2] : (\psi^{-1})^*(B + \coprod_g \varepsilon^* J) \rightarrow J$. Given these definitions, the following diagrams commute.

$$\begin{array}{ccc} A \times_I \llbracket G \rrbracket(P) & \xrightarrow{\psi} & P \\ & \searrow & \swarrow y \\ & I & \end{array} \quad \begin{array}{ccc} B + \coprod_g \varepsilon^* Q & \xrightarrow{\varphi} & \psi^* Q \\ & \searrow & \swarrow [s, x \circ \pi_2] \\ & J & \end{array}$$

This gives us that the isomorphism given in the proofs of [1, Prop. 5.3.1, Prop. 5.4.2] also work for the dependent polynomial case. The rest of the proofs in loc. cit. go then through, as well. Thus K is in both cases again given by a dependent polynomial. \square

Summing up, we are left with the following result.

Corollary 4.7. *All data types for strictly positive signatures can be constructed in any Martin-Löf category.*

Let us see, by means of an example, how the construction in the proof of Thm. 4.5 works intuitively.

Example 4.8. Recall from Ex. 3.4 that partial streams are given by the declaration

codata PStr (A : Set) : $\mathbb{N}^\infty \rightarrow \text{Set}$ **where**

$$\begin{aligned} \text{hd} : (n : \mathbb{N}^\infty) &\rightarrow \text{PStr}(s_\infty n) \rightarrow A \\ \text{tl} : (n : \mathbb{N}^\infty) &\rightarrow \text{PStr}(s_\infty n) \rightarrow \text{PStr } n \end{aligned}$$

By Thm. 4.1, we can construct PStr as the final coalgebra of $F : \mathbf{B}/\mathbf{1} \times \mathbf{B}/\mathbb{N}^\infty \rightarrow \mathbf{B}/\mathbb{N}^\infty$ with $F(A, X) = \prod_{s_\infty} !^* A \times \prod_{s_\infty} X$. Note that F is isomorphic to $\mathbf{B}/\mathbf{1} \times \mathbf{B}/\mathbb{N}^\infty \simeq \mathbf{B}/\mathbf{1} + \mathbb{N}^\infty \xrightarrow{\llbracket P \rrbracket} \mathbf{B}/\mathbb{N}^\infty$, where P is the polynomial

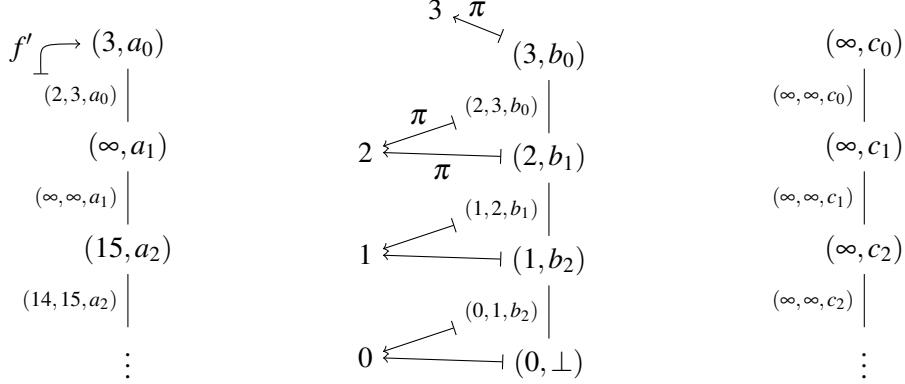
$$P = \mathbf{1} + \mathbb{N}^\infty \xleftarrow{g} 2 \times \mathbb{N}^\infty \xrightarrow{f} \mathbb{N}^\infty \xrightarrow{\text{id}} \mathbb{N}^\infty \quad g(i, k) = \begin{cases} \kappa_1^*, & i = 1 \\ \kappa_2 k, & i = 2 \end{cases} \quad f(i, k) = s_\infty k.$$

If we now fix an object $A \in \mathbf{B}/\mathbf{1}$, then $F(A, -) \cong \llbracket P' \rrbracket$ for the polynomial P' given by

$$P' = \mathbb{N}^\infty \xleftarrow{\pi} \sum_{\mathbb{N}^\infty} \sum_{s_\infty} \prod_{s_\infty} !^* A \xrightarrow{f'} \sum_{\mathbb{N}^\infty} \prod_{s_\infty} !^* A \xrightarrow{\pi} \mathbb{N}^\infty,$$

where π is the projection on the index of a dependent sum and $f'(n, (s_\infty n, v)) = (s_\infty n, v)$.

Recall that we construct in Thm. 4.5 the final coalgebra of $\llbracket P' \rrbracket$ as a subobject of $M_{f'}$. Below, we present three trees that are elements of $M_{f'}$, where only the second and third are actually selected by the equaliser taken in Thm. 4.5.



Here we denote a pair $(k, v) : \sum_{\mathbb{N}^\infty} \prod_{s_\infty} !^* A$ with $k = s_\infty n$ and $vn = a$ by (k, a) , or if $k = 0$ by $(0, \perp)$. Moreover, we indicate the matching of indices in the second tree, which is used to form the equaliser. Note that the second tree is an element of $\text{PStr}(A)^3$, whereas the third is in $\text{PStr}(A)^\infty$. \square

5 Conclusion and Future Work

We have seen how dependent inductive and coinductive types with type constructors, in the style of Agda, can be given semantics in terms of data type closed categories (DTCC), with the restriction that destructors of coinductive types are not allowed to refer to each other. This situation is summed up in the following table.

Condition	Use/Implications
Cloven fibration	Definition of signatures and data types
Data type completeness	Construction of types indexed by objects in base (e.g., vectors for $\mathbb{N} \in \mathbf{B}$) and types agnostic of indices (e.g., initial and final objects, sums and products)
Data type closedness	Constructed types as index; Full interpretation of data types

Moreover, we have shown that a large part of these data types can be constructed as fixed points of polynomial functors.

Let us finish by discussing directions for future work. First, a full interpretation of syntactic data types has also still to be carried out. Here one has to be careful with type equality, which is usually dealt with using split fibrations and a Beck-Chevalley condition. The latter can be defined generally for the data types of this work, in needs to be checked, however, whether this condition is sufficient for giving a sound interpretation. Finally, the idea of using dialgebras has found its way into the syntax of higher inductive types [7], though in that work the used format of dialgebras is likely to be too liberal to

guarantee the existence of semantics. The reason is that the shape of dialgebras used in the present work ensures that we can construct data types from (co)coalgebras, whereas this is not the case in [7]. Thus it is to be investigated what the right notion of dialgebras is for capturing higher (co)inductive types, such that their semantics in terms of trees can always be constructed.

References

- [1] Michael Abbott (2003): *Categories of Containers*. Ph.D. thesis, Leicester.
- [2] Michael Abbott, Thorsten Altenkirch & Neil Ghani (2005): *Containers: Constructing strictly positive types*. *Theoretical Computer Science* 342(1), pp. 3–27, doi:10.1016/j.tcs.2005.06.002.
- [3] Thorsten Altenkirch & Peter Morris (2009): *Indexed containers*. In: *Logic In Computer Science, 2009. LICS’09. 24th Annual IEEE Symposium on*, IEEE, pp. 277–285, doi:10.1109/LICS.2009.33.
- [4] Thorsten Altenkirch, Peter Morris, Fredrik Nordvall Forsberg & Anton Setzer (2011): *A Categorical Semantics for Inductive-Inductive Definitions*. In Andrea Corradini, Bartek Klin & Corina Cîrstea, editors: *Algebra and Coalgebra in Computer Science, Lecture Notes in Computer Science* 6859, Springer Berlin Heidelberg, pp. 70–84, doi:10.1007/978-3-642-22944-2_6.
- [5] Benno van den Berg & Federico De Marchi (2007): *Non-well-founded trees in categories*. *Annals of Pure and Applied Logic* 146(1), pp. 40–59, doi:10.1016/j.apal.2006.12.001.
- [6] Benno van den Berg & Federico de Marchi (2004): *Non-well-founded trees in categories*. arXiv:math/0409158. Available at <http://arxiv.org/abs/math/0409158>.
- [7] Paolo Capriotti (2014): *Mutual and Higher Inductive Types in Homotopy Type Theory*. Available at <http://www.cs.nott.ac.uk/~pvc/away-day-2014/mhit.pdf>.
- [8] James Chapman, Pierre-Évariste Dagand, Conor McBride & Peter Morris (2010): *The Gentle Art of Levitation*. In: *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming, ICFP ’10*, ACM, New York, NY, USA, pp. 3–14, doi:10.1145/1863543.1863547.
- [9] P.-E. Dagand & C. McBride (2013): *A Categorical Treatment of Ornaments*. In: *2013 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS)*, pp. 530–539, doi:10.1109/LICS.2013.60.
- [10] Clément Fumex, Neil Ghani & Patricia Johann (2011): *Indexed Induction and Coinduction, Fibrationally*. In Andrea Corradini, Bartek Klin & Corina Cîrstea, editors: *Algebra and Coalgebra in Computer Science, LNCS* 6859, Springer, pp. 176–191, doi:10.1007/978-3-642-22944-2_13.
- [11] Nicola Gambino & Martin Hyland (2004): *Wellfounded Trees and Dependent Polynomial Functors*. In: *Types for Proofs and Programs, LNCS* 3085, Springer, pp. 210–225, doi:10.1007/978-3-540-24849-1_14.
- [12] Nicola Gambino & Joachim Kock (2013): *Polynomial functors and polynomial monads*. *Math. Proc. Camb. Philos. Soc.* 154(01), pp. 153–192, doi:10.1017/S0305004112000394. Available at <http://arxiv.org/abs/0906.4931>.
- [13] Tatsuya Hagino (1987): *A typed lambda calculus with categorical type constructors*. In: *Category Theory in Computer Science*, pp. 140–157.
- [14] Makoto Hamana & Marcelo Fiore (2011): *A Foundation for GADTs and Inductive Families: Dependent Polynomial Functor Approach*. In: *Proceedings of the Seventh Workshop on Generic Programming, WGP ’11*, ACM, New York, NY, USA, pp. 59–70, doi:10.1145/2036918.2036927.
- [15] Claudio Hermida & Bart Jacobs (1997): *Structural Induction and Coinduction in a Fibration Setting*. *Inf. Comput.* 145, pp. 107–152, doi:10.1006/inco.1998.2725.
- [16] Martin Hofmann (1995): *On the Interpretation of Type Theory in Locally Cartesian Closed Categories*. In: *Proceedings of Computer Science Logic, 8th Workshop, CSL’94, Selected Papers, LNCS* 933, Springer, pp. 427–441, doi:10.1007/BFb0022273.

- [17] B. Jacobs (1999): *Categorical Logic and Type Theory. Studies in Logic and the Foundations of Mathematics* 141, North Holland, Amsterdam.
- [18] Jiho Kim (2010): *Higher-order Algebras and Coalgebras from Parameterized Endofunctors*. *Electronic Notes in Theoretical Computer Science* 264(2), pp. 141–154, doi:10.1016/j.entcs.2010.07.018.
- [19] Andres Löh & José Pedro Magalhães (2011): *Generic programming with indexed functors*. In: *Proceedings of the seventh ACM SIGPLAN workshop on Generic programming*, ACM, pp. 1–12. Available at <http://dl.acm.org/citation.cfm?id=2036920>.
- [20] Ieke Moerdijk & Erik Palmgren (2000): *Wellfounded trees in categories*. *Annals of Pure and Applied Logic* 104(1–3), pp. 189–218, doi:10.1016/S0168-0072(00)00012-9.
- [21] R. A. G. Seely (1984): *Locally cartesian closed categories and type theory*. *Math. Proc. Camb. Philos. Soc.* 95(01), pp. 33–48, doi:10.1017/S0305004100061284.

On Modal μ -Calculus over Finite Graphs with Bounded Strongly Connected Components

Giovanna D'Agostino

Università degli Studi di Udine
DIMI (Dipartimento di Matematica e Informatica)
Udine, Italy
dagostin@dimi.uniud.it

Giacomo Lenzi

Università degli Studi di Salerno
DMI (Dipartimento di Matematica e Informatica)
Fisciano (SA), Italy
gilenzi@unisa.it

For every positive integer k we consider the class $SCCk$ of all finite graphs whose strongly connected components have size at most k . We show that for every k , the Modal μ -Calculus fixpoint hierarchy on $SCCk$ collapses to the level $\Delta_2 = \Pi_2 \cap \Sigma_2$, but not to $Comp(\Sigma_1, \Pi_1)$ (compositions of formulas of level Σ_1 and Π_1). This contrasts with the class of all graphs, where $\Delta_2 = Comp(\Sigma_1, \Pi_1)$.

1 Introduction

The subject of this paper is *Modal μ -Calculus*, an extension of Modal Logic with operators for least and greatest fixpoints of monotone functions on sets. This logic, introduced by Kozen in [17], is a powerful formalism capable of expressing inductive as well as coinductive concepts and beyond (e.g. safety, liveness, fairness, termination, etc.) and is widely used in the area of verification of computer systems, be them hardware or software, see [5].

Like Modal Logic, the μ -Calculus can be given a Kripke semantics on graphs. It results that on arbitrary graphs, the more we nest least and greatest fixpoints, the more properties we obtain. In other words, on the class of all graphs, the fixpoint alternation hierarchy $(\Sigma_n, \Pi_n, \Delta_n)$ is infinite, see [3] and [4]. Whereas the low levels have a clear “temporal logic” meaning (Π_1 gives safety, Σ_1 gives liveness, Π_2 gives fairness), the meaning of the higher levels can be understood in terms of *parity games* (a formula with n alternations corresponds to a parity game with n priorities).

The fixpoint hierarchy may not be infinite anymore if we restrict the semantics to subclasses of graphs. For instance, over the class of all *transitive* graphs (the class known as K4 in Modal Logic), the hierarchy collapses to the class $Comp(\Sigma_1, \Pi_1)$, that is, to compositions of alternation-free formulas, see [1] and [7]. As another example, it is not difficult to show that on finite *trees*, the μ -Calculus collapses to the class $\Delta_1 = \Sigma_1 \cap \Pi_1$.

In this paper we are interested in some classes of finite graphs which generalize finite trees and (up to bisimulation) finite transitive graphs, but are not too far from them. Our classes are characterized by having strongly connected components (s.c.c.) of size bounded by a finite constant.

Note that:

- every finite tree has all s.c.c. of size one, and
- every finite transitive graph vertex-colored with k colors is bisimilar to a graph whose s.c.c. have size k .

In our opinion, the size of the strongly connected components could be an interesting measure of complexity for finite graphs, analogous, but not equivalent, to the important graph-theoretic notion of

tree width, see [12], [23] and [14]. Measures of complexity of finite graphs are gaining importance in the frame of *Fixed Parameter Complexity Theory*, where many problems intractable on arbitrary graphs become feasible when some parameter is fixed, see [11].

The purpose of this paper is to determine to what extent the alternating fixpoint hierarchy collapses on finite graphs with s.c.c. of bounded size. First we give a Π_2 upper bound, which by complementation becomes $\Delta_2 = \Sigma_2 \cap \Pi_2$. Then we show that the Δ_2 bound is tight, in the sense that already on finite graphs with s.c.c. of size one, the μ -Calculus does not collapse to $\text{Comp}(\Sigma_1, \Pi_1)$, that is, to compositions of alternation-free formulas. The latter can be considered as a level very close to Δ_2 in the alternation hierarchy. In fact, Δ_2 includes $\text{Comp}(\Sigma_1, \Pi_1)$ (in arbitrary classes of graphs), and the two levels coincide on the class of all graphs (see [18]).

1.1 Related work

This paper concerns expressiveness of the μ -Calculus in subclasses of graphs, a subject already treated in previous papers. We mention some of them.

An important theorem in the area (despite it predates the invention of Modal μ -Calculus) is the De Jongh-Sambin Theorem, see [25]. The theorem considers the important modal logic GL (Gödel-Löb); this logic, besides being deeply studied as a logic of provability, corresponds to a natural class of graphs, i.e. the transitive, wellfounded graphs. The theorem says that fixpoint modal equations in GL have a unique solution. From the theorem it follows that in GL, the μ -Calculus collapses to Modal Logic, see [27] and [28]. For a proof of this collapse independent of the De Jongh-Sambin Theorem, see [2]; in that paper the collapse is also extended to an extension of the μ -Calculus, where fixpoint variables are not necessarily in positive positions in the formulas.

The work [1] contains a proof of the collapse of the μ -Calculus to the alternation free fragment over transitive graphs (different proofs of this collapse can be found in [8] and [7], see below); the μ -Calculus hierarchy is also studied in other natural classes of graphs, such as the symmetric and transitive class, where it collapses to Modal Logic, and the reflexive class, where the hierarchy is strict.

Recall that [26] characterizes Modal Logic as the bisimulation invariant fragment of First Order Logic, and that likewise, [13] characterizes the μ -Calculus as the bisimulation invariant fragment of Monadic Second Order Logic. The work [8] extends the results of [26] and [13] to several subclasses of graphs, including transitive graphs, rooted graphs, finite rooted graphs, finite transitive graphs, well-founded transitive graphs, and finite equivalence graphs (all these classes except the first one are not first order definable, so classical model theory cannot be directly applied; rather, [8] uses Ehrenfeucht style games). An unexpected behavior arises over finite transitive frames: the bisimulation invariant fragments of First Order and Monadic Second Order Logic coincide, despite μ -Calculus and Modal Logic do not coincide. These fragments are characterized in [8] by means of suitable modal-like operators. From the above results the authors obtain the collapse of the μ -Calculus over transitive frames, as well as the inclusion of the μ -calculus in First Order Logic over finite transitive frames.

Finally we mention that [7] gives a proof of the first order definability of the μ -Calculus over finite transitive frames which is independent from the work in [8], and contains a particular case of Theorem 5.1 below (namely, the case of the graphs called “simple” in [7], i.e. such that every s.c.c. has at most one vertex for each possible color).

2 Preliminaries on Modal μ -Calculus

2.1 Syntax

The syntax of a μ -Calculus formula ϕ (in negation normal form) is the following:

$$\phi ::= X \mid P \mid \neg P \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \diamond \phi \mid \square \phi \mid \mu X. \phi \mid \nu X. \phi,$$

where X ranges over a countable set FV of fixpoint variables, and P ranges over a countable set At of atomic propositions.

The boolean connectives are \neg (negation), \wedge (conjunction) and \vee (disjunction). The modal operators are \diamond (diamond) and \square (box).

Finally, there are the fixpoint operators μ and ν . Intuitively, $\mu X. \phi(X)$ denotes the least fixpoint of the function ϕ (a function mapping sets to sets), and $\nu X. \phi(X)$ denotes the greatest such fixpoint.

Note that negation is applied only in front of atomic propositions. So, not every formula has a negation. However, every sentence (i.e., every formula without free variables) does have a negation, obtained by applying the De Morgan dualities between the pairs \wedge and \vee , \diamond and \square , and μ and ν (the last duality is given by $\neg \mu X. \phi(X) = \nu X. \neg \phi(\neg X)$).

Free and bound fixpoint variables, as well as scopes of fixpoint operators, can be defined in complete analogy with First Order Logic (where fixpoint operators are treated in analogy with first order quantifiers).

The formulas of the μ -Calculus can be composed in a natural way. Let ϕ be a formula and let P be an atom of ϕ . Suppose that ψ is a formula free for P in ϕ (that is, ψ has no free variables X such that some occurrence of P is in the scope of some fixpoint μX or νX). Then we can replace P with ψ everywhere in ϕ . We obtain a μ -calculus formula χ which we call the *composition* of ϕ and ψ (with respect to the atom P).

2.2 Fixpoint hierarchy

The μ -Calculus formulas can be classified according to the alternation depth of their fixpoints. Formally we have a hierarchy of classes $\Sigma_n, \Pi_n, \Delta_n$ as follows.

First, $\Sigma_0 = \Pi_0$ is the set of the formulas without fixpoints.

Then, Π_{n+1} is the smallest class containing $\Sigma_n \cup \Pi_n$ and closed under composition and ν operators.

Dually, Σ_{n+1} is the smallest class containing $\Sigma_n \cup \Pi_n$ and closed under composition and μ operators.

Note that a property is in Π_n if and only if its negation is in Σ_n , and conversely. In this sense, the classes Σ_n and Π_n are dual.

Finally, a property is said to be in Δ_n if it is both in Σ_n and in Π_n .

The *alternation depth* of a μ -Calculus definable property is the least n such that the property is in $\Sigma_n \cup \Pi_n$.

2.3 Graphs and trees

A (*directed*) *graph* is a pair $G = (V, R)$, where V is a set of vertices and R is a binary edge relation on V . Sometimes we denote V by $V(G)$ and R by $R(G)$.

Likewise, an *undirected graph* is a pair $G = (V, S)$ where V is a set of vertices and S is a symmetric relation on V . That is, xSy must imply ySx .

Note that to every directed graph we can associate the underlying undirected graph, by letting $S = R \cup R^{-1}$ (i.e. S is the symmetric closure of R).

A *successor* of a vertex v in G is a vertex w such that vRw . The set $Succ(v)$ is the set of all successors of v in G . We also say that v is a *predecessor* of w .

A *path of length n* in a graph G from v to w is a finite sequence v_1, v_2, \dots, v_n of vertices such that $v_1 = v$, $v_n = w$ and $v_i R v_{i+1}$ for $1 \leq i < n$. A *descendant* of v is a vertex w such that there is a path from v to w .

The *strongly connected component* of a vertex $v \in V$ in a graph G is v itself plus the set of all $w \in V$ such that there is a path from v to w and conversely.

For a positive integer k , we denote by $SCCk$ the class of all finite graphs whose strongly connected components have size at most k .

A *tree* is a graph T having a vertex r (the root) such that for every vertex v of T there is a unique path from r to v .

The *height* of a vertex v of a tree T is the length of the unique path from r to v .

A *subtree* of a tree T is a subset U of T which is still a tree with respect to the induced edge relation $R(T) \cap U^2$.

If $Pred$ is a set of unary predicates, a *Pred-colored graph* is a graph G equipped with a “satisfaction” relation $Rsat \subseteq Pred \times V(G)$, which intuitively specifies which unary predicates are true in which vertices.

One also thinks of the set $Powerset(Pred)$ as a set of “colors” of the vertices of G , where the color of v is the set of all predicates $P \in Pred$ such that $P Rsat v$ holds.

A *pointed graph* is a graph equipped with a distinguished vertex. Similarly one defines pointed colored graphs.

2.4 Semantics

Like in usual Modal Logic, the formulas of the μ -Calculus can be interpreted on (colored pointed) graphs via Kripke semantics. One defines inductively a *satisfaction* relation between graphs and formulas. The clauses of the satisfaction relation are the usual ones for Modal Logic, plus two new rules which are specific for fixpoints.

A pointed, $At \cup FV$ -colored graph $(G, Rsat, v)$ satisfies an atom P if $P Rsat v$ holds, satisfies $\neg P$ if it does not satisfy P , and satisfies a fixpoint variable X if $X Rsat v$ holds.

For the boolean clauses, $(G, Rsat, v)$ satisfies $\phi \wedge \psi$ if it satisfies ϕ and ψ ; and it satisfies $\phi \vee \psi$ if it satisfies ϕ or ψ .

For the modal clauses, $(G, Rsat, v)$ satisfies $\diamond \phi$ if there is w with vRw and $(G, Rsat, w)$ satisfies ϕ ; and it satisfies $\square \phi$ if for every w with vRw we have that $(G, Rsat, w)$ satisfies ϕ .

For the fixpoint clauses, the idea is that $\mu X.\phi(X)$ and $\nu X.\phi(X)$ denote sets which are the least and greatest solutions of the fixpoint equation $X = \phi(X)$, respectively.

Formally, $(G, Rsat, v)$ satisfies a formula $\mu X.\phi$ if v belongs to every set E equal to $\phi(E)$, where $\phi(E)$ is the set of all vertices w such that $(G, Rsat[X := E], w)$ satisfies ϕ , and where $Rsat[X := E]$ is the same relation as $Rsat$, except that X $Rsat[X := E] z$ holds if and only if $z \in E$.

Dually, $(G, Rsat, v)$ verifies a formula $\nu X.\phi(X)$ if v belongs to some set E equal to $\phi(E)$.

A kind of “global” modalities are $\square^*\phi = \nu X.\phi \wedge \square X$ and the dual $\diamond^*\phi = \mu X.\phi \vee \diamond X$. The former means that ϕ is true “always” (i.e. in all descendants of the current vertex), and the latter means that ϕ is true “sometimes” (i.e. in some descendant).

2.5 Bisimulation

Bisimulation between graphs is a generalization of isomorphism, which is intended to capture the fact that two graphs have the same observable behavior.

A *bisimulation* between two (*Pred*-colored) graphs G, H is a relation $B \subseteq V(G) \times V(H)$, such that if vBw holds, then:

- v and w satisfy the same predicates in *Pred*;
- if vRv' in G , then there is $w' \in H$ such that wRw' in H and $v'Bw'$;
- dually, if wRw' in H , then there is $v' \in G$ such that vRv' in G and $v'Bw'$.

Two pointed, colored graphs (G, v) and (H, w) are called *bisimilar* if there is a bisimulation B between G and H such that vBw .

Every pointed graph (G, v) is bisimilar to a tree, and there is a canonical such tree, called the *unfolding* of (G, v) , denoted by $U(G, v)$. The vertices of $U(G, v)$ are the finite paths of G starting from v . There is an edge from π to π' if π' is obtained from π by adding one step at the end. A path π satisfies a predicate if and only if its last vertex does. It results that the function mapping a path to its last vertex is a bisimulation between $U(G, v)$ and (G, v) .

Like Modal Logic, the μ -Calculus is invariant under bisimulation (in fact it can be viewed as a kind of infinitary modal logic). In particular, every μ -Calculus formula which is valid on all trees is valid on all graphs as well.

2.6 Tree width

In this subsection we define tree decompositions and tree width of an undirected graph $G = (V, S)$.

Intuitively, the tree width of a graph measures how far the graph is from being a tree. Being close to a tree is a virtue, because many graph theoretic problems become much easier when restricted to trees.

For the benefit of software verification, [21] argues that programs in many programming languages have control flow diagrams with low tree width (as long as no goto command or similar is used).

Formally, a *tree decomposition* of the graph G is a pair (\mathcal{X}, T) , where $\mathcal{X} = \{X_1, \dots, X_n\}$ is a family of subsets of V , and T is a tree whose nodes are the subsets X_i , satisfying the following properties:

- The union of all sets X_i equals V . That is, each graph vertex is associated with at least one tree node.

- For every edge (v, w) in the graph, there is a subset X_i that contains both v and w . That is, vertices are adjacent in the graph only when the corresponding subtrees have a node in common.
- If X_i and X_j both contain a vertex v , then all nodes X_z of the tree in the (unique) path between X_i and X_j contain v as well. That is, the nodes associated with vertex v form a connected subset of T .

The *width* of a tree decomposition is the size of its largest set X_i minus one. The *tree width* $tw(G)$ of a graph G is the minimum width among all possible tree decompositions of G .

In this paper, we define the tree width of a *directed* graph as the tree width of the underlying undirected graph. We denote by TWk the class of all finite directed graphs whose tree width is at most k .

As a first remark, the tree width of a tree is one (the definition is adjusted so that this is true). In fact, as a tree decomposition we can take all edges of the tree.

Moreover, tree width does not change if we add or remove loops (i.e. edges (v, v)) to the graph.

Less trivially, we have examples of applications of tree width in the following areas:

- Robertson-Seymour Graph Minors Theory, see [23] and [24];
- Complexity Theory, e.g. the Hamiltonian path problem can be solved in polynomial time if the directed tree width is bounded by a constant, see [14], where the directed tree width is a variant of tree width tailored for directed graphs.

3 Model checking and parity games

The μ -Calculus model checking problem is the following algorithmic problem: given a formula ϕ of Modal μ -Calculus and a finite graph G , decide whether ϕ is true in G .

A kind of games closely related to the μ -Calculus model checking problem is *parity games*. In fact, checking a μ -Calculus formula in a finite graph is a problem computationally equivalent (in polynomial time) to solving a finite parity game.

Parity games can be described as follows. There are two players, let us call them *Odd* and *Even*. Let G be a countable graph. Let $\Omega : V(G) \rightarrow \omega$ be a priority function with finite range. Let v_0 be a starting vertex. The two players move along the edges of the graph. On odd positions, player *Odd* moves, and on even positions, player *Even* moves.

If either player has no move, the other wins. Otherwise, the play is an infinite sequence of vertices $v_0, v_1, v_2, v_3 \dots$, and we say that player *Even* wins the play if the smallest number occurring infinitely often in the sequence $\Omega(v_0), \Omega(v_1), \Omega(v_2), \Omega(v_3) \dots$ is even. Otherwise, we say that player *Odd* wins.

A strategy \mathcal{S} of a player Pl is a function from finite sequences of vertices $v_0, v_1, v_2, v_3 \dots v_k$, where v_k is a Pl -vertex, to a successor of v_k . A strategy \mathcal{S} of Pl is *winning* if Pl wins all the play which respect \mathcal{S} .

Parity games can be encoded as Borel games in the sense of Descriptive Set Theory; so, by Martin's Borel Determinacy Theorem, see [20], parity games are *determined*: that is, there is always a player which has a winning strategy in the game.

A strategy \mathcal{S} of a player is called *positional* if $\mathcal{S}(v_1, v_2, v_3 \dots v_k)$ only depends on the last vertex played v_k .

Parity games are important because they enjoy the following very strong form of determinacy:

Lemma 3.1 (*positional determinacy, see [9]*) *If either player has a winning strategy in a parity game, then he has a positional winning strategy.*

Given that model checking and parity games are polynomial time equivalent, one is solvable in polynomial time if and only if the other is. And given the importance of μ -Calculus for system verification, the polynomial time solvability of these problems is a crucial problem in the area.

It is known that the two problems are in the complexity class UP (standing for Unique P), that is, the problems solvable in polynomial time by a nondeterministic Turing machine having at most one accepting computation on each input, see [15]. Note that UP is a subclass of NP , and a $co-UP$ bound follows by complementation.

Several algorithms have been proposed, starting from the first model checking algorithm of [10]; the working time of this algorithm is $O(m \cdot n^{d+1})$, where m is the size of ϕ , n is the size of G and d is the alternation depth of ϕ .

Subsequently, [19] improved the complexity of the Emerson-Lei algorithm to $O(m \cdot n^{\lceil d/2 \rceil + 1})$.

Then we have an algorithm which works “fast” on graphs of bounded tree width (see [21]). Recall that Courcelle’s theory of monadic second order logic [6] implies that on graphs of bounded tree-width k , the model checking problem can be solved in time linear in the size of the graph, that is, the time is $O(n)$. However, the constant hidden in the O (depending on the formula and on the tree width) is large according to Courcelle’s bound. [21] manages to reduce to time $O(n \cdot (km)^2 \cdot d^{2((k+1)m)^2})$, so a little more than exponential in d, k, m .

For the general case, the best we have so far is a subexponential algorithm (see [16]), and a general polynomial algorithm is actively searched.

4 Automata

4.1 Parity automata

Since Rabin automata were introduced in [22], tree automata have been studied as “dynamic” counterparts of various logics. For instance, parity automata are expressively equivalent to μ -Calculus formulas, and can be viewed as a “dynamic” normal form of the μ -Calculus.

There are several equivalent definitions for parity automata, especially differing in the transition function. We choose the following definition.

A *parity automaton* is a tuple $A = (Q, \Lambda, \delta, q_0, \Omega)$ where:

- Q is a finite set of states;
- $\Lambda = Powerset(Pred)$ is the alphabet, where $Pred$ is a finite set;
- $q_0 \in Q$ is the initial state;
- $\Omega : Q \rightarrow \omega$ is the priority function;
- $\delta : Q \times \Lambda \rightarrow Dc(Q)$ is the transition function, where $Dc(Q)$ is the set of all disjunctions of “cover” operators

$$cover(q_1, \dots, q_n) = \diamond q_1 \wedge \dots \wedge \diamond q_n \wedge \square(q_1 \vee \dots \vee q_n),$$

with $q_1, \dots, q_n \in Q$.

A semantic game (in fact a kind of parity game) can be defined from an automaton A and a countable, pointed, *Pred*-colored graph (G, Rsat, v_0) .

Let $V = V(G)$. For $v \in V$, let $\text{color}(v)$ the set of the elements $P \in \text{Pred}$ such that $P \text{ Rsat } v$. This gives a function $\text{color} : V \rightarrow \Lambda$.

The players are called *Duplicator* and *Spoiler*. Positions of the game are, alternately, elements of $Q \times V$ and subsets of $Q \times V$.

The initial position is (q_0, v_0) . On a position (q, v) , *Duplicator* moves by choosing a “marking” function m from $\text{Succ}(v)$ to $\text{Powerset}(Q)$ which, viewed as an interpretation for the atoms Q over the graph $\{v\} \cup \text{Succ}(v)$, satisfies the modal formula $\delta(q, \text{color}(v))$. *Spoiler* then moves by choosing a pair $(q', v') \in m$ with $v' \in \text{Succ}(v)$; the new position becomes (q', v') , and so on.

If ever some player has no moves, the other wins. Otherwise, we have an infinite sequence

$$(q_0, v_0), m_1, (q_1, v_1), m_2, (q_2, v_2), \dots,$$

and *Duplicator* wins if in the sequence $\Omega(q_0), \Omega(q_1), \Omega(q_2), \dots$, the least integer occurring infinitely often is even. Otherwise, *Spoiler* is the winner.

The automaton A *accepts* the graph G if *Duplicator* has a winning strategy in the game of A on G . The *language* defined by A is the set of all graphs accepted by A .

If q is a state of the automaton A , we denote by (A, q) the automaton like A except that the initial state is q .

Like in every two player game, if \mathcal{S} is a strategy of either player in an automaton game, the moves of \mathcal{S} can be organized in a tree, called the *strategy tree* of \mathcal{S} .

In particular, if \mathcal{S} is a strategy for *Duplicator* on a graph G , the strategy tree of \mathcal{S} can be represented as a labeled tree as follows. The nodes are all possible finite prefixes $(q_0, v_0)m_1(q_1, v_1)m_2 \dots (q_n, v_n)$ of a play (ending in a move of *Spoiler*) where *Duplicator* uses \mathcal{S} , with the node $(q_0, v_0)m_1(q_1, v_1)m_2 \dots (q_n, v_n)$ being a successor of the node $(q_0, v_0)m_1(q_1, v_1)m_2 \dots (q_{n-1}, v_{n-1})$. The *label* of the node

$$(q_0, v_0)m_1(q_1, v_1)m_2 \dots (q_n, v_n)$$

is the pair (q_n, v_n) .

Since the transition function are disjunctions of covers, it follows that if T is a strategy tree for *Duplicator* on a graph G , then the second (vertex) components of the labels of the nodes of T form a tree bisimilar to G .

In the following, and in particular in Section 5, we shall need more general automata, where, besides covers, among the disjuncts of the transition function $\delta(q, c)$ we may also find conjunctions of diamonds:

$$\diamond(q_1) \wedge \dots \wedge \diamond(q_n).$$

This kind of automata, however, can be simulated by “cover-automata”, in the following way.

Suppose A is such an automaton.

- First, add a new state q_t with $\Omega(q_t) = 0$ and $\delta(q_t, c) = \text{cover}(q_t) \vee \text{cover}(\emptyset)$ (notice that, starting from q_t , the new automaton accepts any graph).
- Then, substitute any disjunct having the form $\diamond q_1 \wedge \dots \wedge \diamond q_n$ with $\text{cover}(q_1, \dots, q_n, q_t)$.

The new automaton only uses disjunctions of “covers” in the transition function, and is equivalent to A.

Notice finally that the game of a parity automaton on a graph can be coded into a parity game, hence parity automata enjoy positional determinacy by Lemma 3.1. This is a good reason to choose parity automata rather than other, expressively equivalent kinds of automata.

4.2 Weak parity automata

A parity automaton is called *weak* if for every $(q, \lambda) \in Q \times \Lambda$ and every state q' occurring in $\delta(q, \lambda)$, one has $\Omega(q') \leq \Omega(q)$. So, along every transition, the priority does not increase. This implies that in every infinite play, the priority is eventually constant, and *Duplicator* wins if and only if this eventual priority is even.

Weak parity automata are expressively equivalent, on arbitrary graphs, to compositions of Σ_1 and Π_1 formulas of the μ -Calculus.

4.3 Büchi automata

A *Büchi automaton* is a parity automaton where $\Omega : Q \rightarrow \{0, 1\}$. When talking about Büchi automata, one calls *final* a state q such that $\Omega(q) = 0$. Then *Duplicator* wins an infinite play if and only if the play visits final states infinitely often.

Note that a Büchi automaton with conjunctions of diamonds is equivalent to a cover Büchi automaton, because adding a state q_t with $\Omega(q_t) = 0$ to a Büchi automaton produces an automaton of the same class.

In the μ -Calculus fixpoint hierarchy, Büchi automata coincide with the class Π_2 .

4.4 coBüchi automata

The dual of Büchi automata are co-Büchi automata.

A *coBüchi automaton* is a parity automaton where $\Omega : Q \rightarrow \{1, 2\}$. When talking about coBüchi automata, one calls *final* a state q such that $\Omega(q) = 2$. Then *Duplicator* wins an infinite play if and only if the play visits final states always except for a finite number of times.

In the μ -Calculus fixpoint hierarchy, coBüchi automata coincide with the class Σ_2 .

5 The upper bound

Theorem 5.1 *For every k , every Büchi automaton is equivalent in $SCCk$ to a coBüchi automaton.*

Proof: let B be a Büchi automaton. Let Q be the set of states of B . By Lemma 3.1, if *Duplicator* has a winning strategy for B in a graph G of class $SCCk$, then he or she has a positional winning strategy, call it \mathcal{S}_p .

Let π be an infinite play of \mathcal{S}_p . Then π must have, from a certain point on, at least a final state every $|Q|k$ moves. In fact, if this were not true, then π would have infinitely many nonfinal subsequences of size $|Q|k + 1$. Since G is finite, π eventually enters some s.c.c. S where it remains forever. If we take $|Q|k + 1$ consecutive nonfinal moves in S , then since S has at most k elements, by the pigeonhole

principle there is a repeated pair $(q, v), \dots, (q, v)$ among these moves. Now if *Spoiler* repeats the moves he or she did between the two equal pairs above, *Duplicator* is also forced (in \mathcal{S}_p) to repeat his or her moves, because \mathcal{S}_p is positional. So, \mathcal{S}_p has an infinite play with only finitely many nonfinal states, contrary to the fact that \mathcal{S}_p is winning for *Duplicator* in the Büchi automaton B .

Summing up, if *Duplicator* manages to have infinitely many final states in a play, then he or she manages to have final states at most every $|Q|k$ moves, from a certain moment on. This corresponds to the coBüchi automaton C which we are going to define.

The idea is to play B and to memorize the last $|Q|k$ states of the play.

The alphabet of C will be the same of B .

The states of C will be the nonempty lists of states of B with length at most $|Q|k$.

The initial state of C is the list of length one consisting of the initial state of B .

The final states of C will be the lists of length $|Q|k$ containing at least one final state of B .

Finally, the transition function δ_C of C will mimic the function δ_B of B while memorizing the last $|Q|k$ states. Formally, we say that a marking m satisfies $\delta_C(L, \gamma)$ if verifies a formula of the kind

$$\text{cover}(L' q_1, \dots, L' q_n),$$

where:

- $\text{cover}(q_1, \dots, q_n)$ is a disjunct of $\delta_B(\text{last}(L), \gamma)$, and
- $L' = L$ if L has length less than $|Q|k$, and L' is L minus the first element otherwise.

Now if B accepts a graph G then, as we have seen, there is a winning strategy \mathcal{S}_p of *Duplicator* where finals repeat every $|Q|k$ times from a certain point on. So, C also accepts G , with the strategy consisting of playing \mathcal{S}_p , and memorizing the last $|Q|k$ states of the play.

Conversely, if C accepts a graph G , via any winning strategy \mathcal{S}' of *Duplicator*, then in \mathcal{S}' , final states of B occur infinitely often in every infinite play, so B also accepts G with the strategy consisting of taking the last components of the lists of \mathcal{S}' .

So, the automata B and C are equivalent.

Q.E.D.

Corollary 5.1 *For every $k \geq 1$, the μ -Calculus collapses in $\text{SCC}k$ to $\Delta_2 = \Sigma_2 \cap \Pi_2$.*

Proof: we show by induction on $n \geq 2$ that Σ_n and Π_n collapse to Δ_2 . For $n = 2$, Σ_2 is included in Π_2 , so Σ_2 is included in Δ_2 . Π_2 is analogous.

For $n \geq 2$, consider Σ_{n+1} . This class is the closure of $\Sigma_n \cup \Pi_n$ with respect to composition and μ ; by inductive hypothesis, $\Sigma_n \cup \Pi_n$ coincide with Σ_2 , so Σ_{n+1} is the closure of Σ_2 with respect to composition and μ , that is, Σ_{n+1} coincides with Σ_2 , hence it collapses to Δ_2 .

Likewise, consider Π_{n+1} . This class is the closure of $\Sigma_n \cup \Pi_n$ with respect to composition and ν ; by inductive hypothesis, $\Sigma_n \cup \Pi_n$ coincide with Π_2 , so Π_{n+1} is the closure of Π_2 with respect to composition and ν , that is, Σ_{n+1} coincides with Π_2 , hence it collapses to Δ_2 .

Q.E.D.

6 The lower bound

Theorem 6.1 *There is a Büchi automaton which is not equivalent in SCC1 to any weak parity automaton.*

Proof: the proof needs some definitions and lemmas.

Definition 6.1 *Let F be a predicate (standing for final). Let (G, v_0) be a pointed, F -colored graph. This means that each vertex can satisfy F (in which case we call it an F -vertex) or not (in which case we call it N -vertex, N standing for nonfinal).*

We define the following (parity-like) game $\Gamma(G, v_0)$ on G . Call PN and PF two players. The positions are the vertices of G . The initial position is v_0 . On N vertices, player PN moves along one edge. On F vertices, likewise, player PF moves along one edge.

If either player has no move, the other wins. Otherwise, the play is infinite, and player PN wins if the play visits F vertices infinitely often, and player PF wins otherwise (this interchange between players PN and PF in the definition of the winning condition seems to be necessary for the argument to work).

For convenience, let us say that a graph (G, v_0) verifies property Γ if and only if player PN has a winning strategy in the game $\Gamma(G, v_0)$.

Lemma 6.1 *The property Γ is Büchi-expressible.*

Proof: consider the following Büchi automaton B_Γ .

The only predicate is F , whose negation we denote by N .

There are two states q_N and q_F plus an initial state q_0 .

We decree that q_F is final and q_N is nonfinal (the priority of q_0 is irrelevant, let us decide that q_0 is final).

Finally, the transition function δ_Γ of B_Γ is the following:

- $\delta_\Gamma(q_0, N) = \delta_\Gamma(q_N, N) = (\Diamond q_N) \vee (\Diamond q_F);$
- $\delta_\Gamma(q_0, F) = \delta_\Gamma(q_F, F) = \Box(q_N \vee q_F);$
- $\delta_\Gamma(q_F, N) = \delta_\Gamma(q_N, F) = \text{false}$ (the empty disjunction).

Note that B_Γ is equivalent to Γ .

In fact, every winning strategy \mathcal{S} for *Duplicator* in the automaton B_Γ in a graph G can be translated into a winning strategy \mathcal{S}' for player PN in $\Gamma(G)$, which consists in choosing any successor of the current vertex which is marked q_N or q_F in \mathcal{S} (assuming that this current vertex is an N vertex).

Conversely, we translate a strategy \mathcal{S}' winning for player PN in $\Gamma(G)$ into a strategy \mathcal{S} winning for *Duplicator* in B_Γ , as follows.

In a position (q_0, v) or (q_N, v) , where v is an N -vertex, *Duplicator* takes the vertex v' chosen by \mathcal{S}' and marks it with q_N , if v' is a N vertex, and with q_F , if it is an F vertex.

In a position (q_0, v) or (q_F, v) , where v is an F -vertex, *Duplicator* marks all successors of v : with q_N , if the successor is a N vertex, and with q_F , if it is an F vertex (notice that, following the strategy \mathcal{S}' , a play will never reach a position of type (q_N, v) for an F -vertex, or (q_F, v) , for an N -vertex).

Q.E.D

Corollary 6.1 *The property $\Box^*\Gamma$ is Büchi expressible.*

Proof: $\square^*\Gamma$ is the composition of the Büchi (hence Π_2) property Γ and of the Π_1 (hence Π_2) formula \square^*P , where P is an atomic proposition. Since Π_2 is stable under composition, the property in question is Π_2 , or equivalently, is Büchi expressible.

Q.E.D.

Now we show that there is no weak parity automaton equivalent to $\square^*\Gamma$ in $SCC1$.

Definition 6.2 Let W be an automaton. A state q of W is $\square^*\Gamma$ -winning if for every graph G belonging to $SCC1$, if (W, q) accepts G , then G verifies $\square^*\Gamma$.

A graph G witnesses against q if G belongs to $SCC1$, (W, q) accepts G but G does not verify $\square^*\Gamma$ (so q is $\square^*\Gamma$ -winning if and only if there are no witnesses against q).

Definition 6.3 A finite pseudotree is a finite graph obtained from a finite tree by adding loops to some nodes.

Lemma 6.2 Every graph G belonging to $SCC1$ is bisimilar to a finite pseudotree.

Proof: let G_1 the graph G where the loops have been removed. Let G_2 be the unfolding of G , which is a finite tree. Let H be the graph resulting from G_2 by attaching a loop to any bisimilar copy of a vertex of G having a loop. H is bisimilar to G and is a finite pseudotree.

Q.E.D.

Lemma 6.3 Suppose W is a (weak) automaton such that the initial state q_0 is $\square^*\Gamma$ -winning.

If G is a finite pseudotree, and T is a winning strategy tree of Duplicator for W on G , then all the states q belonging to a label of T are $\square^*\Gamma$ -winning.

Proof: Suppose by way of a contradiction that T is a winning strategy tree of Duplicator for W on G , but there exists a node $t \in T$ with label (q, v) , such that q is not $\square^*\Gamma$ -winning.

This means that there exists a graph G_q in $SCC1$ which is accepted by (W, q) such that $\square^*\Gamma$ is false in G_q . Let T_q be a winning strategy tree for (W, q) on G_q .

Consider the tree T' which is obtained from T by substituting the subtree rooted in t with T_q .

Claim 6.1 T' is a strategy tree for (W, q_0) on a finite $SCC1$ -graph G' containing a reachable node g such that (G, g) is isomorphic to G_q .

Proof: letting m be the height of t in T , do the following:

- Replace the subtree of T rooted in t with G_q , and
- for any node $s \neq t$ of T with height m , consider its label (q_s, v_s) and replace the subtree of T rooted in s with the subgraph of G consisting of the descendants of v_s .

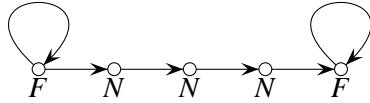
The resulting graph G' is a graph belonging to $SCC1$ containing a reachable node g such that (G, g) is isomorphic to G_q .

Moreover, T' is a winning strategy tree for (W, q_0) on G' . This proves the claim.

Q.E.D.

From the claim, we get a contradiction: by hypothesis, (W, q_0) is equivalent to $\square^*\Gamma$, and by the claim, (W, q_0) accepts G' ; on the other hand, $\square^*\Gamma$ is false in G' . This proves the lemma.

Q.E.D.

Figure 1: The graph G_1

Definition 6.4 For all natural numbers k , let $n = 2^k + 1$, and let G_k be the graph having as set of nodes the set

$$\{v_i, v_{i,1}, v_{i,2}, \dots, v_{i,n} : 0 \leq i < k\} \cup \{v_k\},$$

where, for $0 \leq i \leq k$, the nodes v_i are reflexive F nodes, while the $v_{i,j}$'s are irreflexive nodes satisfying N .

Moreover, if $i < k$ the graph G_k has arches $(v_i, v_{i,1}), (v_{i,1}, v_{i,2}), \dots, (v_{i,n-1}, v_{i,n}), (v_{i,n}, v_{i+1})$.

The root is v_0 .

Note that all graphs G_k are pseudotrees and satisfy $\square^*\Gamma$.

Before passing to the next lemma, let us define N_{loop} to be the graph consisting of one reflexive N node.

Lemma 6.4 For all h and k with $h \leq k$, there exists no weak automaton W with h states having a positional winning strategy tree T of Duplicator on G_k , where the initial state q_0 is $\square^*\Gamma$ -winning.

Proof: By induction on k .

Let $k = 1$. Then W has only one state q_0 , and there exists a winning strategy tree T for Duplicator on G_1 decorated only by q_0 where q_0 is $\square^*\Gamma$ -winning, then $\Omega(q_0)$ is even and $Cover(q_0)$ should be a disjunct of $\delta(q_0, N)$.

But then W would accept N_{loop} , which does not verify $\square^*\Gamma$, and q_0 would not be $\square^*\Gamma$ -winning, contrary to the hypotheses.

Let $k > 1$. Suppose there are W and T such that W has h states with $h \leq k$ and T is a winning strategy tree for Duplicator in the W -game on G_k , where the initial state q_0 is $\square^*\Gamma$ -winning. By Lemma 6.3, we know that all states appearing as labels in T are $\square^*\Gamma$ -winning.

Claim 6.2 There is a node $t \in T$, labeled by (q, v_0) or $(q, v_{0,i})$ for some i , with $\Omega(q) < \Omega(q_0)$.

Proof: Suppose, by way of a contradiction, that all labels (q, v_0) or $(q, v_{0,i})$ in T have the same priority $\Omega(q) = \Omega(q_0)$. First, $\Omega(q_0)$ is even because

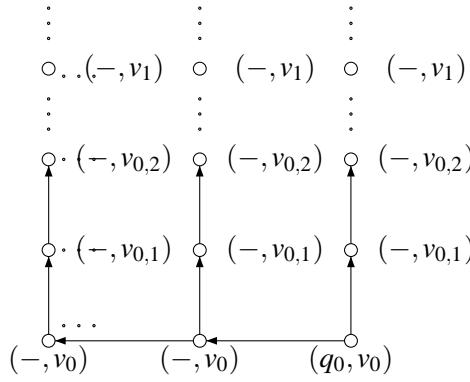


Figure 2: We show the tree T , but for any edge (v, v') in G_k and t in T labeled $(-, v)$ we only draw one successor of t labeled $(-, v')$, whereas there could be many of them.

- there exists an infinite path in T corresponding to the same node v_0 , labeled with states having the same priority of q_0 , and
- T is winning for *Duplicator* in W , which is a weak automaton.

For $i \leq n$ (where $n = 2^k + 1$), let

$$Q_i = \{q \in Q : \exists t \in T \text{ labeled by } (q, v_{0,i})\},$$

where Q is the set of states of W .

Since all Q_i are nonempty subsets of Q which has h elements, and since $2^h < n$, by the pigeonhole principle there must be two levels $i < i+j \leq n$ with $Q_i = Q_{i+j}$. Fix $q^* \in Q_i$: we next prove that (W, q^*) accepts N_{loop} by constructing a winning strategy tree $T_{q^*}^\infty$ for (W, q^*) on N_{loop} as follows.

For any $q \in Q_i$, consider a node $t \in T$ labelled by $(q, v_{0,i})$ and the subtree T_q of T rooted in t (since we suppose that the strategy for *Duplicator* is positional, this tree does not depend on t , but only on q). Erase from T_q all nodes of height greater than j . In this way the leaves of the remaining tree are labeled by pairs $(q', v_{0,i+j})$, for some $q' \in Q_{i+j} = Q_i$. Change the second component of the labels of all nodes of the resulting tree to $v_{0,i}$, and call $T_q^{<j}$ the resulting labeled finite tree.

Now consider the fixed state $q^* \in Q_i$. We define inductively a sequence $T_{q^*}^m, m = 1, 2, 3, \dots$ of finite trees.

- Initially let $T_{q^*}^1 = T_{q^*}^{<j}$;
- inductively, given $T_{q^*}^m$, for all $q \in Q_{i+j} = Q_i$, attach to all leaves of $T_{q^*}^m$ labelled by $(q, v_{0,i})$, a copy of $T_q^{<j}$; call $T_{q^*}^{m+1}$ the result.

The sequence of finite trees $T_{q^*}^m$ converges, in the usual sense, to an infinite tree. Let $T_{q^*}^\infty$ be the limit. Notice that $T_{q^*}^\infty$ is a winning strategy tree of *Duplicator* for the automaton (W, q^*) on the graph N_{loop} .

Having proved that (W, q^*) accepts N_{loop} , we get a contradiction: N_{loop} is a witness against q^* , hence q^* should not be $\square^*\Gamma$ winning, in contradiction with Lemma 6.3. This proves the claim.

Q.E.D.

Using the claim above, we see that there is a node t in T labeled by (q_1, v_1) for some $\square^*\Gamma$ -winning state q_1 with $\Omega(q_1) < \Omega(q_0)$ (just follow a path from a node labeled (q, v_0) or $(q, v_{0,i})$ with $\Omega(q) < \Omega(q_0)$ to a node labeled (q_1, v_1)).

Consider now the automaton W' which is W restricted to states which are “reachable” using the W -transition function from q_1 . We have:

- if h' is the number of states of W' , then $h' < h$, since $\Omega(q_0) > \Omega(q_1)$ and W is a weak automaton;
- the subtree T' of T rooted at t is a winning strategy tree for W' on the graph G_{k-1} , and from $h' < h \leq k$ it follows $h' \leq k - 1$;
- (W', q_1) is equivalent to (W, q_1) , hence, q_1 is a $\square^*\Gamma$ -winning state for W' as well.

From the points above, we easily obtain a contradiction by induction. (As a final remark, notice that the positionality hypothesis is not necessary, but has been added to simplify the inductive step). This proves the lemma.

Q.E.D.

Now the proof of Theorem 6.1 is concluded as follows.

Suppose for an absurdity that there is a weak automaton W equivalent to $\square^*\Gamma$ in $SCC1$. Let k be the number of states of W . Consider the graph G_k . Then W accepts G_k , so there is a winning strategy tree T of *Duplicator* for W on G_k , and the initial state of T is $\square^*\Gamma$ -winning. But this is in contrast with Lemma 6.4. So W cannot exist.

Q.E.D.

Corollary 6.2 *Over the class $SCC1$ (hence also over $SCCk$ for any $k > 1$) we have:*

- $\Delta_2 \neq Comp(\Sigma_1, \Pi_1)$;
- *the μ -Calculus does not collapse to $Comp(\Sigma_1, \Pi_1)$.*

7 Conclusions and future work

Let us mention a couple of applications of our results. The first application (of Section 5) is to the model checking problem:

Corollary 7.1 *For every k , the μ -Calculus model checking problem for a fixed formula ϕ is quadratic (i.e. $O(n^2)$) for graphs of class $SCCk$.*

Proof: the algorithm consists in first translating ϕ into a Büchi automaton (which takes a time depending on ϕ and k but not on the graph), and then applying the algorithm of [19] with $d = 2$.

Q.E.D.

As a second application (of Section 6 this time) let us consider tree width:

Corollary 7.2 *On the class $TW1$, the μ -Calculus does not collapse to $Comp(\Sigma_1, \Pi_1)$.*

Proof: Suppose for an absurdity that $\mu = Comp(\Sigma_1, \Pi_1)$ on $TW1$. Now every pseudotree has an underlying undirected graph of tree width one. Then $\mu = Comp(\Sigma_1, \Pi_1)$ on pseudotrees. By Lemma 6.2, every finite graph belonging to $SCC1$ is bisimilar to a finite pseudotree. So, by our hypothesis and by invariance of the μ -Calculus under bisimulation, we would have that $\mu = Comp(\Sigma_1, \Pi_1)$ on $SCC1$. But this is in contrast with Corollary 6.2.

Q.E.D.

By the previous corollary, we have a lower bound on the μ -calculus hierarchy on the class $TW1$, hence also on the larger classes TWk for every $k > 1$. It would be interesting to come up with an *upper* bound on TWk as well, and more generally, to investigate the expressiveness of μ -Calculus on classes given by other, algorithmically interesting graph-theoretic measures (e.g. cliquewidth, DAG-width, etc.) This will be the subject of future papers.

Acknowledgments

The work has been partially supported by the PRIN project *Innovative and multi-disciplinary approaches for constraint and preference reasoning* and the GNCS project *Logics, automata, and games for the formal verification of complex systems*.

References

- [1] L. Alberucci and A. Facchini, The Modal mu-Calculus Hierarchy over Restricted Classes of Transition Systems, *Journal of Symbolic Logic*, (4) 74 (2009), 1367–1400.
- [2] L. Alberucci and A. Facchini, On Modal μ -Calculus and Gödel-Löb Logic, *Studia Logica* 91 (2009), 145–169.
- [3] A. Arnold, The mu-Calculus Alternation-Depth Hierarchy is Strict on Binary Trees. *ITA* (33) 4/5 (1999), 329–340.
- [4] J. C. Bradfield, The Modal mu-Calculus Alternation Hierarchy is Strict, *Proceedings of CONCUR 1996*, 233–246.
- [5] E. M. Clarke, O. Grumberg and D. A. Peled, *Model Checking*, MIT Press, 1999.
- [6] B. Courcelle, Graph rewriting: An algebraic and logic approach, In: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B: Formal Models and Semantics*, Elsevier, Amsterdam, 1990, 193–242.
- [7] G. D'Agostino and G. Lenzi, On the μ -Calculus over Transitive and Finite Transitive Frames, submitted.
- [8] A. Dawar and M. Otto, Modal Characterisation Theorems over Special Classes of Frames, *Annals of Pure and Applied Logic* 161 (2009), 1–42.
- [9] E. A. Emerson and C. S. Jutla, Tree Automata, Mu-Calculus and Determinacy, *IEEE Proc. Foundations of Computer Science* (1991), 368–377.
- [10] E. A. Emerson and C. L. Lei. Efficient Model Checking in Fragments of the Propositional μ -Calculus. In: *Symposium on Logic in Computer Science*, pages 267–278. IEEE Computer Society Press, June 1986.
- [11] J. Flum, M. Grohe, *Parameterized Complexity Theory*, Springer, 2006.
- [12] R. Halin, S-Functions for Graphs, *J. Geometry* 8 (1976), 171–186.
- [13] D. Janin, I. Walukiewicz, On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic, *CONCUR 1996*, 263–277.
- [14] T. Johnson, N. Robertson, P. D. Seymour and R. Thomas, Directed Tree-Width, *J. Combin. Theory Ser. B* 82 (2001), 138–155.
- [15] M. Jurdziński, Deciding the Winner in Parity Games Is in $UP \cap co-UP$, *Information Processing Letters* (68) 3 (1998), 119–124.
- [16] M. Jurdziński, M. Paterson, U. Zwick, A Deterministic Subexponential Algorithm for Solving Parity Games, *SIAM J. Comput.* (4) 38 (2008), 1519–1532.

- [17] D. Kozen, Results on the Propositional μ -Calculus. *Theor. Comput. Sci.* 27 (1983), 333–354.
- [18] O. Kupferman, M. Y. Vardi: $\Pi_2 \cap \Sigma_2 \equiv AFMC$. Proceedings of ICALP 2003, 697–713.
- [19] D. E. Long, A. Browne, E. M. Clarke, S. Jha, and W. R. Marrero, An Improved Algorithm for the Evaluation of Fixpoint Expressions, In CAV '94, volume 818 of LNCS, Springer-Verlag, 1994, 338–350.
- [20] D. Martin, Borel Determinacy, *Annals of Mathematics. Second series* (2) 102 (1975), 363–371.
- [21] J. Obrdžálek, Fast Mu-Calculus Model Checking when Tree-Width Is Bounded, Proceedings of CAV 2003, 80–92.
- [22] M. Rabin, Decidability of Second-Order Theories and Automata on Infinite Trees, *Transactions of the American Mathematical Society* 141 (1969), 1–35.
- [23] N. Robertson and P. Seymour, Graph Minors III: Planar Tree-Width, *Journal of Combinatorial Theory, Series B*, vol. 36 (1984), 49–64.
- [24] N. Robertson and P. D. Seymour, Graph Minors. V. Excluding a Planar Graph, *J. Combin. Theory Ser. B* 41 (1986), 92–114.
- [25] C. Smoryński, *Self-reference and Modal Logic*, Springer, 1985.
- [26] J. van Benthem, *Modal Correspondence Theory*, Ph.D. Thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.
- [27] J. Van Benthem, Modal Frame Correspondences and Fixed Points, *Studia Logica* 83 (2006), 133–155.
- [28] A. Visser, Löb's Logic Meets the μ -Calculus, in: A. Middeldorp, V. van Oostrom, F. van Raamsdonk and R. de Vrijer (eds.), *Processes, Terms and Cycles, Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop on the Occasion of his 60th Birthday*, Springer, 2005, 14–25.

Athina P. Petropulu. "Higher-Order Spectral Analysis."
2000 CRC Press LLC. <<http://www.engnetbase.com>>.

Higher-Order Spectral Analysis

Athina P. Petropulu
Drexel University

76.1	Introduction
76.2	Definitions and Properties of HOS
76.3	HOS Computation from Real Data
76.4	Linear Processes Nonparametric Methods • Parametric Methods
76.5	Nonlinear Processes
76.6	Applications/Software Available
	Acknowledgments
	References

76.1 Introduction

The past 20 years witnessed an expansion of power spectrum estimation techniques, which have proved essential in many applications, such as communications, sonar, radar, speech/image processing, geophysics, and biomedical signal processing [13, 11, 7]. In power spectrum estimation the process under consideration is treated as a superposition of statistically uncorrelated harmonic components. The distribution of power among these frequency components is the power spectrum. As such, phase relations between frequency components are suppressed. The information in the power spectrum is essentially present in the autocorrelation sequence, which would suffice for the complete statistical description of a Gaussian process of known mean. However, there are applications where one would need to obtain information regarding deviations from the Gaussianity assumption and presence of nonlinearities. In these cases power spectrum is of little help, and one would have to look beyond the power spectrum or autocorrelation domain. Higher-Order Spectra (HOS) (of order greater than 2), which are defined in terms of higher-order cumulants of the data, do contain such information [16]. The third-order spectrum is commonly referred to as bispectrum, the fourth-order one as trispectrum, and in fact, the power spectrum is also a member of the higher-order spectral class; it is the second-order spectrum.

HOS consist of higher-order moment spectra, which are defined for deterministic signals, and cumulant spectra, which are defined for random processes. In general, there are three motivations behind the use of HOS in signal processing: (1) to suppress Gaussian noise of unknown mean and variance; (2) to reconstruct the phase as well as the magnitude response of signals or systems; and (3) to detect and characterize nonlinearities in the data.

The first motivation stems from the property of Gaussian processes to have zero higher-order spectra. Due to this property, HOS are high signal-to-noise ratio domains, in which one can perform detection, parameter estimation, or even signal reconstruction even if the time domain noise is spatially correlated. The same property of cumulant spectra can provide means of detecting and characterizing deviations of the data from the Gaussian model.

The second motivation is based on the ability of cumulant spectra to preserve the Fourier-phase of signals. In the modeling of time series, second-order statistics (autocorrelation) have been heavily used because they are the result of least-squares optimization criteria. However, an accurate phase reconstruction in the autocorrelation domain can be achieved only if the signal is minimum phase. Nonminimum phase signal reconstruction can be achieved only in the HOS domain, due to the HOS ability to preserve phase. Figure 76.1 shows two signals, a nonminimum phase and a minimum phase, with identical magnitude spectra but different phase spectra. Although power spectrum cannot distinguish between the two signals, the bispectrum that uses phase information can.

Being nonlinear functions of the data, HOS are quite natural tools in the analysis of nonlinear systems operating under a random input. General relations for arbitrary stationary random data passing through an arbitrary linear system exist and have been studied extensively. Such expression, however, are not available for nonlinear systems, where each type of nonlinearity must be studied separately. Higher-order correlations between input and output can detect and characterize certain nonlinearities [34], and for this purpose several higher-order spectra-based methods have been developed.

The organization of this chapter is as follows. First the definitions and properties of cumulants and higher-order spectra are introduced. Then two methods for the estimation of HOS from finite length data are outlined and the asymptotic statistics of the obtained estimates are presented. Following that, parametric and nonparametric methods for HOS-based identification of linear systems are described, and the use of HOS in the identification of some particular nonlinear systems is briefly discussed. The chapter concludes with a section on applications of HOS and available software.

76.2 Definitions and Properties of HOS

In this chapter we will consider random one-dimensional processes only. The definitions can be easily extended to the two-dimensional case [15].

The joint moments of order r of the random variables x_1, \dots, x_n are given by [22]

$$\begin{aligned} M\text{ om}\left[x_1^{k_1}, \dots, x_n^{k_n}\right] &= E\{x_1^{k_1}, \dots, x_n^{k_n}\} \\ &= (-j)^r \frac{\partial^r \Phi(\omega_1, \dots, \omega_n)}{\partial \omega_1^{k_1} \dots \partial \omega_n^{k_n}}|_{\omega_1=\dots=\omega_n=0}, \end{aligned} \quad (76.1)$$

where $k_1 + \dots + k_n = r$, and $\Phi()$ is their joint characteristic function. The joint cumulants are defined as

$$C\text{ um}[x_1^{k_1}, \dots, x_n^{k_n}] = (-j)^r \frac{\partial^r \ln \Phi(\omega_1, \dots, \omega_n)}{\partial \omega_1^{k_1} \dots \partial \omega_n^{k_n}}|_{\omega_1=\dots=\omega_n=0}. \quad (76.2)$$

For a stationary discrete time random process $X(k)$, (k denotes discrete time), the *moments* of order n are given by

$$m_n^x(\tau_1, \tau_2, \dots, \tau_{n-1}) = E\{X(k)X(k+\tau_1)\dots X(k+\tau_{n-1})\}, \quad (76.3)$$

where $E\{.\}$ denotes expectation. The n th order *cumulants* are functions of the moments of order up to n , i.e.,

1st order cumulants:

$$c_1^x = m_1^x = E\{X(k)\} \quad (\text{mean}) \quad (76.4)$$

2nd order cumulants:

$$c_2^x(\tau_1) = m_2^x(\tau_1) - (m_1^x)^2 \quad (\text{covariance}) \quad (76.5)$$

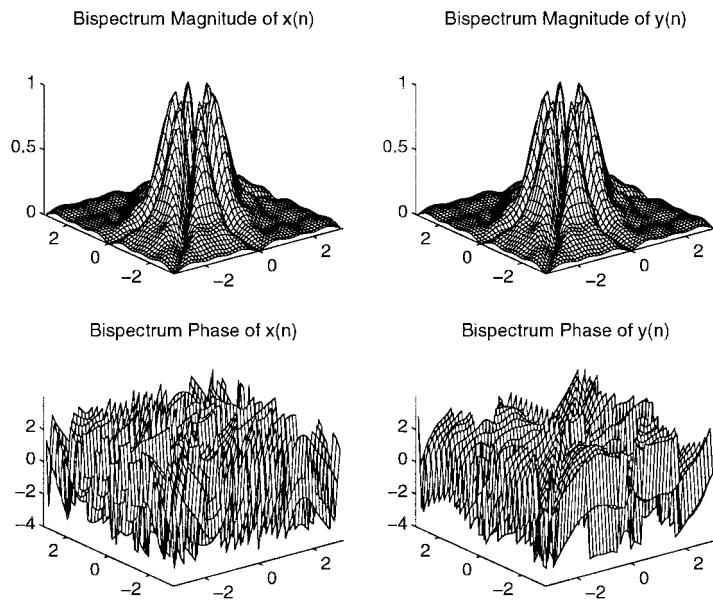
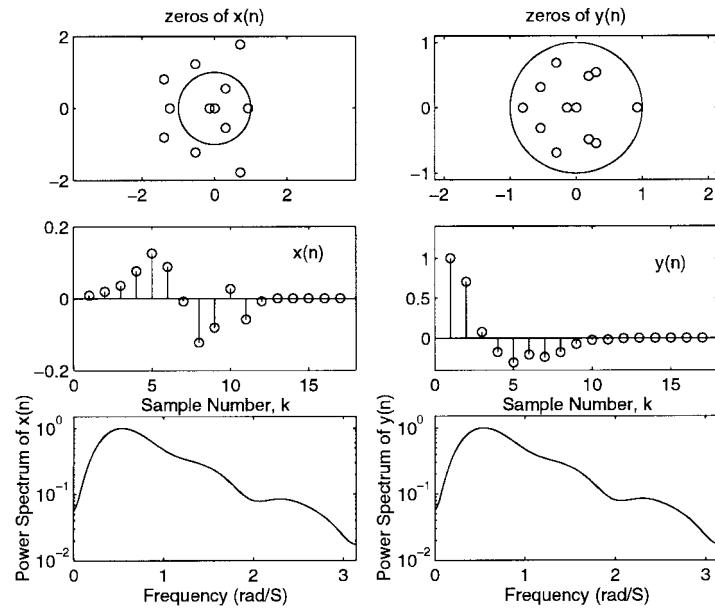


FIGURE 76.1: $x(n)$ is a nonminimum phase signal and $y(n)$ is a minimum phase one. Although their power spectra are identical, their bispectra are different because they contain phase information.

3rd order cumulants:

$$c_3^x(\tau_1, \tau_2) = m_3^x(\tau_1, \tau_2) - (m_1^x)[m_2^x(\tau_1) + m_2^x(\tau_2) + m_2^x(\tau_2 - \tau_1)] + 2(m_1^x)^3 \quad (76.6)$$

4th order cumulants:

$$\begin{aligned} c_4^x(\tau_1, \tau_2, \tau_3) &= m_4^x(\tau_1, \tau_2, \tau_3) - m_2^x(\tau_1)m_2^x(\tau_3 - \tau_2) - m_2^x(\tau_2)m_2^x(\tau_3 - \tau_1) \\ &\quad - m_2^x(\tau_3)m_2^x(\tau_2 - \tau_1) \\ &\quad - m_1^x[m_3^x(\tau_2 - \tau_1, \tau_3 - \tau_1) + m_3^x(\tau_2, \tau_3) + m_3^x(\tau_2, \tau_4) + m_3^x(\tau_1, \tau_2)] \\ &\quad + (m_1^x)^2[m_2^x(\tau_1) + m_2^x(\tau_2) + m_2^x(\tau_3) + m_2^x(\tau_3 - \tau_1) + m_2^x(\tau_3 - \tau_2) \\ &\quad + m_1^x(\tau_2 - \tau_1)] - 6(m_1^x)^4 \end{aligned} \quad (76.7)$$

where $m_3^x(\tau_1, \tau_2)$ is the 3rd order moment sequence, and m_1^x is the mean. The general relationship between cumulants and moments can be found in [16].

Some important properties of moments and cumulants are summarized next.

[P1] If $X(k)$ is Gaussian, the $c_n^x(\tau_1, \tau_2, \dots, \tau_{n-1}) = 0$ for $n > 2$. In other words, all the information about a Gaussian process is contained in its first and second-order cumulants. This property can be used to suppress Gaussian noise, or as a measure for non-Gaussianity in time series.

[P2] If $X(k)$ is symmetrically distributed, then $c_3^x(\tau_1, \tau_2) = 0$. Third-order cumulants suppress not only Gaussian processes, but also all symmetrically distributed processes, such as uniform, Laplace, and Bernoulli-Gaussian.

[P3] For cumulants additivity holds. If $X(k) = S(k) + W(k)$, where $S(k)$, $W(k)$ are stationary and statistically independent random processes, then $c_n^x(\tau_1, \tau_2, \dots, \tau_{n-1}) = c_n^s(\tau_1, \tau_2, \dots, \tau_{n-1}) + c_n^w(\tau_1, \tau_2, \dots, \tau_{n-1})$. It is important to note that additivity does not hold for moments.

If $W(k)$ is Gaussian representing noise which corrupts the signal of interest, $S(k)$, then by means of (P2) and (P3), we get that $c_n^x(\tau_1, \tau_2, \dots, \tau_{n-1}) = c_n^s(\tau_1, \tau_2, \dots, \tau_{n-1})$, for $n > 2$. In other words, in higher-order cumulant domains the signal of interest propagates noise free. Property (P3) can also provide a measure of statistical dependence of two processes.

[P4] if $X(k)$ has zero mean, then $c_n^x(\tau_1, \dots, \tau_{n-1}) = m_n^x(\tau_1, \dots, \tau_{n-1})$, for $n \leq 3$.

Higher-order spectra are defined in terms of either cumulants (e.g., cumulant spectra) or moments (e.g., moment spectra).

Assuming that the n th order cumulant sequence is absolutely summable, the n th order **cumulant spectrum** of $X(k)$, $C_n^x(\omega_1, \omega_2, \dots, \omega_{n-1})$, exists, and is defined to be the $(n-1)$ -dimensional Fourier transform of the n th order cumulant sequence. In general, $C_n^x(\omega_1, \omega_2, \dots, \omega_{n-1})$ is complex, i.e., it has magnitude and phase. In an analogous manner, **moment spectrum** is the multi-dimensional Fourier transform of the moment sequence.

If $v(k)$ is a stationary non-Gaussian process with zero mean and n th order cumulant sequence

$$c_n^v(\tau_1, \dots, \tau_{n-1}) = \gamma_n^v \delta(\tau_1, \dots, \tau_{n-1}), \quad (76.8)$$

where $\delta(\cdot)$ is the delta function, $v(k)$ is said to be n th order white. Its n th order cumulant spectrum is then flat and equal to γ_n^v .

Cumulant spectra are more useful in processing random signals than moment spectra since they possess properties that the moment spectra do not share: (1) the cumulants of the sum of two independent random processes equals the sum of the cumulants of the process; (2) cumulant spectra of order > 2 are zero if the underlying process is Gaussian; (3) cumulants quantify the degree of statistical dependence of time series; and (4) cumulants of higher-order white noise are multidimensional impulses, and the corresponding cumulant spectra are flat.

76.3 HOS Computation from Real Data

The definitions of cumulants presented in the previous section are based on expectation operations, and they assume infinite length data. In practice we always deal with data of finite length; therefore, the cumulants can only be approximated. Two methods for cumulants and spectra estimation are presented next for the third-order case.

Indirect Method:

Let $X(k), k = 1, \dots, N$ be the available data.

1. Segment the data into K records of M samples each. Let $X^i(k), k = 1, \dots, M$, represent the i th record.
2. Subtract the mean of each record.
3. Estimate the moments of each segments $X^i(k)$ as follows:

$$m_3^{x_i}(\tau_1, \tau_2) = \frac{1}{M} \sum_{l=l_1}^{l_2} X^i(l) X^i(l + \tau_1) X^i(l + \tau_2) ,$$

$$\begin{aligned} l_1 &= \max(0, -\tau_1, -\tau_2), \quad l_2 = \min(M-1, M-2), \\ |\tau_1| &< L, \quad |\tau_2| < L, \quad i = 1, 2, \dots, K . \end{aligned} \quad (76.9)$$

Since each segment has zero mean, its third-order moments and cumulants are identical, i.e., $c_3^{x_i}(\tau_1, \tau_2) = m_3^{x_i}(\tau_1, \tau_2)$.

4. Compute the average cumulants as:

$$\hat{c}_3^x(\tau_1, \tau_2) = \frac{1}{K} \sum_{i=1}^K m_3^{x_i}(\tau_1, \tau_2) \quad (76.10)$$

5. Obtain the third-order spectrum (bispectrum) estimate as

$$\hat{C}_3^x(\omega_1, \omega_2) = \sum_{\tau_1=-L}^L \sum_{\tau_2=-L}^L \hat{c}_3^x(\tau_1, \tau_2) e^{-j(\omega_1 \tau_1 + \omega_2 \tau_2)} w(\tau_1, \tau_2) , \quad (76.11)$$

where $L < M-1$, and $w(\tau_1, \tau_2)$ is a two-dimensional window of bounded support, introduced to smooth out edge effects. The bandwidth of the final bispectrum estimate is $\Delta = 1/L$.

A complete description of appropriate windows that can be used in (76.11) and their properties can be found in [16]. A good choice of cumulant window is:

$$w(\tau_1, \tau_2) = d(\tau_1) d(\tau_2) d(\tau_1 - \tau_2) , \quad (76.12)$$

where

$$d(\tau) = \begin{cases} \frac{1}{\pi} |\sin \frac{\pi \tau}{L}| + (1 - \frac{|\tau|}{L}) \cos \frac{\pi \tau}{L} & |\tau| \leq L \\ 0 & |\tau| > L \end{cases} \quad (76.13)$$

which is known as the minimum bispectrum bias supremum [17].

Direct Method

Let $X(k), k = 1, \dots, N$ be the available data.

1. Segment the data into K records of M samples each. Let $X^i(k)$, $k = 1, \dots, M$, represent the i th record.
2. Subtract the mean of each record.
3. Compute the Discrete Fourier Transform $F_x^i(k)$ of each segment, based on M points, i.e.,

$$F_x^i(k) = \sum_{n=0}^{M-1} X^i(n) e^{-j \frac{2\pi}{M} nk}, \quad k = 0, 1, \dots, M-1, \quad i = 1, 2, \dots, K. \quad (76.14)$$

4. The third-order spectrum of each segment is obtained as

$$C_3^{x_i}(k_1, k_2) = \frac{1}{M} F_x^i(k_1) F_x^i(k_2) F_x^{i*}(k_1 + k_2), \quad i = 1, \dots, K. \quad (76.15)$$

Due to the bispectrum symmetry properties, $C_3^{x_i}(k_1, k_2)$ need to be computed only in the triangular region $0 \leq k_2 \leq k_1$, $k_1 + k_2 < M/2$.

5. In order to reduce the variance of the estimate additional smoothing over a rectangular window of size $(M_3 \times M_3)$ can be performed around each frequency, assuming that the third-order spectrum is smooth enough, i.e.,

$$\tilde{C}_3^{x_i}(k_1, k_2) = \frac{1}{M_3^2} \sum_{n_1=-M_3/2}^{M_3/2-1} \sum_{n_2=-M_3/2}^{M_3/2-1} C_3^{x_i}(k_1 + n_1, k_2 + n_2). \quad (76.16)$$

6. Finally, the third-order spectrum is given as the average over all third-order spectra, i.e.,

$$\hat{C}_3^x(\omega_1, \omega_2) = \frac{1}{K} \sum_{i=1}^K \tilde{C}_3^{x_i}(\omega_1, \omega_2), \quad \omega_i = \frac{2\pi}{M} k_i, \quad i = 1, 2. \quad (76.17)$$

The final bandwidth of this bispectrum estimate is $\Delta = M_3/M$, which is the spacing between frequency samples in the bispectrum domain.

For large N , and as long as

$$\Delta \rightarrow 0, \text{ and } \Delta^2 N \rightarrow \infty \quad (76.18)$$

[32], both the direct and the indirect methods produce asymptotically unbiased and consistent bispectrum estimates, with real and imaginary part variances:

$$\begin{aligned} \text{var} \left(\text{Re} \left[\hat{C}_3^x(\omega_1, \omega_2) \right] \right) &= \text{var} \left(\text{Im} \left[\hat{C}_3^x(\omega_1, \omega_2) \right] \right) \\ &= \frac{1}{\Delta^2 N} C_2^x(\omega_1) C_2^x(\omega_2) C_2^x(\omega_1 + \omega_2) = \begin{cases} \frac{V L^2}{MK} C_2^x(\omega_1) C_2^x(\omega_2) C_2^x(\omega_1 + \omega_2) & \text{indirect} \\ \frac{M}{K M_3^2} C_2^x(\omega_1) C_2^x(\omega_2) C_2^x(\omega_1 + \omega_2) & \text{direct} \end{cases} \end{aligned} \quad (76.19)$$

where V is the energy of the bispectrum window.

From the above expressions, it becomes apparent that the bispectrum estimate variance can be reduced by increasing the number of records, or reducing the size of the region of support of the window in the cumulant domain (L), or increasing the size of the frequency smoothing window (M_3), etc. The relation between the parameters M , K , L , M_3 should be such that (76.18) is satisfied.

76.4 Linear Processes

Let $x(k)$ be generated by exciting a linear time-invariant (LTI) system with frequency response $H(\omega)$ with a non-Gaussian process $v(k)$. Its n th order spectrum can be written as

$$C_n^x(\omega_1, \omega_2, \dots, \omega_{n-1}) = C_n^v(\omega_1, \omega_2, \dots, \omega_{n-1}) H(\omega_1) \cdots H(\omega_{n-1}) H^*(\omega_1 + \cdots + \omega_{n-1}). \quad (76.20)$$

If $v(k)$ is n th order white then (76.20) becomes

$$C_n^x(\omega_1, \omega_2, \dots, \omega_{n-1}) = \gamma_n^v H(\omega_1) \cdots H(\omega_{n-1}) H^*(\omega_1 + \cdots + \omega_{n-1}), \quad (76.21)$$

where γ_n^v is a scalar constant and equals the n th order spectrum of $v(k)$. For a linear non-Gaussian random process $X(k)$, the n th order spectrum can be factorized as in (76.21) for every order n , while for a nonlinear process such a factorization might be valid for some orders only (it is always valid for $n = 2$).

If we express $H(\omega) = |H(\omega)| \exp\{j\phi_h(\omega)\}$, then (76.21) can be written as

$$|C_n^x(\omega_1, \omega_2, \dots, \omega_{n-1})| = \gamma_n^v |H(\omega_1)| \cdots |H(\omega_{n-1})| |H^*(\omega_1 + \cdots + \omega_{n-1})|, \quad (76.22)$$

and

$$|\psi_n^x(\omega_1, \omega_2, \dots, \omega_{n-1})| = \phi_h(\omega_1) + \cdots + \phi_h(\omega_{n-1}) - \phi_h(\omega_1 + \cdots + \omega_{n-1}), \quad (76.23)$$

where $\psi_n^x()$ is the phase of the n th order spectrum.

It can be shown easily that the cumulant spectra of successive orders are related as follows:

$$C_n^x(\omega_1, \omega_2, \dots, 0) = C_{n-1}^x(\omega_1, \omega_2, \dots, \omega_{n-2}) H(0) \frac{\gamma_n^v}{\gamma_{n-1}^v}. \quad (76.24)$$

As a result, the power spectrum of a Gaussian linear process can be reconstructed from the bispectrum up to a constant term, i.e.,

$$C_3^x(\omega, 0) = C_2^x(\omega) \frac{\gamma_3^v}{\gamma_2^v}. \quad (76.25)$$

To reconstruct the phase $\phi_h(\omega)$ from the bispectral phase $\psi_3^x(\omega_1, \omega_2)$ several algorithms have been suggested. A description of different phase estimation methods can be found in [14] and also in [16].

76.4.1 Nonparametric Methods

Consider $x(k)$ generated as shown in Fig. 76.2. The system transfer function can be written as

$$H(z) = cz^{-r} I(z^{-1}) O(z) = cz^{-r} \frac{\prod_i (1 - a_i z^{-1})}{\prod_i (1 - b_i z^{-1})} \Pi_i (1 - c_i z), \quad |a_i|, |b_i|, |c_i| < 1, \quad (76.26)$$

where $I(z^{-1})$ and $O(z)$ are the minimum and maximum phase parts of $H(z)$, respectively; c is a constant; and r is an integer. The output n th order cumulant equals [2]

$$\begin{aligned} c_n^x(\tau_1, \dots, \tau_{n-1}) &= c_n^y(\tau_1, \dots, \tau_{n-1}) + c_n^w(\tau_1, \dots, \tau_{n-1}) \\ &= c_n^y(\tau_1, \dots, \tau_{n-1}) \end{aligned} \quad (76.27)$$

$$= \gamma_n^v \sum_{k=0}^{\infty} h(k) h(k + \tau_1) \cdots h(k + \tau_{n-1}), \quad n \geq 3 \quad (76.28)$$

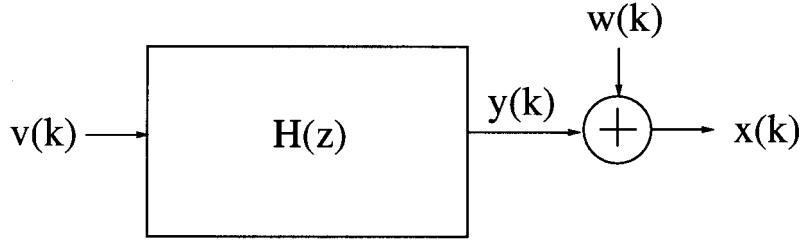


FIGURE 76.2: Single channel model.

where the noise contribution in (76.27) was zero due to the Gaussianity assumption. The Z-domain equivalent of (76.28) for $n = 3$ is

$$C_3^x(z_1, z_2) = \gamma_3^v H(z_1) H(z_2) H\left(z_1^{-1} z_2^{-1}\right). \quad (76.29)$$

Taking the logarithm of $C_3^x(z_1, z_2)$ followed by an inverse 2-D Z-transform we obtain the output bicepstrum $b_x(m, n)$. The bicepstrum of linear processes is nonzero only along the axes ($m = 0, n = 0$) and the diagonal $m = n$ [21]. Along these lines the bicepstrum is equal to the complex cepstrum, i.e.,

$$b_x(m, n) = \begin{cases} \hat{h}(m) & m \neq 0, n = 0 \\ \hat{h}(n) & n \neq 0, m = 0 \\ \hat{h}(-n) & m = n, m \neq 0 \\ \ln(c\gamma_n^v) & m = n = 0, \\ 0 & \text{elsewhere} \end{cases} \quad (76.30)$$

where $\hat{h}(n)$ denotes complex cepstrum [20]. From (76.30), the system impulse response $h(k)$ can be reconstructed from $b_x(m, 0)$ (or $b_x(0, m)$, or $b_x(m, m)$), within a constant and a time delay, via inverse cepstrum operations. The minimum and maximum phase parts of $H(z)$ can be reconstructed by applying inverse cepstrum operations on $b_x(m, 0)u(m)$ and $b_x(m, 0)u(-m)$, respectively, where $u(m)$ is the unit step function.

To avoid phase unwrapping with the logarithm of the bispectrum which is complex, the bicepstrum can be estimated using the group delay approach:

$$b_x(m, n) = \frac{1}{m} F^{-1} \left\{ \frac{F[\tau_1 c_3^x(\tau_1, \tau_2)]}{C_3^x(\omega_1, \omega_2)} \right\}, \quad m \neq 0 \quad (76.31)$$

with $b_x(0, n) = b_x(n, 0)$, and $F\{\cdot\}$ and $F^{-1}\{\cdot\}$ denoting 2-D Fourier transform operator and its inverse, respectively.

The cepstrum of the system can also be computed directly from the cumulants of the system output based on the equation [21]:

$$\begin{aligned} \sum_{k=1}^{\infty} k \hat{h}(k) [c_3^x(m-k, n) - c_3^x(m+k, n+k)] + k \hat{h}(-k) [c_3^x(m-k, n-k) - c_3^x(m+k, n)] \\ = mc_3^x(m, n) \end{aligned} \quad (76.32)$$

If $H(z)$ has no zeros on the unit circle its cepstrum decays exponentially, thus (76.32) can be truncated to yield an approximate equation. An overdetermined system of truncated equations can be formed for different values of m and n , which can be solved for $\hat{h}(k)$, $k = \dots, -1, 1, \dots$. The system response $h(k)$ then can be recovered from its cepstrum via inverse cepstrum operations.

The bicepstrum approach for system reconstruction described above led to estimates with smaller bias and variance than other parametric approaches at the expense of higher computational complexity [21]. The analytic performance evaluation of the bicepstrum approach can be found in [25].

The inverse Z-transform of the logarithm of the trispectrum (fourth-order spectrum), or otherwise tricepstrum, $t_x(m, n, l)$, of linear processes is also zero everywhere except along the axes and the diagonal $m = n = l$. Along these lines it equals the complex cepstrum, thus $h(k)$ can be recovered from slices of the tricepstrum based on inverse cepstrum operations.

For the case of nonlinear processes, the bicepstrum will be nonzero everywhere [4]. The distinctly different structure of the bicepstrum corresponding to linear and nonlinear processes has led to tests of linearity [4].

A new nonparametric method has been recently proposed in [1, 26] in which the cepstrum $\hat{h}(k)$ is obtained as:

$$\hat{h}(-k) = \frac{\hat{p}_n^x(k; e^{j\beta_1}) - \hat{p}_n^x(k; e^{j\beta_2})}{e^{j(n-2)\beta_1 k} - e^{j(n-2)\beta_2 k}}, \quad k \neq 0, n > 2 \quad (76.33)$$

where $p_n^x(k; e^{j\beta_i})$ is the time domain equivalent of the n th order spectrum slice defined as:

$$P_n^x(z; e^{j\beta_i}) = C_n^x(z, e^{j\beta_i}, \dots, e^{j\beta_i}). \quad (76.34)$$

The denominator of (76.33) is nonzero if

$$|\beta_1 - \beta_2| \neq \frac{2\pi l}{k(n-2)}, \quad \text{for every integer } k \text{ and } l. \quad (76.35)$$

This method reconstructs a complex system using two slices of the n th order spectrum. The slices, defined as shown above, can be selected arbitrarily as long as their distance satisfy (76.35). If the system is real, one slices is sufficient for the reconstruction. It should be noted that the cepstra appearing in (76.33) require phase unwrapping. The main advantage of this method is that the freedom to choose the higher-order spectra areas to be used in the reconstruction allows one to avoid regions dominated by noise or finite data length effects. Also, corresponding to different slice pairs various independent representations of the system can be reconstructed. Averaging out these representations can reduce estimation errors [26].

Along the lines of system reconstruction from selected HOS slices, another method has been proposed in [28, 29] where the $\log H(k)$ is obtained as a solution to a linear system of equations. Although logarithmic operation is involved, no phase unwrapping is required and the principal argument can be used instead of real phase. It was also shown that, as long as the grid size and the distance between the slices are coprime, reconstruction is always possible.

76.4.2 Parametric Methods

One of the popular approaches in system identification has been the construction of a white noise driven, linear time invariant model from a given process realization.

Consider the real autoregressive moving average (ARMA) stable process $y(k)$ given by:

$$\sum_{i=0}^p a(i)y(k-i) = \sum_{j=0}^q b(j)v(k-j) \quad (76.36)$$

$$x(k) = y(k) + w(k) \quad (76.37)$$

where $a(i)$, $b(j)$ represent the AR and MA parameters of the system, $v(k)$ is an independent identically distributed random process, and $w(k)$ represents zero-mean Gaussian noise.

Equations analogous to the Yule-Walker equations can be derived based on third-order cumulants of $x(k)$, i.e.,

$$\sum_{i=0}^p a(i)c_3^x(\tau - i, j) = 0, \quad \tau > q, \quad (76.38)$$

or

$$\sum_{i=1}^p a(i)c_3^x(\tau - i, j) = -c_3^x(\tau, j), \quad \tau > q, \quad (76.39)$$

where it was assumed $a(0) = 1$. Concatenating (76.39) for $\tau = q + 1, \dots, q + M$, $M \geq 0$ and $j = q - p, \dots, q$, the matrix equation

$$\underline{C}\underline{a} = \underline{c} \quad (76.40)$$

can be formed, where \underline{C} and \underline{c} are a matrix and a vector, respectively, formed by third-order cumulants of the process according to (76.39), and the vector \underline{a} contains the AR parameters. If the AR order p is unknown and (76.40) is formed based on an overestimate of p , the resulting matrix \underline{C} always has rank p . In this case, the AR parameters can be obtained using a low-rank approximation of \underline{C} [5].

Using the estimated AR parameters, $\hat{a}(i)$, $i = 1, \dots, p$, a p th order filter with transfer function $\hat{A}(z) = 1 + \sum_{i=1}^p \hat{a}(i)z^{-1}$ can be constructed. Based on the filtered through $\hat{A}(z)$ process $x(k)$, i.e., $\tilde{x}(k)$, or otherwise known as the residual time series [5], the MA parameters can be estimated via any MA method [15], for example:

$$b(k) = \frac{c_3^{\tilde{x}}(q, k)}{c_3^{\tilde{x}}(q, 0)}, \quad k = 0, 1, \dots, q \quad (76.41)$$

known as the $c(q, k)$ formula [6].

Practical problems associated with the described approach are sensitivity to model order mismatch, and AR estimation errors that propagate in the estimation of the MA parameters. A significant amount of research has been devoted to the ARMA parameter estimation problem. A thorough review of existing ARMA system identification methods can be found in [15, 16]; a more recent method can be found in [24].

76.5 Nonlinear Processes

Despite the fact that progress has been established in developing the theoretical properties of nonlinear models, only a few statistical methods exist for detection and characterization of nonlinearities from a finite set of observations. In this section, we will consider nonlinear Volterra systems excited by Gaussian stationary inputs. Let $y(k)$ be the response of a discrete time invariant p th order Volterra filter whose input is $x(k)$. Then,

$$y(k) = h_0 + \sum_i \sum_{\tau_1, \dots, \tau_i} h_i(\tau_1, \dots, \tau_i) x(k - \tau_1) \cdots x(k - \tau_i), \quad (76.42)$$

where $h_i(\tau_1, \dots, \tau_i)$ are the Volterra kernels of the system, which are symmetric functions of their arguments; for causal systems $h_i(\tau_1, \dots, \tau_i) = 0$ for any $\tau_i < 0$.

The output of a second-order Volterra system when the input is zero-mean stationary is

$$y(k) = h_0 + \sum_{\tau_1} h_1(\tau_1) x(k - \tau_1) + \sum_{\tau_1} \sum_{\tau_2} h_2(\tau_1, \tau_2) x(k - \tau_1) x(k - \tau_2). \quad (76.43)$$

Equation (76.43) can be viewed as a parallel connection of a linear system $h_1(\tau_1)$ and a quadratic system $h_2(\tau_1, \tau_2)$ as illustrated in Fig. 76.3. Let

$$c_2^{xy}(\tau) = E \{x(k + \tau)[y(k) - m_1^y]\} \quad (76.44)$$

be the cross-covariance of input and output, and

$$c_3^{xy}(\tau_1, \tau_2) = E \{x(k + \tau_1)x(k + \tau_2)[y(k) - m_1^y]\} \quad (76.45)$$

be the third-order cross-cumulant sequence of input and output.

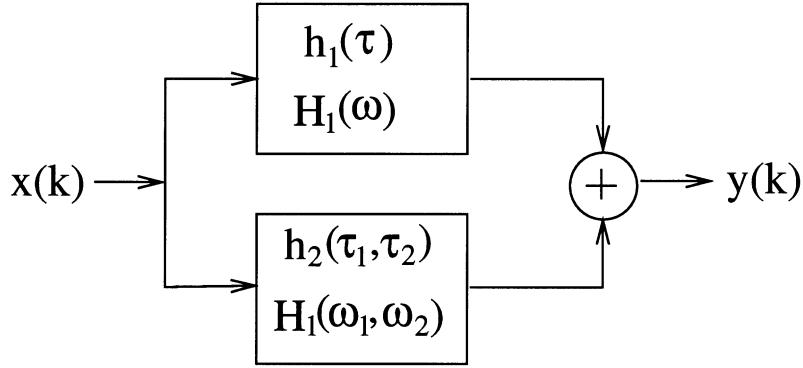


FIGURE 76.3: Second-order Volterra system. Linear and quadratic parts are connected in parallel.

It can be shown that the system's linear part can be identified by

$$H_1(-\omega) = \frac{C_2^{xy}(\omega)}{C_2^x(\omega)}, \quad (76.46)$$

and the quadratic part by

$$H_2(-\omega_1, -\omega_2) = \frac{C_3^{xy}(\omega_1, \omega_2)}{2C_2^x(\omega_1)C_2^x(\omega_2)}, \quad (76.47)$$

where $C_2^{xy}(\omega)$ and $C_3^{xy}(\omega_1, \omega_2)$ are the Fourier transforms of $c_2^{xy}(\tau)$ and $c_3^{xy}(\tau_1, \tau_2)$, respectively. It should be noted that the above equations are valid only for Gaussian input signals. More general results assuming non-Gaussian input have been obtained in [9, 27]. Additional results on particular nonlinear systems have been reported in [3, 33].

An interesting phenomenon caused by a second-order nonlinearity is the quadratic phase coupling. There are situations where nonlinear interaction between two harmonic components of a process contribute to the power of the sum and/or difference frequencies. The signal

$$x(k) = A \cos(\lambda_1 k + \theta_1) + B \cos(\lambda_2 k + \theta_2) \quad (76.48)$$

after passing through the quadratic system:

$$z(k) = x(k) + \epsilon x^2(k), \quad \epsilon \neq 0. \quad (76.49)$$

contains cosinusoidal terms in $(\lambda_1, \theta_1), (\lambda_2, \theta_2), (2\lambda_1, 2\theta_1), (2\lambda_2, 2\theta_2), (\lambda_1 + \lambda_2, \theta_1 + \theta_2), (\lambda_1 - \lambda_2, \theta_1 - \theta_2)$. Such a phenomenon that results in phase relations that are the same as the frequency relations is called quadratic phase coupling [12]. Quadratic phase coupling can arise only among harmonically related components. Three frequencies are harmonically related when one of them is the sum or difference of the other two. Sometimes it is important to find out if peaks at harmonically related positions in the power spectrum are in fact phase coupled. Due to phase suppression, the power spectrum is unable to provide an answer to this problem.

As an example, consider the process [30]

$$X(k) = \sum_{i=1}^6 \cos(\lambda_i k + \phi_i) \quad (76.50)$$

where $\lambda_1 > \lambda_2 > 0, \lambda_4 + \lambda_5 > 0, \lambda_3 = \lambda_1 + \lambda_2, \lambda_6 = \lambda_4 + \lambda_5, \phi_1, \dots, \phi_5$ are all independent, uniformly distributed random variables over $(0, 2\pi)$, and $\phi_6 = \phi_4 + \phi_5$. Among the six frequencies, $(\lambda_1, \lambda_2, \lambda_3)$ and $(\lambda_4, \lambda_5, \lambda_6)$ are harmonically related, however, only λ_6 is the result of phase coupling between λ_4 and λ_5 . The power spectrum of this process consists of six impulses at $\lambda_i, i = 1, \dots, 6$ (see Fig. 76.4), offering no indication whether each frequency component is independent or result of frequency coupling. On the other hand, the bispectrum of $X(k)$, $C_3^x(\omega_1, \omega_2)$ (evaluate in its principal region) is zero everywhere, except at point (λ_4, λ_5) of the (ω_1, ω_2) plane, where it exhibits an impulse (Fig. 76.4(b)). The peak indicates that only λ_4, λ_5 are phase coupled.

The bicoherence index, defined as

$$P_3^x(\omega_1, \omega_2) = \frac{C_3^x(\omega_1, \omega_2)}{\sqrt{C_2^x(\omega_1) C_2^x(\omega_2) C_2^x(\omega_1 + \omega_2)}}, \quad (76.51)$$

has been extensively used in practical situations for the detection and quantification of quadratic phase coupling [12]. The value of the bicoherence index at each frequency pair indicates the degree of coupling among the frequencies of that pair. Almost all bispectral estimators can be used in (76.51). However, estimates obtained based on parametric modeling of the bispectrum have been shown to yield superior resolution [30, 31] than the ones obtained with conventional methods.

76.6 Applications/Software Available

Applications of HOS span a wide range of areas [19] such as oceanography (description of wave phenomena), earth sciences (atmospheric pressure, turbulence), crystallography, plasma physics (wave interaction, nonlinear phenomena), mechanical systems (vibration analysis, knock detection), economic time series, biomedical signal analysis (ultrasonic imaging, detection of wave coupling) image processing (texture modeling and characterization, reconstruction, inverse filtering), speech processing (pitch detection, voiced/unvoiced decision), communications (equalization, interference cancellation), array processing (direction of arrival estimation, estimation of number of sources, beamforming, source signal estimation, source classification), harmonic retrieval (frequency estimation), and time delay estimation. Over 500 references can be found in [37]. Additional references can be found in [16, 19, 23].

A software package for signal processing with HOS is the Hi-Spec toolbox, product of Mathworks, Inc. The functions included in Hi-Spec together with a short description are included in Table 76.1.

Acknowledgments

Most of the material presented in this chapter is based on the book *Higher-Order Spectra Analysis: A Non-Linear Signal Processing Framework* [16]. The author wishes to thank Dr. C.L. Nikias for his

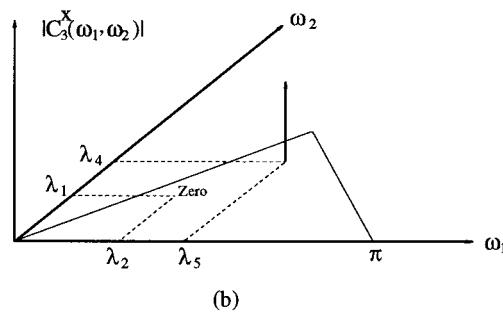
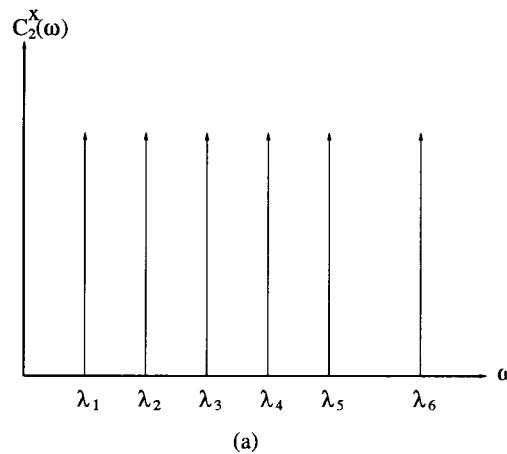


FIGURE 76.4: Quadratic phase coupling. (a) The power spectrum of the process described in Eq. (76.50) cannot determine what frequencies are coupled. (b) The corresponding magnitude bispectrum is zero everywhere in the principle region, except at points corresponding to phase coupled frequencies.

valuable input on the organization of the material. She also thanks U.R. Abeyratne for producing the figures in this chapter. Support for this work came from NSF under grant MIP-9553227 and the Whitaker Foundation.

TABLE 76.1 Functions Included in the Hi-Spec Package

Function name	Description
AR_RCEST	AR parameter estimation based on cumulants
ARMA_QS	ARMA parameter estimation via the Q-slice algorithm
ARMA RTS	ARMA parameter estimation via the residual time series method
ARMA_SYN	Generates ARMA synthetics
BICEPS	System identification via the bicepstrum approach
BISPEC_D	Bispectrum estimation via the direct method
BISPEC_I	Bispectrum estimation via the indirect method
CUM_EST	Estimates 2nd, 3rd, or 4th order cumulants
CUM_TRUE	Computes the theoretical cumulants of an ARMA model
DOA	Direction-of-arrival estimation
DOA_GEN	Generates synthetics for direction-of-arrival estimation
GL_STAT	Detection statistics for Hinich's Gaussianity and linearity tests
HARM EST	Estimates frequencies of harmonics in colored noise
HARM_GEN	Generates synthetics for the harmonic retrieval problem
MA_EST	MA parameters estimation
MATUL	System identification via the Matsuoka-Ulrych algorithm
QPC_GEN	Simulation generator for quadratic phase coupling
QPC_TOR	Detects quadratic phase coupling via parametric modeling of bispectrum
RP_IID	Generates samples of an i.i.d. random process
TDE	Estimates time delay between two signals using the parametric cross-cumulant method
TDE_GEN	Synthetics for time delay estimation

References

- [1] Abeyratne, U.R. and Petropulu, A.P., α -Weighted cumulant projections: a novel tool for system identification, *29th Annual Asilomar Conference on Signals, Systems and Computers*, California, Oct. 1995.
- [2] Brillinger, D.R. and Rosenblatt, M., Computation and interpretation of kth-order spectra, *Spectral Analysis of Time Series*, B. Harris, Ed., John Wiley & Sons, New York, 1967, 189–232.
- [3] Brillinger, D.R., The identification of a particular nonlinear time series system, *Biometrika*, 64(3), 509–515, 1977.
- [4] Erdem, A.T. and Tekalp, A.M., Linear bispectrum of signals and identification of nonminimum phase FIR systems driven by colored input, *IEEE Trans. on Signal Processing*, 40, 1469–1479, June 1992.
- [5] Giannakis, G.B. and Mendel, J.M., Cumulant-based order determination of non-Gaussian ARMA models, *IEEE Trans. on Acoustics, Speech and Signal Processing*, 38, 1411–1423, 1990.
- [6] Giannakis, G.B., Cumulants: a powerful tool in signal processing, *Proc. IEEE*, 75, 1987.
- [7] Haykin, S., *Nonlinear Methods of Spectral Analysis*, 2nd ed., Berlin, Germany, Springer-Verlag, 1983.
- [8] Hinich, M.J., Testing for gaussianity and linearity of a stationary time series, *J. Time Series Analysis*, 3(3), 169–176, 1982.
- [9] Hinich, M.J., Identification of the coefficients in a nonlinear time series of the quadratic type, *J. Economics*, 30, 269–288, 1985.
- [10] Huber, P.J., Kleiner, B. et.al., Statistical methods for investigating phase relations in stochastic processes, *IEEE Trans. on Audio and Electroacoustics*, Au-19(1), 78–86, 1976.
- [11] Kay, S.M., *Modern Spectral Estimation*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [12] Kim, Y.C. and Powers, E.J., Digital bispectral analysis of self-excited fluctuation spectral, *Phys. Fluids*, 21(8), 1452–1453, Aug. 1978.
- [13] Marple, Jr., S.L., *Digital Spectral Analysis with Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [14] Matsuoka, T. and Ulrych, T.J., Phase estimation using bispectrum, *Proc. of IEEE*, 72, 1403–1411, Oct., 1984.
- [15] Mendel, J.M., Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications, *IEEE Proc.*, 79, 278–305, March 1991.

- [16] Nikias, C.L. and Petropulu, A.P., *Higher-Order Spectra Analysis: a Nonlinear Signal Processing Framework*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [17] Nikias, C.L. and Raghubeer, M.R., Bispectrum estimation: a digital signal processing framework, *Proc. IEEE*, 75(7), 869–891, July 1987.
- [18] Nikias, C.L. and Chiang, H.-H., Higher-order spectrum estimation via noncausal autoregressive modeling and deconvolution, *IEEE Trans. Acoustics, Speech and Signal Processing*, 36(12), 1911–1913, Dec. 1988.
- [19] Nikias, C.L. and Mendel, J.M., Signal processing with higher-order spectra, *IEEE Signal Processing Magazine*, 10–37, July 1993.
- [20] Oppenheim, A.V. and Schafer, R.W., *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ., 1989.
- [21] Pan, R. and Nikias, C.L., The complex cepstrum of higher order cumulants and nonminimum phase system identification, *IEEE Trans. on Acoust., Speech and Signal Processing*, 36(2), 186–205, Feb. 1988.
- [22] Papoulis, A., *Probability random variables and stochastic processes*, McGraw-Hill, New York, 1984.
- [23] Petropulu, A.P., Higher-order spectra in biomedical signal processing, *CRC Press Biomedical Engineering Handbook*, CRC Press, Boca Raton, FL, 1995.
- [24] Petropulu, A.P., Noncausal nonminimum phase ARMA modeling of non-Gaussian processes, *IEEE Trans. on Signal Processing*, 43(8), 1946–1954, Aug. 1995.
- [25] Petropulu, A.P and Nikias, C.L., The complex cepstrum and bicepstrum: analytic performance evaluation in the presence of Gaussian noise, *IEEE Transactions Acoustics, Speech and Signal Processing, special mini-section on Higher-Order Spectral Analysis*, ASSP-38(7), July 1990.
- [26] Petropulu, A.P. and Abeyratne U.R., Signal reconstruction for higher-order spectra slices, *IEEE Trans. on Signal Processing*, Sept. 1997.
- [27] Powers, E.J., Ritz, C.K. et.al., Applications of digital polyspectral analysis to nonlinear systems modeling and nonlinear wave phenomena, *Workshop on Higher-Order Spectral Analysis*, Vail, CO, 73–77, June 1989.
- [28] Pozidis, H. and Petropulu, A.P., System reconstruction from selected bispectrum slices, *IEEE Signal Processing Workshop on Higher-Order Statistics*, Banff, Alberta, Canada, June 1997.
- [29] Pozidis, H. and Petropulu, A.P., System reconstruction using selected regions of the discretized HOS, *IEEE Transactions on Signal Processing*, submitted in 1997.
- [30] Raghubeer, M.R. and Nikias, C.L., Bispectrum estimation: A parametric approach, *IEEE Trans. on Acoust., Speech and Signal Processing*, ASSP 33(5), 1213–1230, Oct. 1985.
- [31] Raghubeer, M.R. and Nikias, C.L., Bispectrum estimation via AR modeling, *Signal Processing*, 10, 35–48, 1986.
- [32] Rao, T. Subba and Gabr, M.M., An introduction to bispectral analysis and bilinear time series models, *Lecture Notes in Statistics*, 24, Springer-Verlag, New York, 1984, 24.
- [33] Rozario, N. and Papoulis, A., The identification of certain nonlinear systems by only observing the output, *Workshop on Higher-Order Spectral Analysis*, Vail, CO, 73–77, June 1989.
- [34] Schetzen, M., *The Volterra and Wiener Theories on Nonlinear System*, updated edition, Krieger Publishing Company, Malabar, FL, 1989.
- [35] Swami, A. and Mendel, J.M., ARMA parameter estimation using only output cumulants, *IEEE Trans. Acoust., Speech and Signal Processing*, 38, 1257–1265, July 1990.
- [36] Tick, L.J., The estimation of transfer functions of quadratic systems, *Technometrics*, 3(4), 562–567, Nov. 1961.
- [37] United Signals & Systems, Inc., Comprehensive bibliography on higher-order statistics (spectra), Culver City, CA, 1992.

Sheaf Representations and Duality in Logic

Steve Awodey

Abstract The fundamental duality theories relating algebra and geometry that were discovered in the mid-20th century can also be applied to logic via its algebraization under categorical logic. They thereby result in known and new completeness theorems. This idea can be taken even further via what is sometimes called “categorification” to establish a new connection between logic and geometry, a glimpse of which can also be had in topos theory.

Preface

Shortly after finishing my PhD thesis, I received a friendly letter from Professor Lambek in which he expressed interest in a result of mine that extended his work with Moerdijk [14]. He later cited my result in some papers on the philosophy of mathematics (including [12, 13]), in which he developed a congenial position that attempted to reconcile the various competing ones in foundations on the basis of results concerning the free topos, the sheaf representations considered here, and related considerations from categorical logic.

The particular result in question, discussed in section 4 below, extends prior results by Lambek and Moerdijk [14] and Lambek [13], and was later extended further in joint work with my PhD students, first Henrik Forssell [2, 3], and then Spencer Breiner [4, 5]. This line of thought is, however, connected to a deeper one in modern mathematics, as I originally learned from the papers of Lambek. That insight inspired my original contribution and also the later joint work with my students, and it continues to fascinate and inspire me. The purpose of this survey is to sketch that line of thought, which owes more to Lambek than to anyone else.

Steve Awodey
Carnegie Mellon University, Pittsburgh, PA, USA, e-mail: awodey@cmu.edu

The main idea, in a nutshell, is that the ground-breaking duality theories developed in the mid-20th century can also be applied to logic, via its algebraization under categorical logic, and they thereby result in known and new completeness theorems. This insight, which is already quite remarkable, can as it turns out be taken even further—via what is sometimes called “categorification”—to establish an even deeper relation between logic and geometry, a glimpse of which can also be had in topos theory, and elsewhere.

1 Gelfand duality

Perhaps the ur-example of the sort of duality theory that we have in mind is the relation between topological spaces and commutative rings given by Gelfand duality (see [9] Ch. 4). To give a brief (and ahistorical) sketch, let X be a space and consider the ring of real-valued continuous functions on X , with pointwise algebraic operations,

$$\mathcal{C}(X) = \text{Top}(X, \mathbb{R}).$$

This construction is a (contravariant) functor from “geometry” to “algebra”,

$$\mathcal{C} : \text{Top}^{\text{op}} \longrightarrow \text{CRng}.$$

The functor \mathcal{C} can be shown to be full and faithful if we restrict to compact Hausdorff spaces X and (necessarily bounded) continuous functions $\mathcal{C}^*(X)$,

$$\mathcal{C}^* : \text{KHaus}^{\text{op}} \hookrightarrow \text{CRng}.$$

It then requires some further work to determine exactly *which* rings are of the form $\mathcal{C}^*(X)$ for some space X . These are called *C^* -algebras*, and they can be characterized as commutative rings A satisfying the following conditions ([9], §4.4):

1. the additive group of A is divisible and torsion free,
2. A has a partial order compatible with the ring structure and such that $a^2 \geq 0$ for all $a \in A$,
3. A is Archimedean, i.e. for every $a \in A$ there is an integer n such that $n \cdot 1_A \geq a$,
4. if $1_A \geq n \cdot a$ for all positive integers n , then $a \leq 0$,
5. A is complete in the norm

$$\|a\| = \inf\{q \in \mathbb{Q}^+ \mid q \cdot 1_A \geq a \text{ and } q \cdot 1_A \geq -a\}.$$

There are many equivalent specifications (most using complex numbers in place of reals).

Theorem 1 (Gelfand duality) *The category \mathbf{KHaus} of compact Hausdorff spaces is dual to the category $\mathbf{C}^*\mathbf{Alg}$ of C^* -algebras and their homomorphisms, via the functor \mathcal{C}^* :*

$$\mathbf{KHaus}^{\text{op}} \simeq \mathbf{C}^*\mathbf{Alg}.$$

How can we recover the space X from its ring of functions $\mathcal{C}^*(X)$?

- The points $x \in X$ determine maximal ideals in the ring $\mathcal{C}^*(X)$,

$$M_x = \{ f : X \rightarrow \mathbb{R} \mid f(x) = 0 \},$$

and every maximal ideal in $\mathcal{C}^*(X)$ is of this form for a unique $x \in X$.

- For any ring A , the (Zariski) topology on the set $\mathbf{Max}(A)$ of maximal ideals has a basis of open sets of the form:

$$B_a = \{ M \in \mathbf{Max}(A) \mid a \notin M \}, \quad a \in A.$$

- If A is a C^* -algebra, then this specification determines a compact Hausdorff space $X = \mathbf{Max}(A)$ such that $A \cong \mathcal{C}^*(X)$.

A key step in the proof is the following:

Proposition 1 *Let A be a C^* -algebra. For any maximal ideal M in A , the quotient field A/M is isomorphic to \mathbb{R} .*

It follows that there is an injection of rings,

$$A \rightarrowtail \prod_{M \in \mathbf{Max}(A)} A/M \cong \mathbb{R}^{\mathbf{Max}(A)}.$$

The image of this map can be shown to consist of the Zariski continuous functions, i.e. it is $\mathcal{C}^*(\mathbf{Max}(A))$.

2 Grothendieck duality for commutative rings

Grothendieck extended the Gelfand duality from C^* -algebras to *all* commutative rings by generalizing on the “geometric” side from compact Hausdorff spaces to the new notion of (*affine*) *schemes*,

$$\mathbf{Scheme}_{\text{aff}}^{\text{op}} \simeq \mathbf{CRng}.$$

The essential difference is to generalize the “ring of values” from the constant ring \mathbb{R} to a ring \mathcal{R} that “varies continuously over the space X ”, i.e. a *sheaf of rings*. The various rings \mathcal{R}_x that are the stalks of \mathcal{R} generalize the *local rings* of real-valued functions that vanish at the points $x \in X$. This change allows *every* commutative ring A to be seen as a ring of continuous functions

on a suitable space X_A (the prime spectrum of A), where the values of the functions are in a suitable sheaf of (local) rings \mathcal{R} on X_A (see [9] Ch. 5).

Definition 1 A ring is called *local* if it has a unique maximal ideal. Equivalently, if $0 \neq 1$, and

$$x + y \text{ is a unit} \quad \text{implies} \quad x \text{ is a unit or } y \text{ is a unit.} \quad (1)$$

Theorem 2 (Grothendieck sheaf representation) Let A be a ring. There is a space X_A with a sheaf of rings \mathcal{R} such that:

1. for every $p \in X_A$, the stalk \mathcal{R}_p is a local ring,
2. there is an isomorphism,

$$A \cong \Gamma(\mathcal{R}),$$

where $\Gamma(\mathcal{R})$ is the ring of global sections.

Thus every ring is isomorphic to the ring of global sections of a sheaf of local rings.

The space X_A in the theorem is the *prime spectrum* $\text{Spec}(A)$ of the ring A :

- points $p \in \text{Spec}(A)$ are prime ideals $p \subseteq A$,
- the (Zariski) topology has basic opens of the form:

$$B_f = \{p \in \text{Spec}(A) \mid f \notin p\}, \quad f \in A.$$

Note the similarity to the space $\text{Max}(A)$ of maximal ideals from the Gelfand case. Unlike that case, however, the functor

$$\text{Spec} : \text{CRng}^{\text{op}} \longrightarrow \text{Top}$$

is not full, and so we need to equip the spaces $\text{Spec}(A)$ with an additional structure.

The *structure sheaf* \mathcal{R} is determined at a basic open set B_f by “localizing” A at f ,

$$\mathcal{R}(B_f) = [f]^{-1}A$$

where $A \rightarrow [f]^{-1}A$ freely inverts all of the elements f, f^2, f^3, \dots

The stalk \mathcal{R}_p of this sheaf at a point $p \in \text{Spec}(A)$ is then seen to be the localization of A at $S_p = A \setminus p$,

$$\mathcal{R}_p = S_p^{-1}A.$$

The *affine scheme* $(\text{Spec}(A), \mathcal{R})$ presents A as a “ring of continuous functions” in the following sense:

- each element $f \in A$ determines a “continuous function”,

$$\hat{f} : \text{Spec}(A) \longrightarrow \mathcal{R},$$

except that the ring \mathcal{R} is itself “varying continuously over the space $\text{Spec}(A)$ ” – i.e. it is a sheaf – and the function \hat{f} is then a global section of the sheaf \mathcal{R} .

- Each stalk \mathcal{R}_p is a local ring, with a unique maximal ideal, corresponding to “those functions $\hat{f} : \text{Spec}(A) \rightarrow \mathcal{R}$ that vanish at p ”.
- $(\text{Spec}(A), \mathcal{R})$ is a “representation” of A in the sense that $f \mapsto \hat{f}$ is an isomorphism of rings

$$A \cong \Gamma(\mathcal{R}).$$

There is always an injective homomorphism from the global sections of a sheaf into the product of all the stalks,

$$\Gamma(\mathcal{R}) \rightarrowtail \prod_p \mathcal{R}_p.$$

Thus we have the following:

Corollary 1 (“Subdirect-product representation”) *Every ring A is isomorphic to a subring of a “direct product” of local rings. I.e. there is an injective ring homomorphism*

$$A \rightarrowtail \prod_p \mathcal{R}_p,$$

where the \mathcal{R}_p are all local rings.

3 Lambek-Moerdijk sheaf representation for toposes

Definition 2 Call a (small, elementary) topos \mathcal{E} *sublocal*¹ if its subterminal lattice $\text{Sub}_{\mathcal{E}}(1)$ has a unique maximal ideal. Equivalently, $0 \not\cong 1$ and for $x, y \in \text{Sub}_{\mathcal{E}}(1)$:

$$x \vee y = 1 \quad \text{implies} \quad x = 1 \text{ or } y = 1.$$

Note the formal analogy to the concept of local ring. In [14] the following analogue of the Grothendieck sheaf representation for rings is given for toposes (henceforth, *topos* unqualified will mean small, elementary topos):

Theorem 3 (Lambek-Moerdijk sheaf representation) *Let \mathcal{E} be a topos. There is a space $X_{\mathcal{E}}$ with a sheaf of toposes $\tilde{\mathcal{E}}$ such that:*

1. *for every $p \in X_{\mathcal{E}}$, the stalk $\tilde{\mathcal{E}}_p$ is a sublocal topos,*

¹ In the original work [14], and elsewhere, the term *local* was used for the concept here called *sublocal*, and another term was then required for the stronger condition that we call *local* in Definition 3 below.

2. for the topos $\Gamma(\tilde{\mathcal{E}})$ of global sections, there is an isomorphism,

$$\mathcal{E} \cong \Gamma(\tilde{\mathcal{E}}).$$

Thus every topos is isomorphic to the topos of global sections of a sheaf of sublocal toposes.

The space X mentioned in the theorem is what may be called the *sub-spectrum of the topos*, $X = \text{sSpec}(\mathcal{E})$; it is the prime ideal spectrum of the distributive lattice $\text{Sub}(1)$:

- the points $P \in \text{sSpec}(\mathcal{E})$ are prime ideals $P \subseteq \text{Sub}(1)$,
- the topology has basic opens of the form:

$$B_q = \{P \in \text{Spec}(\mathcal{E}) \mid q \notin P\}, \quad q \in \text{Sub}(1).$$

Note the close analogy to the space $\text{Spec}(A)$ for a commutative ring A .

The lattice of all open sets of $\text{sSpec}(\mathcal{E})$ is then (isomorphic to) the ideal completion of the lattice $\text{Sub}(1)$,

$$\mathcal{O}(\text{Spec}(\mathcal{E})) = \text{Idl}(\text{Sub}(1)).$$

Next, let us define a *structure sheaf* $\tilde{\mathcal{E}}$ on $\text{sSpec}(\mathcal{E})$ by “slicing” \mathcal{E} at $q \in \text{Sub}(1)$,

$$\tilde{\mathcal{E}}(B_q) = \mathcal{E}/q.$$

This takes the place of the localization of a ring A at a basic open B_f :

$$\mathcal{R}_A(B_f) = [f]^{-1}A.$$

Note that \mathcal{E}/q “inverts” all those elements $p \in \text{Sub}(1)$ with $q \leq p$, in the sense that the canonical map $q^* : \mathcal{E} \rightarrow \mathcal{E}/q$ takes every $p \rightarrowtail 1$ to $q \wedge p \rightarrowtail q$, and so if $q \leq p$ then $q^*p = 1_q : q \rightarrowtail q$.

The fact that $\tilde{\mathcal{E}}$ is indeed a sheaf on $\text{sSpec}(\mathcal{E})$ comes down to showing that, for any $p, q \in \text{Sub}(1)$, there is a canonical equalizer of toposes (and logical morphisms),

$$\mathcal{E}/p \vee q \rightarrowtail \mathcal{E}/p \times \mathcal{E}/q \rightrightarrows \mathcal{E}/p \wedge q.$$

This in turn says that in a diagram of the form:

$$\begin{array}{ccccc}
& X & \xrightarrow{\quad} & Q & \\
P \downarrow & \swarrow & \downarrow & \searrow & \downarrow \\
& Y & \xleftarrow{\quad} & & \\
& p \wedge q & \xrightarrow{\quad} & q & \\
\downarrow & \swarrow & \downarrow & \searrow & \downarrow \\
p & \xrightarrow{\quad} & p \vee q & &
\end{array}$$

with a pushout-pullback of monos in the base, and the two vertical squares involving X given as pullbacks, one can complete the cube as indicating by first forming the pushout Y on the top face, and then obtaining the front vertical map from Y , and the resulting new vertical faces will then also be pullbacks. This is a rather special “descent condition” for the presheaf $\mathcal{E}/-$.

The stalk $\tilde{\mathcal{E}}_P$ of this sheaf at a point $P \in \text{sSpec}(\mathcal{E})$ is computed as the filter-quotient of \mathcal{E} over the complement of the prime ideal $P \subseteq \text{Sub}_{\mathcal{E}}(1)$, i.e. the prime filter $P^c = \text{Sub}(1) \setminus P$. Thus for the stalk we have the (filtered) colimit (taken in Cat , but again a topos):

$$\tilde{\mathcal{E}}_P = \varinjlim_{q \not\in P} \mathcal{E}/q.$$

For this stalk topos, one then has the subterminal lattice:

$$\text{Sub}_{\tilde{\mathcal{E}}_P}(1) \cong \text{Sub}_{\mathcal{E}}(1)/P^c,$$

where $\text{Sub}_{\mathcal{E}}(1)/P^c$ is the quotient Heyting algebra by the prime filter P^c . Since for the prime filter P^c we have $p \vee q \in P^c$ implies $p \in P^c$ or $q \in P^c$, it thus follows that the stalk topos $\tilde{\mathcal{E}}_P$ is indeed sublocal.

Finally, for the global sections of $\tilde{\mathcal{E}}$ we have simply:

$$\Gamma(\tilde{\mathcal{E}}) \cong \tilde{\mathcal{E}}(B_{\top}) = \mathcal{E}/1 \cong \mathcal{E}.$$

Thus the topos of global sections of $\tilde{\mathcal{E}}$ is indeed isomorphic to the original topos \mathcal{E} . In this way, \mathcal{E} is isomorphic to the global sections of a sheaf of sublocal toposes.

Again, there is always an injection from the global sections into the product of the stalks, which in this case gives a conservative logical morphism of the form

$$\mathcal{E} \cong \Gamma(\tilde{\mathcal{E}}) \rightarrowtail \prod_{P \in \text{sSpec}(\mathcal{E})} \tilde{\mathcal{E}}_P.$$

Corollary 2 *Every topos has a conservative logical morphism into a product of sublocal toposes.*

3.1 Lambek's modified sheaf representation for toposes

Now consider the following logical interpretation of the sheaf representation theorem for toposes and its corollary.

- A topos \mathcal{E} can be regarded as the syntactic category $\mathcal{E}_{\mathbb{T}}$ of a theory \mathbb{T} in Intuitionistic Higher-Order Logic (IHOL). Thus for any sentence ϕ in the language of the theory \mathbb{T} ,

$$\mathcal{E}_{\mathbb{T}} \models \phi \quad \text{iff} \quad \mathbb{T} \vdash \phi.$$

- A sublocal topos \mathcal{S} is one that is *consistent* $\mathcal{S} \not\models \perp$ and has the *disjunction property*

$$\mathcal{S} \models \phi \vee \psi \quad \text{iff} \quad \mathcal{S} \models \phi \text{ or } \mathcal{S} \models \psi,$$

for all sentences ϕ, ψ . Such sublocal toposes are more Set-like than a general one, and can thus be regarded as suitable semantics for logical theories.

- The “subdirect-product representation” given by Corollary 2 is a logical completeness theorem with respect to interpretations $\mathcal{E}_{\mathbb{T}} \rightarrow \mathcal{S}$ of \mathbb{T} into sublocal toposes \mathcal{S} . It says that, for any theory \mathbb{T} in IHOL, a sentence ϕ is provable, $\mathbb{T} \vdash \phi$, iff it is true in every interpretation of \mathbb{T} in a sublocal topos \mathcal{S} . Thus IHOL is complete with respect to models in sublocal toposes.
- The sheaf representation is a Kripke-style completeness theorem for IHOL, with $\tilde{\mathcal{E}}$ as a “sheaf of possible worlds” (see [12]).

Under this interpretation, however, the present sheaf representation is not entirely satisfactory, because we would really like the “semantic toposes” \mathcal{S} to be even more Set-like, in addition to being sublocal, by also having the *existence property*:

$$\mathcal{S} \models (\exists x : A)\varphi(x) \quad \text{iff} \quad \mathcal{S} \models \varphi(a), \text{ for some closed } a : A.$$

Definition 3 A topos \mathcal{S} will be called *local* if the terminal object 1 is both indecomposable and projective, i.e. the global sections functor

$$\Gamma = \text{Hom}_{\mathcal{S}}(1, -) : \mathcal{S} \longrightarrow \text{Set}$$

preserves all finite coproducts and epimorphisms. Note that a local topos is exactly one that is consistent and has both the disjunction and existence properties.

In the paper [12], Lambek gave the following improvement over the sublocal sheaf representation:

Theorem 4 (Lambek sheaf representation) *Let \mathcal{E} be a topos. There is a faithful logical functor $\mathcal{E} \rightarrow \mathcal{F}$ and a space X with a sheaf of toposes $\tilde{\mathcal{F}}$ such that:*

1. *for every $p \in X$, the stalk $\tilde{\mathcal{F}}_p$ is a local topos,*

2. for the global sections of $\tilde{\mathcal{F}}$ there is an isomorphism $\mathcal{F} \cong \Gamma(\tilde{\mathcal{F}})$.

Thus every topos is a subtopos of one that is isomorphic to the global sections of a sheaf of local toposes.

The proof was inspired by the Henkin completeness theorem for higher-order logic [8], and first performs a sort of “Henkinization” of \mathcal{E} to get a bigger topos $\mathcal{E} \rightarrowtail \mathcal{F}$ with witnesses for all existential quantifiers, in a suitable sense. This result then suffices for a subdirect-product embedding of any topos \mathcal{E} into a product of local toposes, and therefore gives the desired logical completeness of IHOL with respect to such toposes, which are much more Set-like.

4 Local sheaf representation for toposes

The result from [1] mentioned above was this:

Theorem 5 (Local topos sheaf representation)

Let \mathcal{E} be a topos. There is a space $X_{\mathcal{E}}$ with a sheaf of toposes $\tilde{\mathcal{E}}$ such that:

1. for every $p \in X_{\mathcal{E}}$, the stalk $\tilde{\mathcal{E}}_p$ is a local topos,
2. for the global sections of $\tilde{\mathcal{E}}$ there is an equivalence $\mathcal{E} \simeq \Gamma(\tilde{\mathcal{E}})$.

Thus every topos is equivalent to the global sections of a sheaf of local toposes.

As before, this gives a subdirect-product representation of \mathcal{E} ,

$$\mathcal{E} \rightarrowtail \prod_{p \in X} \mathcal{S}_p,$$

into a product of local toposes $\mathcal{S}_p = \tilde{\mathcal{E}}_p$, and therefore implies the desired logical completeness of IHOL with respect to local toposes. This stronger result also gives better “Kripke semantics” for IHOL, since the “sheaf of possible worlds” (in the sense of [12]) now has local stalks.

For classical higher-order logic, something more can be said:

Lemma 1 Every Boolean, local topos \mathcal{S} is well-pointed, i.e. the global sections functor,

$$\Gamma = \text{Hom}_{\mathcal{S}}(1, -) : \mathcal{S} \rightarrow \text{Set}$$

is faithful.

Corollary 3 Every Boolean topos is isomorphic to the global sections of a sheaf of well-pointed toposes.

For Boolean toposes \mathcal{B} , we therefore have an embedding,

$$\mathcal{B} \rightarrowtail \prod_{p \in X} \mathcal{S}_p$$

as a subdirect-product of *well-pointed* toposes \mathcal{S}_p (this is [7], Thm 3.22). The logical counterpart now says:

Corollary 4 *Classical HOL is complete with respect to models in well-pointed toposes.*

A well-pointed topos is essentially a model of classical Zermelo set theory ([16], §VI.10). Indeed, it is worth emphasizing that the models of HOL here are *standard* models of classical HOL (i.e. with full function and power sets), taken in *varying* models \mathcal{S} of set theory.

Finally, taking the global sections $\Gamma : \mathcal{S}_p \rightarrow \text{Set}$ of each well-pointed topos \mathcal{S}_p , we get a faithful functor from any Boolean topos \mathcal{B} into a power of Set :

$$\mathcal{B} \rightarrowtail \prod_{p \in X} \mathcal{S}_p \rightarrowtail \prod_{p \in X} \text{Set} \cong \text{Set}^X.$$

However, the various composites $\mathcal{B} \rightarrow \mathcal{S}_p \rightarrow \text{Set}$ are now not logical functors, because they need not preserve exponentials; they do, however, preserve the first-order logical structure (they are also exact; thus we have another proof of [7] theorem 3.24). These composites are exactly what the logician calls a “Henkin” or “non-standard” model of HOL in Set . In this way, we recover the familiar “Henkin completeness theorem for HOL” [8]:

Corollary 5 *Classical HOL is complete with respect to Henkin models in Set .*

For the proof of the local topos sheaf representation theorem, these “Henkin models” will be taken as the points of the space $X_{\mathcal{E}}$, which we call the *space of models* (following [6]). In the sublocal case, the points were the *prime ideals* $P \subseteq \text{Sub}(1)$. These correspond exactly to the *lattice homomorphisms*

$$p : \text{Sub}_{\mathcal{E}}(1) \longrightarrow \mathbf{2}.$$

For the local case, we instead take *coherent functors*

$$P : \mathcal{E} \longrightarrow \text{Set},$$

which correspond to (Henkin) models of the “theory” \mathcal{E} .²

The topology on $X_{\mathcal{E}}$ can be described roughly as follows (see [1] for more details, but the idea for this topology originates with [10, 6]; it was also used in [2, 5]). To simplify things, let us regard \mathcal{E} as a classifying topos for a theory

² Of course, the collection of all such functors may be too big to form a *set*. The remedy, as explained in the paper [1], is to choose a suitable cardinal bound κ on the size of the models P .

\mathbb{T} , and say that a model $P : \mathcal{E} \rightarrow \text{Set}$ “satisfies” a sentence ϕ , which we may identify with its interpretation $\phi \mapsto 1_{\mathcal{E}}$, if $[\phi]^P = P(\phi) = 1$. Then we could mimic the subspectrum by taking as a basic open set all those models P that satisfy some fixed ϕ :

$$V_{\phi} = \{P \in X_{\mathcal{E}} \mid P \models \phi\}.$$

However, it turns out that there are too few such basic opens; thus we will also use formulas $\phi(x)$ with free variables. In order to say when $P \models \phi(x)$ we therefore equip each model P with a “labelling” $\alpha : \kappa \rightarrow |P|$ by elements of some fixed, large set κ , and we then define the notion of satisfaction of a formula by such a labelled model $(P, \alpha) \models \phi(x)$, which we write suggestively as $P \models \phi(\alpha)$. Thus the points of $X_{\mathcal{E}}$ are actually pairs (P, α) , and the basic open sets then have the form

$$V_{\phi(x)} = \{(P, \alpha) \in X_{\mathcal{E}} \mid P \models \phi(\alpha)\}$$

for all formulas $\phi(x)$. (This description is not entirely accurate, but it gives the idea for present purposes; see [1, 2, 5] for details.)

The *structure sheaf* $\tilde{\mathcal{E}}$ on $X_{\mathcal{E}}$ is again defined by “slicing” \mathcal{E} ,

$$\tilde{\mathcal{E}}(A) = \mathcal{E}/A \quad \text{for } A \in \mathcal{E},$$

but now it is first shown to be a *stack* on \mathcal{E} itself (with respect to the coherent topology). What this means is:

1. for any $A, B \in \mathcal{E}$, the canonical map is an equivalence,

$$\mathcal{E}/A + B \simeq \mathcal{E}/A \times \mathcal{E}/B,$$

2. for any epimorphism $e : B \twoheadrightarrow A$, the canonical map is an equivalence,

$$\mathcal{E}/A \simeq \text{des}(\mathcal{E}/B, e),$$

where $\text{des}(\mathcal{E}/B, e)$ is the category of objects of \mathcal{E}/B equipped with *descent data* with respect to $e : B \twoheadrightarrow A$.

The stack is then strictified to a *sheaf* of categories (see [1]), and then finally transferred from \mathcal{E} to the space $X_{\mathcal{E}}$ of models using a topos-theoretic covering theorem due to Butz and Moerdijk [6]. Call the resulting sheaf of categories on $X_{\mathcal{E}}$ again $\tilde{\mathcal{E}}$.

The *stalk* $\tilde{\mathcal{E}}_{(P, \alpha)}$ of the (transferred) sheaf at a point (P, α) can be calculated as the colimit,

$$\tilde{\mathcal{E}}_{(P, \alpha)} = \varinjlim_{A \in \int^P} \mathcal{E}/A,$$

where the (filtered!) category of elements $\int_{\mathcal{E}} P$ of the model $P : \mathcal{E} \rightarrow \text{Set}$ takes the place of the prime filter. As a key step, one shows that these stalks are indeed *local* toposes whenever $P : \mathcal{E} \rightarrow \text{Set}$ is a coherent functor.

Finally, for the *global sections* functor $\Gamma : \mathbf{Sh}(X_{\mathcal{E}}) \rightarrow \mathbf{Set}$, we still have:

$$\Gamma(\tilde{\mathcal{E}}) \simeq \mathcal{E}/1 \cong \mathcal{E}.$$

In this way, \mathcal{E} is indeed equivalent to the topos of global sections of a sheaf of local toposes on a space.

5 Stone duality for Boolean algebras

The foregoing sheaf representations for toposes suggest an analogous treatment for *pretoposes*, which would actually be somewhat better, because the \mathbf{Set} -valued models used for the points (and coming from the global sections of the stalks) would then all be *standard* models, rather than Henkin style, non-standard models. This suggests the possibility of a *duality theory for first-order logic*, analogous to that for affine schemes and commutative rings, with pretoposes playing the role of rings, the space of models playing the role of the prime spectrum, and the sheaf representation providing a structure sheaf.

This is more than just an analogy: it is a generalization of the classical Stone duality for Boolean algebras (= Boolean rings). From a logical point of view, the classical duality theory for Boolean algebras is the propositional case of the first-order one that we are proposing for pretoposes. (There is also a generalization from classical to intuitionistic logic, which is less of a stretch.) Thus let us briefly review the “propositional case” of classical Stone duality for Boolean algebras, before proceeding to the “first-order” case of pretoposes.

Recall (e.g. from [9], Ch. 5) that for a Boolean algebra B we have the Stone space $\mathbf{Stone}(B)$, which is defined exactly as the subterminal lattice $\mathbf{Sub}_{\mathcal{E}}(1)$ of a topos \mathcal{E} , i.e. $\mathbf{Stone}(B) = \mathbf{Spec}(B)$ is the prime spectrum of B (prime ideals in a Boolean algebra are always maximal, thus are exactly the complements of the ultrafilters, which are the usual points of $\mathbf{Stone}(B)$). We can represent the *points* $p \in \mathbf{Spec}(B)$ as Boolean homomorphisms,

$$p : B \longrightarrow \mathbf{2}.$$

And we can recover the Boolean algebra B from the space $\mathbf{Spec}(B)$ as the *clopen subsets*, which are represented by continuous maps,

$$f : \mathbf{Spec}(B) \longrightarrow \mathbf{2},$$

where (the underlying set of) $\mathbf{2}$ is given the discrete topology. Note that this is also a sheaf representation – but a constant one! The stalks are *local* Boolean algebras, which are always just $\mathbf{2}$.

Stone's representation theorem for Boolean algebras then says that there is always an injective homomorphism,

$$B \rightarrowtail \mathbf{2}^X \cong \mathcal{P}(X),$$

for a set X , which we can take to be the set of points of $\text{Spec}(B)$, i.e. the ultrafilters. This is therefore the usual subdirect-product embedding resulting from the sheaf representation.

There is, moreover, a contravariant equivalence of categories,

$$\begin{array}{ccc} & \text{Spec} & \\ \text{Bool} & \begin{array}{c} \swarrow \curvearrowright \\ \simeq \\ \searrow \curvearrowleft \end{array} & \text{Stone}^{\text{op}} \\ & \text{Clop} & \end{array}$$

Both of the functors Spec and Clop are given by homming into $\mathbf{2}$, albeit in two different categories.

Logically, a Boolean algebra B is always the “Lindenbaum-Tarski algebra” of a theory \mathbb{T} in *propositional* logic, and a Boolean homomorphism $B \rightarrowtail \mathbf{2}$ is then the same thing as a \mathbb{T} -model, i.e. a “truth-valuation”. Thus the points of $\text{Spec}(B)$ are *models* of the propositional theory \mathbb{T} . We are going to generalize this situation by replacing Boolean algebras with (Boolean) pretoposes, representing *first-order* logical theories, and replacing $\mathbf{2}$ -valued models with Set-valued models.

6 Stone duality for Boolean pretoposes

M. Makkai [15] has discovered a Stone duality for Boolean pretoposes with respect to what he terms *ultragroupoids* on the geometric/semantic side. These are groupoids (of models and isomorphisms) equipped with a primitive structure of ultraproducts of models, together with groupoid homomorphisms that preserve ultraproducts. The result is an equivalence of categories:

$$\begin{array}{ccc} \text{BoolPreTop} & \begin{array}{c} \swarrow \curvearrowright \\ \simeq \\ \searrow \curvearrowleft \end{array} & \text{UltraGpd}^{\text{op}} \end{array}$$

which, as in the propositional case, is mediated by homming into a special object, now Set in place of $\mathbf{2}$. This replacement, and the remarkable duality theory that results, is an instance of what is sometime called “categorification”, an idea that plays a guiding role throughout categorical logic. It follows in particular that every Boolean pretopos \mathcal{B} has a pretopos embedding into a power of Set.

$$\mathcal{B} \rightarrowtail \text{Set}^X,$$

where X is a set of “models”, i.e. pretopos functors $M : \mathcal{B} \rightarrow \text{Set}$.

We will show below that this last fact—which is essentially Gödel’s completeness theorem for first-order logic—is also a “subdirect-product representation” resulting from a sheaf representation of \mathcal{B} . But first we need to make a suitable “space of models”.

In joint work with H. Forssell [2, 3] Makkai’s ultragroupoids of models were replaced by *topological* groupoids of models, equipped with a Stone-Zariski type logical topology similar to the one used above for the local sheaf representation for toposes. In overview, our (topological) generalization of Stone duality from Boolean algebras to Boolean pretoposes works like this:

Boolean algebra B propositional theory	Boolean pretopos \mathcal{B} first-order theory
homomorphism $B \rightarrow \mathbf{2}$ truth-valuation	pretopos functor $\mathcal{B} \rightarrow \mathbf{Set}$ elementary model
topological space $\text{Spec}(B)$ of all valuations	topological groupoid $\text{Spec}(\mathcal{B})$ of all models and isos
continuous function $\text{Spec}(B) \rightarrow \mathbf{2}$ clopen set	continuous functor $\text{Spec}(\mathcal{B}) \rightarrow \mathbf{Set}$ coherent sheaf

To give a bit more detail of a few of the steps:

- The spectrum $\text{Spec}(\mathcal{B})$ of a Boolean pretopos \mathcal{B} is not just a space, but a *topological groupoid*, consisting of a space of (labelled) models (M, α) and a space of isos $i : M \cong N$. These are topologized by a *logical topology* of the same kind already considered, where the basic opens (of the space of models) are determined by satisfaction of formulas,

$$V_{\phi(x)} = \{(M, \alpha) \in \text{Spec}(\mathcal{B}) \mid M \models \phi(\alpha)\}.$$

- Morphisms $f : \text{Spec}(\mathcal{B}) \rightarrow \text{Spec}(\mathcal{B}')$ are just continuous groupoid homomorphisms. Every pretopos functor $F : \mathcal{B}' \rightarrow \mathcal{B}$ gives rise to such a homomorphism, essentially by precomposition, since

$$\text{Spec} : \text{BoolPreTop} \rightarrow \text{StoneTopGpd}^{\text{op}}$$

is representable,

$$\text{Spec}(\mathcal{B}) \simeq \text{BoolPreTop}(\mathcal{B}, \mathbf{Set}).$$

Thinking of such a pretopos functor $F : \mathcal{B}' \rightarrow \mathcal{B}$ as a “translation of theories”, the semantic functor $\text{Spec}(F)$ acts on models in the corresponding way.

- Recovering \mathcal{B} from $\text{Spec}(\mathcal{B})$ amounts to recovering an elementary theory (up to pretopos completion) from its models. This is done using hard results from topos theory due mainly to Joyal-Tierney and Joyal-Moerdijk [11, 10, 6]. Specifically, one shows that the category of *equivariant* sheaves on the topological groupoid $\text{Spec}(\mathcal{B})$ is equivalent to the (Grothendieck) topos of sheaves on \mathcal{B} for the coherent topology,

$$\text{Sh}_{\text{eq}}(\text{Spec}(\mathcal{B})) \simeq \text{Sh}(\mathcal{B}).$$

Logically, this gives two different presentations of the (Grothendieck) classifying topos of a first-order theory \mathbb{T} , such that $\mathcal{B} = \mathcal{B}_{\mathbb{T}}$ is the pretopos completion of (the syntactic category of) \mathbb{T} , and $\text{Spec}(\mathcal{B})$ is then the groupoid of \mathbb{T} -models.

It follows that \mathcal{B} is equivalent to the subcategory of *coherent objects* of this topos; thus \mathcal{B} is equivalent to the category of coherent, equivariant sheaves on the topological groupoid $\text{Spec}(\mathcal{B})$. These can be shown to correspond to certain continuous homomorphisms $\text{Spec}(\mathcal{B}) \rightarrow \text{Set}$, where the latter is the topological groupoid of sets, equipped with a suitable topology. In this sense, the coherent, equivariant sheaves generalize the clopen sets in a Stone space.

Unlike in the case of Boolean algebras, however, and unlike in Makkai’s theorem using ultragroupoids, we do not have an equivalence of categories, but only an *adjunction* [2, 3]:

Theorem 6 (Awodey-Forsell) *There is a contravariant adjunction,*

$$\begin{array}{ccc} & \text{Spec} & \\ \text{BoolPreTop} & \begin{array}{c} \nearrow \\ \searrow \end{array} & \text{StoneTopGpd}^{\text{op}}, \\ & \text{Coh} & \end{array}$$

in which both functors are given by homming into Set .

In particular, the “semantic” functor,

$$\text{Spec} : \text{BPreTop} \rightarrow \text{StoneTopGpd}^{\text{op}}$$

is *not full*: there are continuous functors between the groupoids of models that do not come from a “translation of theories”. Compare the case of commutative rings A, B , where an arbitrary continuous function

$$f : \text{Spec}(B) \rightarrow \text{Spec}(A)$$

need not come from a ring homomorphism $h : A \rightarrow B$.

We can of course characterize the “semantic functors” arising from a pretopos morphism as those that pull coherent sheaves back to coherent sheaves. Such “coherent” maps $f : \text{Spec}(\mathcal{B}) \rightarrow \text{Spec}(\mathcal{B}')$ will then correspond to pretopos maps $F : \mathcal{B}' \rightarrow \mathcal{B}$, simply by $f(M) \cong M \circ F$.

7 Sheaf representation for pretoposes

We now want to cut down the morphisms between the semantic groupoids $\text{Spec}(\mathcal{B})$ to just the coherent ones that come from pretopos functors. We will do this by endowing $\text{Spec}(\mathcal{B})$ with additional structure that is preserved by all such “syntactic” maps. Specifically, as for rings and affine schemes, we can equip the spectrum $\text{Spec}(\mathcal{B})$ of the pretopos \mathcal{B} with a “structure sheaf” $\tilde{\mathcal{B}}$, defined just as in the sheaf representation for toposes:

- Start with the pseudofunctor $\tilde{\mathcal{B}} : \mathcal{B}^{\text{op}} \rightarrow \text{Cat}$ with,

$$\tilde{\mathcal{B}}(X) \cong \mathcal{B}/X, \quad X \in \mathcal{B}.$$

The prestack $\tilde{\mathcal{B}}$ is actually a *stack* for the coherent topology, because \mathcal{B} is a pretopos.

- Strictify $\tilde{\mathcal{B}}$ to get a sheaf of categories (also called $\tilde{\mathcal{B}}$) on \mathcal{B} . The “stalk” of $\tilde{\mathcal{B}}$ at a “point” $M : \mathcal{E} \rightarrow \text{Set}$ (a pretopos functor) is then

$$\tilde{\mathcal{E}}_M \simeq \varinjlim_{A \in \int M} \mathcal{E}/A,$$

which is a *local* Boolean pretopos (1 is indecomposable and projective).

- There is an equivalence of Grothendieck toposes,

$$\text{Sh}(\mathcal{B}) \simeq \text{Sh}_{\text{eq}}(\text{Spec}(\mathcal{B})),$$

between sheaves on the pretopos \mathcal{B} , for the coherent Grothendieck topology, and equivariant sheaves on the topological groupoid $\text{Spec}(\mathcal{B})$ of (labelled) models.

- Move $\tilde{\mathcal{B}}$ across this equivalence in order to get an equivariant sheaf on $\text{Spec}(\mathcal{B})$. The result (also called $\tilde{\mathcal{B}}$) is thus a sheaf of local, Boolean pretoposes on $\text{Spec}(\mathcal{B})$.

And from a logical point of view:

- $\mathcal{B} = \mathcal{B}_{\mathbb{T}}$ is the Boolean pretopos completion of (the syntactic category of) a theory \mathbb{T} in (classical) FOL, and $\text{Spec}(\mathcal{B})$ is then the groupoid of \mathbb{T} -models.
- $\tilde{\mathcal{B}}$ is a sheaf of “local theories”. The stalk $\tilde{\mathcal{B}}_M$ at a \mathbb{T} -model M is a well-pointed pretopos representing the complete theory of M , with parameters

for all the elements of M added; it is what the logician calls the “elementary diagram” of the model M .

- As before, $\tilde{\mathcal{B}}$ has global sections $\Gamma(\tilde{\mathcal{B}}) \simeq \mathcal{B}$. So the original pretopos $\mathcal{B}_{\mathbb{T}}$ turns out to be the “theory of all the \mathbb{T} -models”.
- Since each stalk $\tilde{\mathcal{B}}_M$ is local, and well-pointed, the global sections functor $\Gamma_M : \tilde{\mathcal{B}}_M \rightarrow \text{Set}$ is a faithful *pretopos* morphism, i.e. a model in *Set*. In fact, the model $M : \mathcal{B} \rightarrow \text{Set}$ is naturally isomorphic to the composite:

$$M : \mathcal{B} \simeq \Gamma(\tilde{\mathcal{B}}) \rightarrow \tilde{\mathcal{B}}_M \xrightarrow{\Gamma_M} \text{Set}.$$

In sum, we have the following (see [4, 5]):

Theorem 7 (Awodey-Breiner) *Let \mathcal{B} be a Boolean pretopos. There is a topological groupoid G with an equivariant sheaf of pretoposes $\tilde{\mathcal{B}}$ such that:*

1. *for every $g \in G$, the stalk $\tilde{\mathcal{B}}_g$ is a well-pointed pretopos,*
2. *for the global sections of $\tilde{\mathcal{B}}$ there is an equivalence $\mathcal{B} \simeq \Gamma(\tilde{\mathcal{B}})$.*

Thus every Boolean pretopos is equivalent to the global sections of a sheaf of well-pointed pretoposes.

There is again an analogous result for the general (i.e. non-Boolean) case, with local pretoposes in place of well-pointed ones in the stalks. The associated subdirect-product representation is then the following:

Corollary 6 *For any pretopos \mathcal{E} , there is a pretopos embedding,*

$$\mathcal{E} \rightarrowtail \prod_{g \in X_{\mathcal{E}}} \mathcal{E}_g$$

*with each \mathcal{E}_g a local pretopos and $X_{\mathcal{E}}$ the set of points of the topological groupoid $\text{Spec}(\mathcal{E})$. If moreover \mathcal{B} is Boolean, then the local pretoposes \mathcal{B}_g are all well-pointed, and \mathcal{B} therefore embeds (as a pretopos!) into a power of *Set*:*

$$\mathcal{B} \rightarrowtail \prod_{g \in X_{\mathcal{B}}} \mathcal{B}_g \rightarrowtail \prod_{g \in X_{\mathcal{B}}} \text{Set} \simeq \text{Set}^{X_{\mathcal{B}}}.$$

In logical terms, the last statement is essentially the Gödel completeness theorem for first-order logic, repackaged. Of course, the proof made use of the equivalent fact that \mathcal{B} has enough pretopos functors $M : \mathcal{B} \rightarrow \text{Set}$.

8 Logical schemes

For a Boolean pretopos \mathcal{B} , call the pair

$$(\text{Spec}(\mathcal{B}), \tilde{\mathcal{B}})$$

just constructed an *affine logical scheme*. A morphism of affine logical schemes

$$(f, \tilde{f}) : (\text{Spec}(\mathcal{A}), \tilde{\mathcal{A}}) \longrightarrow (\text{Spec}(\mathcal{B}), \tilde{\mathcal{B}})$$

consists of a continuous groupoid homomorphism

$$f : \text{Spec}(\mathcal{A}) \longrightarrow \text{Spec}(\mathcal{B}),$$

together with a pretopos functor over $\text{Spec}(\mathcal{B})$

$$\tilde{f} : \tilde{\mathcal{B}} \longrightarrow f_* \tilde{\mathcal{A}}.$$

Theorem 8 (Awodey-Breiner) *Every pretopos functor $\mathcal{B} \longrightarrow \mathcal{A}$ induces a morphism of the associated affine logical schemes $\text{Spec}(\mathcal{A}) \longrightarrow \text{Spec}(\mathcal{B})$. Moreover, the functor*

$$\text{Spec} : \text{BoolPreTop} \longrightarrow \text{LogScheme}_{\text{aff}}^{\text{op}}$$

is full and faithful: every map of schemes comes from an essentially unique map of pretoposes.

Corollary 7 (First-order logical duality) *There is an equivalence,*

$$\text{BoolPreTop} \simeq \text{LogScheme}_{\text{aff}}^{\text{op}}.$$

The category of Boolean pretoposes is thus dual to the category of affine logical schemes. We can now start to “patch together” affine pieces of the form $(\text{Spec}(\mathcal{B}), \tilde{\mathcal{B}})$, in order to make a general notion of a “logical scheme”, consisting of a topological groupoid of structures not tied to any one theory, equipped with a sheaf of local theories, and locally equivalent to an affine scheme. The first few steps in this direction are explored in [5].

References

1. Awodey, S., Sheaf representation for topoi. *Journal of Pure and Applied Algebra*, 145, pp. 107–121, 2000.
2. Awodey, S. and H. Forssell, First-order logical duality. *Annals of Pure and Applied Logic*, 164(3), pp. 319–348, 2013.
3. Forssell, H., Topological representation of geometric theories. *Mathematical Logic Quarterly*, 58, pp. 380–393, 2012.
4. Awodey, S. and S. Breiner, Scheme representation for first-order logic. TACL 2013. Sixth International Conference on Topology, Algebra and Categories in Logic, pp. 10–13, 2014.
5. Breiner, S., *Scheme Representation for First-Order Logic*. Ph.D. thesis, Carnegie Mellon University, 2013. Available as [arXiv:1402.2600](https://arxiv.org/abs/1402.2600).
6. Butz, C. and Ieke Moerdijk, Representing topoi by topological groupoids. *Journal of Pure and Applied Algebra*, 130(3), pp. 223–235, 1998.

7. Freyd, P., Aspects of topoi, *Bulletin of the Australian Mathematical Society* 7, pp. 1–76, 1972.
8. Henkin, L., Completeness in the theory of types. *Journal of Symbolic Logic* 15, pp. 81–91, 1950.
9. Johnstone, P.T., *Stone Spaces*. Cambridge Studies in Advanced Mathematics 3, Cambridge University Press, 1982.
10. Joyal, A. and I. Moerdijk, Toposes as homotopy groupoids. *Advances in Mathematics*, 80(1), pp. 22–38, 1990.
11. Joyal, A. and M. Tierney, An extension of the Galois theory of Grothendieck. *Memoirs of the AMS*, 308, 1984.
12. Lambek, J., On the sheaf of possible worlds. In Adamek, J. and Mac Lane, S. (Eds.), *Categorical Topology*, World Scientific, Singapore, 1989.
13. Lambek, J., What is the world of mathematics? *Annals of Pure and Applied Logic*, 126, pp. 149–158, 2004.
14. Lambek, J. and I. Moerdijk, Two sheaf representations of elementary toposes. In A.S. Troelstra and D. van Dalen (Eds.), *Brouwer Centenary Symposium*, North-Holland, Amsterdam, 1982.
15. Makkai, M., Stone duality for first order logic. *Advances in Mathematics*, 65(2), pp. 97–170, 1987.
16. Mac Lane, S. and I. Moerdijk, *Sheaves in Geometry and Logic*. Universitext. Springer, 2nd edition, 1992.

Duoidally enriched Freyd categories^{*}

Chris Heunen^[0000–0001–7393–2640] and Jesse Sigal^[0000–0002–5117–8752]

School of Informatics, University of Edinburgh, United Kingdom,
 {chris.heunen, jesse.sigal}@ed.ac.uk

Abstract. Freyd categories provide a semantics for first-order effectful programming languages by capturing the two different orders of evaluation for products. We enrich Freyd categories in a duoidal category, which provides a new, third choice of parallel composition. Duoidal categories have two monoidal structures which account for the sequential and parallel compositions. The traditional setting is recovered as a full coreflective subcategory for a judicious choice of duoidal category. We give several worked examples of this uniform framework, including the parameterised state monad, basic separation semantics for resources, and interesting cases of change of enrichment.

Keywords: Freyd category · duoidal category · Kleisli category · Lawvere theory · monad

1 Introduction

Computational effects encapsulate interactions of a computer program with its environment in a modular way, and are a staple of modern programming languages [17]. Originally captured by strong monads [15], they have been extended to Arrows to deal with input as well as output [12], to Lawvere theories to better combine effects algebraically [20], to PROs and PROPs to deal with non-cartesian settings [13], and to Freyd categories to deal with effects that are not higher-order [14].

Freyd categories let one compose effectful computations both in sequence and, to some extent, in parallel, and reason about such compositions rigorously. For an effectful computation $f: a \rightarrow b$, we may embed it, the domain, and the codomain into a larger context by extending with $- \otimes c$ for any object c and monoidal-like operation \otimes , which we write as $f \otimes \text{id}: a \otimes c \rightarrow b \otimes c$. Intuitively, $f \otimes \text{id}$ does not interact with c . Effectful computations need not commute as they may alter the environment: $(f \otimes \text{id}).(\text{id} \otimes g) \neq (\text{id} \otimes g).(f \otimes \text{id})$ in general.

But what if we want to track more data about computations than just types and effects? For example, suppose we want to annotate every computation with its resource needs: there could *e.g.* be a set R of resources, and every computation f requires a certain subset $P \subseteq R$ of resources for it to execute. Sequencing two computations needs all resources to execute both, so if $f: a \rightarrow b$ and $g: b \rightarrow c$

^{*} Jesse Sigal is partly funded by Huawei.

require resources P and Q respectively, then $g.f$ requires $P \cup Q$. The same is true for parallel composition: if $f_1: a_1 \rightarrow b_1$ and $f_2: a_2 \rightarrow b_2$ require P_1 and P_2 respectively, then $f_1 \otimes f_2: a_1 \otimes a_2 \rightarrow b_1 \otimes b_2$ requires $P_1 \cup P_2$. However, it is often desirable to restrict P_1 and P_2 by requiring $P_1 \cap P_2 = \emptyset$ so that morphisms composed in parallel use different resources. If we have an identity map $\text{id}: a \rightarrow a$ for all a which requires $\emptyset \subseteq R$, then we can always form $f \otimes \text{id}$ for any f , but what of the general case?

This article proposes a solution that achieves just this: enrich Freyd categories in duoidal categories. Duoidal categories carry two interacting monoidal structures that will account for the sequential and parallel composition of both the effectful computations and the extra data we want to track, such as the resources above. We provide a concrete example for resources in Section 3.1.

Section 2 introduces duoidally enriched Freyd categories. Section 3 shows the breadth of such categories by treating disparate examples: separation semantics for resources as above, indexed state monads, and Kleisli categories of Lawvere theories. Section 4 shows that a judicious choice of duoidal enriching category recovers traditional Freyd categories as a full coreflective subcategory, and Section 5 gives an abstract characterisation of duoidally enriched Freyd categories in purely algebraic terms. Section 6 considers changing the enriching duoidal category, accounting for *e.g.* changing the underlying permission model in the example above. Section 7 concludes and suggests directions for future work.

Related work Morrison and Penneys define a \mathbf{V} -monoidal category [16] for braided monoidal \mathbf{V} as a \mathbf{V} -category with parallel composition that interacts well with the braid. In the case \mathbf{V} is braided (and thus duoidal), our definition of a \mathbf{V} -Freyd category is similar. However, we also require bifunctionality of the hom objects, an important difference for some of our constructions.

The abstract characterisation in Section 5 is inspired by Fujii's characterisation of PROs and PROPs [7] as monoids in $\mathbf{MonCat}_{\mathbf{Lax}}(\mathbf{N}^{\text{op}} \times \mathbf{N}, \mathbf{Set})$ and $\mathbf{MonCat}_{\mathbf{Lax}}(\mathbf{P}^{\text{op}} \times \mathbf{P}, \mathbf{Set})$ respectively, where \mathbf{N} and \mathbf{P} have natural numbers as objects and equalities respectively bijections as morphisms.

Garner and López Franco describe a general framework for commutativity using categories enriched in the sequential product of a duoidal category [8]. Their framework requires the duoidal category to be *normal*, meaning that the two units are isomorphic. Only with this requirement and others do they define a monoidal structure on their category of enriched categories, and do not define a monoidal enriched category. We do not require normality.

Finally, Forcey [6], and Batanin and Markl [4] enrich over duoidal categories, but using the parallel product instead. We choose to enrich over the sequential product in order to define examples in which this is the appropriate choice.

2 Duoidally enriched Freyd categories

This section introduces duoidally enriched Freyd categories (in Section 2.3), but first we discuss Freyd categories (in Section 2.1) and duoidal categories (in Section 2.2).

2.1 Freyd categories

Freyd categories provide semantics for first-order call-by-value programming languages with effects [20]. We will generalise the definition of a Freyd category slightly so that the effect free fragment need not have products, beginning with the following preliminary definitions [14,18].

Definition 1. A category \mathbf{C} is binoidal when it comes with endofunctors $(-)\ltimes x$ and $x\rtimes (-)$ for each object x such that $x\ltimes y = x\rtimes y$ for all y ; write $x\otimes y$ for this object. A morphism $f: x \rightarrow y$ is central if for any morphism $g: x' \rightarrow y'$ the two maps $(y\rtimes g).(f\ltimes x')$ and $(f\ltimes y').(x\rtimes g)$ of type $x\otimes x' \rightarrow y\otimes y'$ are equal, as are the two maps $(y'\rtimes f).(g\ltimes x)$ and $(g\ltimes y).(x'\rtimes f)$ of type $x'\otimes x \rightarrow y'\otimes y$. Central morphisms form a wide subcategory $Z(\mathbf{C})$ called the centre.

Definition 2. A binoidal category \mathbf{C} is premonoidal when equipped with an object e and families of central isomorphisms $\alpha: (x\otimes y)\otimes z \rightarrow x\otimes(y\otimes z)$, $\lambda: e\otimes x \rightarrow x$, and $\rho: x\otimes e \rightarrow x$ that are natural in each component and satisfy triangle and pentagon equations.

Definition 3. A functor $F: \mathbf{C} \rightarrow \mathbf{D}$ between premonoidal categories is a premonoidal functor when equipped with central morphisms $\eta: e_{\mathbf{D}} \rightarrow F(e_{\mathbf{C}})$ and $\mu: F(x)\otimes_{\mathbf{D}} F(y) \rightarrow F(x\otimes_{\mathbf{C}} y)$ such that μ is natural in each component, and the following diagrams commute:

$$\begin{array}{ccc}
 & e_{\mathbf{D}} \otimes_{\mathbf{D}} F(x) \xrightarrow{\eta \otimes \text{id}} F(e_{\mathbf{C}}) \otimes_{\mathbf{D}} F(x) & \\
 & \downarrow \lambda_{\mathbf{D}} & \downarrow \mu \\
 & F(x) \xleftarrow[F\lambda_{\mathbf{C}}]{} F(e_{\mathbf{C}} \otimes_{\mathbf{C}} x) & \\
 \begin{array}{c} (F(x) \otimes_{\mathbf{D}} F(y)) \otimes_{\mathbf{D}} F(z) \xrightarrow{\alpha_{\mathbf{D}}} F(x) \otimes_{\mathbf{D}} (F(y) \otimes_{\mathbf{D}} F(z)) \\ \downarrow \mu \otimes \text{id} \\ F(x \otimes_{\mathbf{C}} y) \otimes_{\mathbf{D}} F(z) \end{array} & \begin{array}{c} F(x) \otimes_{\mathbf{D}} F(y \otimes_{\mathbf{C}} z) \\ \downarrow \text{id} \otimes \mu \\ F(x \otimes_{\mathbf{C}} (y \otimes_{\mathbf{C}} z)) \end{array} & \begin{array}{c} F(x) \otimes_{\mathbf{D}} e_{\mathbf{D}} \xrightarrow{\text{id} \otimes \eta} F(x) \otimes_{\mathbf{D}} F(e_{\mathbf{C}}) \\ \downarrow \rho_{\mathbf{D}} \\ F(x) \end{array} \\
 F((x \otimes_{\mathbf{C}} y) \otimes_{\mathbf{C}} z) \xrightarrow[F\alpha_{\mathbf{C}}]{} F(x \otimes_{\mathbf{C}} (y \otimes_{\mathbf{C}} z)) & \downarrow \mu & \downarrow \mu \\
 & F(x) \xleftarrow[F\rho_{\mathbf{C}}]{} F(x \otimes_{\mathbf{C}} e_{\mathbf{C}}) &
 \end{array}$$

A premonoidal functor is strong (strict) when η and μ are isomorphisms (identities).

Note that a strict premonoidal functor F preserves associators and unitors on the nose. Recall that a functor $F: \mathbf{C} \rightarrow \mathbf{D}$ between monoidal categories is *lax monoidal* when it comes with a morphism $\eta: I \rightarrow F(I)$ and a natural transformation $\mu: F(X) \otimes F(Y) \rightarrow F(X \otimes Y)$ satisfying coherence conditions. It is *strong monoidal* when η and μ are invertible. Lax/strong monoidal functors are closed under composition. Here now is our definition of a Freyd category.

Definition 4. A Freyd category consists of a monoidal category \mathbf{M} and a premonoidal category \mathbf{C} with the same objects, and an identity-on-objects strict premonoidal functor $J: \mathbf{M} \rightarrow \mathbf{C}$ whose image lies in $Z(\mathbf{C})$. A morphism $J \rightarrow J'$ of Freyd categories consists of a strong monoidal functor $F_0: \mathbf{M} \rightarrow \mathbf{M}'$ and a strong premonoidal functor $F_1: \mathbf{C} \rightarrow \mathbf{C}'$ such that $F_1J = J'F_0$. Freyd categories and their morphisms form a category **Freyd**.

2.2 Duoidal categories

A duoidal category carries two interacting monoidal structures, that one may intuitively think of as sequential and parallel composition, but let us give the definition [2, Definition 6.1] before examples.

Definition 5. A category \mathbf{V} is duoidal when it comes with two monoidal structures $(\mathbf{V}, *, J)$ and (\mathbf{V}, \circ, I) , a natural transformation $\zeta_{A,B,C,D}: (A \circ B) * (C \circ D) \rightarrow (A * C) \circ (B * D)$, and three morphisms $\Delta: J \rightarrow J \circ J$, $\nabla: I * I \rightarrow I$, and $\epsilon: J \rightarrow I$ such that (I, ∇, ϵ) is a monoid in $(\mathbf{V}, *, J)$ and (J, Δ, ϵ) is a comonoid in (\mathbf{V}, \circ, I) , and the following diagrams commute:

$$\begin{array}{ccc}
 & J * (A \circ B) \xrightarrow{\Delta * \text{id}} (J \circ J) * (A \circ B) & \\
 ((A \circ B) * (C \circ D)) * (E \circ F) \xrightarrow{\alpha} (A \circ B) * ((C \circ D) * (E \circ F)) & \downarrow \lambda & \downarrow \zeta \\
 \zeta * \text{id} \downarrow & \downarrow \text{id} * \zeta & A \circ B \xleftarrow{\lambda \circ \lambda} (J * A) \circ (J * B) \\
 ((A * C) \circ (B * D)) * (E \circ F) & (A \circ B) * ((C * E) \circ (D * F)) & \\
 \zeta \downarrow & \downarrow \zeta & (A \circ B) * J \xrightarrow{\text{id} * \Delta} (A \circ B) * (J \circ J) \\
 ((A * C) * E) \circ ((B * D) * F) \xrightarrow{\alpha \circ \alpha} (A * (C * E)) \circ (B * (D * F)) & \rho \downarrow & \downarrow \zeta \\
 & A \circ B \xleftarrow{\rho \circ \rho} (A * J) \circ (B * J) &
 \end{array}$$

$$\begin{array}{ccc}
 & I \circ (A * B) \xleftarrow{\nabla \circ \text{id}} (I * I) \circ (A * B) & \\
 ((A \circ B) \circ C) * ((D \circ E) \circ F) \xrightarrow{\alpha * \alpha} (A \circ (B \circ C)) * (D \circ (E \circ F)) & \downarrow \lambda & \uparrow \zeta \\
 \zeta \downarrow & \downarrow \zeta & A * B \xleftarrow{\lambda * \lambda} (I \circ A) * (I \circ B) \\
 ((A \circ B) * (D \circ E)) \circ (C * F) & (A * D) \circ ((B \circ C) * (E \circ F)) & \\
 \zeta \circ \text{id} \downarrow & \downarrow \text{id} \circ \zeta & (A * B) \circ I \xleftarrow{\text{id} \circ \nabla} (A * B) \circ (I * I) \\
 ((A * D) \circ (B * E)) \circ (C * F) \xrightarrow{-\alpha} (A * D) \circ ((B * E) \circ (C * F)) & \rho \downarrow & \uparrow \zeta \\
 & A * B \xleftarrow{\rho * \rho} (A \circ I) * (B \circ I) &
 \end{array}$$

We may write $(\mathbf{V}, *, J, \circ, I)$ or $(\mathbf{V}, *, \circ)$ to be explicit about the role of each monoidal structure.

Example 1. Any braided monoidal category becomes duoidal by letting both monoidal structures coincide and ζ be the middle-four interchange $x \otimes y \otimes z \otimes w \rightarrow x \otimes z \otimes y \otimes w$ up to associativity. In particular, any symmetric or cartesian monoidal category is duoidal [2, Proposition 6.10, Example 6.19].

Example 2. If $(\mathbf{V}, *, J, \circ, I)$ is duoidal, so is $(\mathbf{V}^{\text{op}}, \circ, I, *, J)$, with opposite structure maps [2, Section 6.1.2].

Example 3. If (\mathbf{V}, \otimes, I) is a monoidal category with products, $(\mathbf{V}, \otimes, I, \times, 1)$ is duoidal with $\zeta = \langle \pi_1 \otimes \pi_1, \pi_2 \otimes \pi_2 \rangle$, $\Delta = \langle \text{id}, \text{id} \rangle$, and ∇ and ϵ terminal maps. Similarly, if a monoidal category \mathbf{V} has coproducts, $(\mathbf{V}, +, 0, \otimes, I)$ is duoidal [2, Example 6.19].

Example 4. If $(\mathbf{V}, *, J, \circ, I)$ is small and duoidal, straightforward calculation shows Day convolution [5] of each monoidal structure makes the category of presheaves $([\mathbf{V}^{\text{op}}, \mathbf{Set}], *_{\text{Day}}, \mathbf{V}(-, J), \circ_{\text{Day}}, \mathbf{V}(-, I))$ again duoidal where

$$(F *_{\text{Day}} G)(A) = \int^{B,C} \mathbf{V}(A, B * C) \times F(B) \times G(C)$$

and likewise for \circ_{Day} . An analogous construction holds for $[\mathbf{V}, \mathbf{Set}]$ by starting with \mathbf{V}^{op} .

Example 5. An endofunctor on **Set** is finitary when it preserves filtered colimits and is therefore determined on finite sets. Finitary endofunctors are closed under functor composition, \circ , with unit Id ; closed under Day convolution with products, \times_{Day} , with unit $\mathbf{Set}(1, -) \cong \text{Id}$; making $([\mathbf{Set}, \mathbf{Set}]_f, \times_{\text{Day}}, \text{Id}, \circ, \text{Id})$ a duoidal category. [8]

Example 6. For a small monoidal category (\mathbf{M}, \oplus, e) , the category of **Set**-valued endoprofunctors $\mathbf{Prof}(\mathbf{M}) := [\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{Set}]$ is duoidal $(\mathbf{Prof}(\mathbf{M}), \oplus_{\text{Day}}, \diamond)$ with profunctor composition $(P \diamond Q)(a, c) := \int^b P(a, b) \times Q(b, c)$ (having unit $\mathbf{M}(-, -)$) and Day convolution of \oplus on both sides $(P \oplus_{\text{Day}} Q)(a, b) := \int^{a_1, a_2, b_1, b_2} \mathbf{M}(a, a_1 \oplus a_2) \times \mathbf{M}(b_1 \oplus b_2, b) \times P(a_1, b_1) \times Q(a_2, b_2)$ (having unit $\mathbf{M}(-, e) \times \mathbf{M}(e, -)$). [8]

Example 7. An important example for us is the category **Subset** of *distinguished subsets*. Objects are pairs of sets (X, A) such that $X \subseteq A$ and morphisms $f: (X, A) \rightarrow (Y, B)$ are functions $f: A \rightarrow B$ with $f(X) \subseteq Y$. We call X the *distinguished subset*. Composition and identities are as in **Set**. We may suppress the distinguished subset X by writing $a \in A$ when $a \in X$. Next, we give two monoidal structures on **Subset**.

The first is the cartesian product: $(X, A) \times (Y, B) := (X \times Y, A \times B)$ on objects, and $f \times g$ as in **Set** on morphisms, with unit $(1, 1)$. Associators and unitors are as in **Set**. This is also a categorical product.

The second is the *disjunctive product*: on objects $(X, A) \otimes (Y, B)$ is defined as $(X \times Y, (A \times Y) \cup (X \times B))$ with unit $(1, 1)$. We again have $f \times g$ on morphisms, which is well-defined. Finally, the coherence maps are restricted versions of those for the cartesian product.

Now $(\mathbf{Subset}, \otimes, (1, 1), \times, (1, 1))$ is duoidal by Example 3: Δ and ∇ are unitors, ϵ is the identity, and $\zeta: ((X, A) \times (Y, B)) \otimes ((Z, C) \times (W, D)) \rightarrow ((X, A) \otimes (Z, C)) \times ((Y, B) \otimes (W, D))$ is the restricted middle-four interchange; all axioms are inherited from $(\mathbf{Set}, \times, 1)$ via Example 1.

The important difference between $(\mathbf{Subset}, \otimes, \times)$ and $(\mathbf{Set}, \times, \times)$ is that ζ is not invertible in the former (as it is not surjective as a **Set** map). This allows Freyd categories enriched in **Subset** a premonoidal-like structure.

2.3 Concrete definition

We are now ready for the titular notion of this paper. We first give a concrete definition, leaving an abstract characterisation to Section 5.

Definition 6. Let $(\mathbf{V}, *, J, \circ, I)$ be a duoidal category and (\mathbf{M}, \oplus, e) a monoidal category. A \mathbf{V} -Freyd category over \mathbf{M} consists of

- a bifunctor $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{V}$
- an extranatural family $\text{idt}: I \rightarrow \mathbf{C}(a, a)$, meaning $\mathbf{C}(\text{id}, f).\text{idt} = \mathbf{C}(f, \text{id}).\text{idt}$
- an extranatural family $\text{seq}: \mathbf{C}(a, b) \circ \mathbf{C}(b, c) \rightarrow \mathbf{C}(a, c)$, meaning seq is natural in a and c , and $\text{seq}.(\text{id} \circ \mathbf{C}(f, \text{id})) = \text{seq}.(\mathbf{C}(\text{id}, f) \circ \text{id})$
- a morphism $\text{zero}: J \rightarrow \mathbf{C}(e, e)$
- a natural family $\text{par}: \mathbf{C}(a_1, b_1) * \mathbf{C}(a_2, b_2) \rightarrow \mathbf{C}(a_1 \oplus a_2, b_1 \oplus b_2)$

satisfying the following axioms:

- (i) idt is the identity for seq , that is, $\text{seq}.(\text{idt} \circ \text{id}) = \lambda$ and symmetrically;
- (ii) seq is associative, that is, $\text{seq}.(\text{seq} \circ \text{id}) = \text{seq}.(\text{id} \circ \text{seq}).\alpha$;
- (iii) zero is the identity for par , that is, $\mathbf{C}(\lambda^{-1}, \lambda).\text{par}.(\text{zero} * \text{id}) = \lambda$ and symmetrically;
- (iv) par is associative, that is, $\mathbf{C}(\alpha^{-1}, \alpha).\text{par}.(\text{par} * \text{id}) = \text{par}.(\text{id} * \text{par}).\alpha$;
- (v) idt respects zero via $\text{idt}.\epsilon = \text{zero}$;
- (vi) idt respects par via $\text{idt}.\nabla = \text{par}.(\text{idt} * \text{idt})$;
- (vii) seq respects zero via $\text{seq}.(\text{zero} \circ \text{zero}).\Delta = \text{zero}$;
- (viii) seq respects par via $\text{seq}.(\text{par} \circ \text{par}).\zeta = \text{par}.(\text{seq} * \text{seq})$.

See Appendix A for diagrams expressing the axioms.

Definition 7. A morphism of \mathbf{V} -Freyd categories consists of a strong monoidal functor $F_0: \mathbf{M} \rightarrow \mathbf{M}'$ and a natural transformation $F_1: \mathbf{C}(a, b) \rightarrow \mathbf{C}'(F_0a, F_0b)$ satisfying:

- $F_1.\text{idt} = \text{idt}'$;
- $F_1.\text{seq} = \text{seq}'.(F_1 \circ F_1)$;
- $\mathbf{C}'(\text{id}, \mu).\text{par}'.(F_1 * F_1) = \mathbf{C}'(\mu, \text{id}).F_1.\text{par}$.

\mathbf{V} -Freyd categories and morphisms between them form a category **V-Freyd**.

Our definition differs from the duoidally enriched categories of Batanin and Markl [4] in a few important ways. They use $*$ for sequencing and \circ for parallel composition. Their analogues to axioms v to viii are $\text{idt} = \text{zero}.\epsilon$, $\text{idt} = \text{par}.(\text{idt} \circ \text{idt}).\Delta$, $\text{seq}.(\text{zero} * \text{zero}) = \text{zero}.\nabla$, and $\text{seq}.(\text{par} * \text{par}) = \text{par}.(\text{seq} \circ \text{seq}).\zeta$. Additionally, their monoidal structure is more enriched while we inherit ours from a **Set**-category, namely \mathbf{M} . Thus, we believe both notions are not inter-expressible.

3 Examples

This section works out three applications of duoidally enriched Freyd categories: resource management (in Section 3.1), indexed state (in Section 3.2), and Kleisli categories of Lawvere theories (in Section 3.3).

3.1 Stateful functions and separated monoids

To deal with resources abstractly, we first introduce the novel notion of a separated monoid.

Definition 8. A monoid (M, \bullet, e) is separated when it comes with a binary relation \parallel such that: $e \parallel m$ and $m \parallel e$; and $mm' \parallel n$ iff $m \parallel n$ and $m' \parallel n$; and $m \parallel nn'$ iff $m \parallel n$ and $m \parallel n'$.

Examples include $(\mathbb{N}, +, 0)$ with $x \parallel y$ iff $x = 0$ or $y = 0$; finite subsets $(\mathcal{P}_f(R), \cup, \emptyset)$ of a fixed set R , with $P \parallel Q$ iff $P \cap Q = \emptyset$; and products of separated monoids under pointwise separation. Separated monoids parametrise duoidal categories of resources as follows.

Definition 9. Let (M, \parallel) be a separated monoid. The category \mathbf{Label}_M of M -labelled sets has as objects functions $\ell: A \rightarrow M$ and as morphisms functions $f: A \rightarrow A'$ with $\ell' f = \ell$. This category has a monoidal structure \bullet as follows: on objects, $\ell \bullet \ell': A \times A' \rightarrow M$ sends (a, a') to $\ell(a) \bullet \ell'(a')$; on morphisms, $f \bullet f' = f \times f'$; the unit $\text{cst}_e: 1 \rightarrow M$ picks out $e \in M$. There is a second monoidal structure \parallel as follows: on objects, $\ell \parallel \ell'$ is the restriction of $\ell \bullet \ell'$ to $\{(a, a') \mid \ell(a) \parallel \ell'(a')\}$; on morphisms, $f \parallel f' = f \times f'$. The category $(\mathbf{Label}_M, \parallel, \text{cst}_e, \bullet, \text{cst}_e)$ is duoidal with $\zeta: (\ell_1 \bullet \ell'_1) \parallel (\ell_2 \bullet \ell'_2) \rightarrow (\ell_1 \parallel \ell_2) \bullet (\ell'_1 \parallel \ell'_2)$ the restricted version of the ζ for $(\mathbf{Set}, \times, 1, \times, 1)$.

Think of objects in \mathbf{Label}_M as sets of elements labelled with their resource needs. The multiplication of M combines resources, and the separation \parallel relates non-conflicting resources. We will now describe an enriched Freyd category where morphisms are labelled by resources as in the introduction.

Fix a countable family $R = \{x, y, z, \dots\}$ of sets which we think of as resources. The set $\mathcal{P}_f(R)$ of finite subsets of R is a monoid under union, and becomes a separated monoid under disjointness. For set of resources $Q \in \mathcal{P}_f(R)$, fix a product of sets $\prod_{x \in Q} x =: \Pi_Q$ which thus combines the resources in Q . Write $\pi_{Q'}: \Pi_Q \rightarrow \Pi_{Q'}$ for the projection if $Q' \subseteq Q$, and given a map $f: a \times \Pi_{Q'} \rightarrow b \times \Pi_Q$ for sets a and b , write $f^Q_{Q'}$ for the map $a \times \Pi_Q \rightarrow b \times \Pi_Q$ induced by f when $Q' \subseteq Q$ which leaves the extra resources $Q \setminus Q'$ unchanged.

We will define a $\mathbf{Label}_{\mathcal{P}_f(R)}$ -Freyd category over \mathbf{Set} of state-transforming functions. Let $\mathbf{C}(a, b)$ be the function from the disjoint union of $\mathbf{Set}(a \times \Pi_Q, b \times \Pi_Q)$ over $Q \in \mathcal{P}_f(R)$ to $\mathcal{P}_f(R)$, that sends $f: a \times \Pi_Q \rightarrow b \times \Pi_Q$ to Q . Thus, a map $f \in \mathbf{C}(a, b)$ with label Q is an effectful computation from a to b which can effect only resources in Q . This becomes a bifunctor under pre- and post-composition. Writing \cup for \bullet and \cap for \parallel for the sake of concreteness, the structure maps are:

$$\begin{aligned} \text{idt}: \text{cst}_\emptyset &\rightarrow \mathbf{C}(a, a) & \text{zero}: \text{cst}_\emptyset &\rightarrow \mathbf{C}(1, 1) \\ \star &\mapsto (\emptyset, \text{id}_{a \times 1}) & \star &\mapsto (\emptyset, \text{id}) \\ \text{seq}: \mathbf{C}(a, b) \cup \mathbf{C}(b, c) &\rightarrow \mathbf{C}(a, c) & ((P, f), (Q, g)) &\mapsto \left(P \cup Q, g_Q^{P \cup Q} \cdot f_P^{P \cup Q} \right) \end{aligned}$$

$$\begin{aligned} \mathsf{par}: \mathbf{C}(a, b) \cap \mathbf{C}(a', b') &\rightarrow \mathbf{C}(a \times a', b \times b') \\ ((Q, f), (Q', f')) &\mapsto \\ &\left(Q \cup Q', \left(\text{id} \times \langle \pi_Q, \pi_{Q'} \rangle^{-1} \right) m^{-1}.(f \times f').m.(\text{id} \times \langle \pi_Q, \pi_{Q'} \rangle) \right) \end{aligned}$$

where $\langle \pi_Q, \pi_{Q'} \rangle: \Pi_{Q \cup Q'} \rightarrow \Pi_Q \times \Pi_{Q'}$ is invertible because $Q \cap Q' = \emptyset$ and m is middle-four interchange. So par places maps in parallel up to rearranging state.

3.2 Indexed state

An important computational effect is global state. However, it is often inflexible as the type of storage remains constant over time. In this example the type can vary. We use the duoidal category of finitary endofunctors on **Set** of Example 5 to give a $[\mathbf{Set}, \mathbf{Set}]_f$ -Freyd category over **Set** based on the state monad $(s \times (-))^s$, extending Atkey's example [3]. Define $\mathbf{C}(a, b) = (b \times (-))^a$, which is a bifunctor via pre- and post-composition. The natural structure maps are:

$$\begin{aligned} \mathsf{idt}_X: X &\rightarrow (a \times X)^a & \mathsf{zero}_X: X &\rightarrow (1 \times X)^1 \\ x &\mapsto \lambda a.(x, a) & x &\mapsto \lambda \star.(x, \star) \\ \mathsf{seq}_X: \left(b \times \left((c \times X)^b \right) \right)^a &\rightarrow (c \times X)^a & f &\mapsto \text{eval}.f \\ f &\mapsto \text{eval}.f \\ \mathsf{par}_X: \int^{Y, Z} X^{Y \times Z} \times (b \times Y)^a \times (c \times Z)^{a'} &\rightarrow ((b \times c) \times X)^{a \times a'} \\ (k, f, g) &\mapsto (\text{id} \times k).m.(f \times g) \end{aligned}$$

where $\text{eval}: b \times (c \times X)^b \rightarrow c \times X$ is the evaluation map and m is the middle-four interchange. idt and seq are the unit and multiplication of a state monad but with varying types of state.

3.3 Kleisli categories of Lawvere theories

Lawvere theories model effectful computations. Functional programmers might be more familiar with Kleisli categories of monads, to which they are closely related. Here we describe an indexed version, which models independent effects in parallel. Let **Law** be the category of Lawvere theories. Its initial object is the theory \mathcal{S} of sets, the unit for the *tensor product* \otimes of Lawvere theories [10]. This makes **Law** a symmetric monoidal category, with the special property that there exist inclusion maps $\phi_i: \mathcal{L}_i \rightarrow \mathcal{L}_1 \otimes \mathcal{L}_2$. Thus the functor category $[\mathbf{Law}, \mathbf{Set}]$ is monoidal under Day convolution with unit the constant functor $\mathbf{Law}(\mathcal{S}, -) \simeq \mathbb{1}$. As this category also has products, Example 3 makes it duoidal.

Now, **Law** is equivalent to the category of finitary monads [1, Chapter 3]: any Lawvere theory \mathcal{L} induces a monad $T(\mathcal{L})$, and any map θ of Lawvere theories induces a monad morphism $T(\theta)$. Every monad T on **Set** is canonically bistrong: there are maps $\text{st}_T: a \times Tb \rightarrow T(a \times b)$ and $\text{st}'_T: Ta \times b \rightarrow T(a \times b)$ making the

two induced maps $(a \times Tb) \times c \rightarrow T((a \times b) \times c)$ equal. Each monad morphism $T(\theta)$ preserves strength: $T(\theta)_{a \times b} \circ st_{T(\mathcal{L})} = st_{T(\mathcal{L}')}.(id \times T(\theta)_b)$.

We now show a **[Law, Set]**-Freyd category over **Set** given by the Kleisli construction on Lawvere theories. Define on objects $\mathbf{C}(a, b) = T(-)(b)^a$, and on morphisms $\mathbf{C}(f, g): \mathbf{C}(a, b) \Rightarrow \mathbf{C}(a', b')$ by $\mathbf{C}(f, g)_{\mathcal{L}}(k) = T(\mathcal{L})(g).k.f$, finally:

$$\begin{aligned} idt_{\mathcal{L}}: 1 &\rightarrow T(\mathcal{L})(a)^a & zero_{\mathcal{L}}: 1 &\rightarrow T(\mathcal{L})(1)^1 \\ \star &\mapsto \eta & \star &\mapsto \eta \\ seq_{\mathcal{L}}: T(\mathcal{L})(b)^a \times T(\mathcal{L})(c)^b &\rightarrow T(\mathcal{L})(c)^a & (f, g) &\mapsto \mu.T(\mathcal{L})g.f \\ par_{\mathcal{L}}: \int^{\mathcal{L}_1, \mathcal{L}_2} \mathbf{Law}(\mathcal{L}_1 \otimes \mathcal{L}_2, \mathcal{L}) \times T(\mathcal{L}_1)(b_1)^{a_1} \times T(\mathcal{L}_2)(b_2)^{a_2} &\rightarrow T(\mathcal{L})(b_1 \times b_2)^{a_1 \times a_2} \\ (\theta, f_1, f_2) &\mapsto T(\theta).\mu.T(\mathcal{L}_1 \otimes \mathcal{L}_2)(st').st.(T(\phi_1) \times T(\phi_2)).(f_1 \times f_2) \end{aligned}$$

Intuitively, par lets us put Kleisli maps in parallel as long as their effects are forced to commute (by \otimes). So $idt_{\mathcal{L}}$ and $seq_{\mathcal{L}}$ are the identity and composition for the Kleisli category of $T(\mathcal{L})$. The definition of $par_{\mathcal{L}}$ seems noncanonical because of the use of $T(\mathcal{L}_1 \otimes \mathcal{L}_2)(st').st$, but it is not: $\mu.T(\mathcal{L}_1 \otimes \mathcal{L}_2)(st').st.(T(\phi_1) \times T(\phi_2))$ and $\mu.T(\mathcal{L}_1 \otimes \mathcal{L}_2)(st).st'.(T(\phi_1) \times T(\phi_2))$ are equal by definition of $\mathcal{L}_1 \otimes \mathcal{L}_2$.

4 Adjunction between Subset-Freyd and Freyd

Now let us explain how **V**-Freyd categories generalise Freyd categories. Our approach is similar to Power's [19] in that we work with **Subset**-enriched categories. Take **V** = **Subset** and consider a **Subset**-Freyd category $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{Subset}$; it comes equipped with a premonoidal-like structure via par and idt . We call a morphism $f \in \mathbf{C}(a, b)$ which is a member of the distinguished subset a *distinguished morphism*. We will show they are central in the premonoidal sense.

First observe that $idt: (1, 1) \rightarrow \mathbf{C}(a, a)$ is a **Subset** morphism, so $idt(\star) \in \mathbf{C}(a, a)$ is distinguished. Thus, for $g \in \mathbf{C}(a', b')$ we find $(idt(\star), g) \in \mathbf{C}(a, a) \otimes \mathbf{C}(a', b')$ by definition of \otimes . Hence the pair is in the domain of par , giving $par(idt(\star), g) \in \mathbf{C}(a \oplus a', a \oplus b')$ which we denote by $a \rtimes_{\text{par}} g$. Similarly, for any $f \in \mathbf{C}(a, b)$ we have $f \ltimes_{\text{par}} b' \in \mathbf{C}(a \oplus b', b \oplus b')$. We may also construct $f \ltimes_{\text{par}} a'$ and $b \rtimes_{\text{par}} g$. Hence it makes sense to ask if $seq(a \rtimes_{\text{par}} g, f \ltimes_{\text{par}} b') = seq(f \ltimes_{\text{par}} a', b \rtimes_{\text{par}} g)$, and if this equation (and its mirrored version by placing g on the left) holds for all f , we call g *central* in analogy to the binoidal case from Definition 1.

Next we claim that distinguished morphisms $g \in \mathbf{C}(a', b')$ are central. Note that $(idt(\star), g) \in \mathbf{C}(a', a') \times \mathbf{C}(a', b')$ and $(g, idt(\star)) \in \mathbf{C}(a', b') \times \mathbf{C}(b', b')$ are distinguished and in the domain of seq . For any $f \in \mathbf{C}(a, b)$, we have $((idt(\star), f), (g, idt(\star))) \in (\mathbf{C}(a, a) \times \mathbf{C}(a, b)) \otimes (\mathbf{C}(a', b') \times \mathbf{C}(b', b'))$ and similarly $((f, idt(\star)), (idt(\star), g)) \in (\mathbf{C}(a, b) \times \mathbf{C}(b, b)) \otimes (\mathbf{C}(a', a') \times \mathbf{C}(a', b'))$ by definition of \otimes and are thus in the domain of $seq \otimes seq$. We now apply $par.(seq \otimes seq)$ to each pair and find they equal $par(f, g)$. Axiom viii states $par.(seq \otimes seq) = seq.(par \times par).\zeta$ and therefore $seq(a \rtimes_{\text{par}} g, f \ltimes_{\text{par}} b') = par(f, g) = seq(f \ltimes_{\text{par}} a', b \rtimes_{\text{par}} g)$ (and the mirrored equation analogously), so g is central.

Distinguished morphisms have their centrality preserved by **Subset-Freyd** maps as they are mapped to distinguished morphisms, but central morphisms need not be distinguished. Thus, Definition 7 ensures that membership in the distinguished subset is preserved by **Subset-Freyd** maps, so centrality of distinguished morphisms of \mathbf{C} is preserved by all maps. Furthermore, bifunctionality of \mathbf{C} ensures that for all $f \in \mathbf{M}(a, b)$, $\mathbf{C}(\text{id}, f)(\text{idt}(\star)) \in \mathbf{C}(a, b)$, and so the image of \mathbf{M} is central and this centrality is preserved. The same is true for a Freyd category $J: \mathbf{M} \rightarrow \mathbf{C}$, the image of \mathbf{M} under J is central and this centrality is preserved by all morphisms of Freyd categories. This preservation requirement is the difference between Freyd categories and **Subset-Freyd** categories: the latter can require more central morphisms than the image of \mathbf{M} to have centrality preserved. The rest of this subsection proves that there is an adjunction between **Freyd** and **Subset-Freyd**. The left adjoint $\mathfrak{F}: \mathbf{Freyd} \rightarrow \mathbf{Subset-Freyd}$ is a free functor that only requires the image of \mathbf{M} to be preserved. The right adjoint $\mathfrak{U}: \mathbf{Subset-Freyd} \rightarrow \mathbf{Freyd}$ forgets the extra distinguished central morphisms.

Proposition 1. *There is a functor $\mathfrak{F}: \mathbf{Freyd} \rightarrow \mathbf{Subset-Freyd}$ defined on objects as $\mathfrak{F}(\mathbf{C})(a, b) = (J(\mathbf{M}(a, b)), \mathbf{C}(a, b))$ and $\mathfrak{F}(\mathbf{C})(f, g) = \mathbf{C}(Jf, Jg)$.*

Proof (Proof sketch). $\mathfrak{F}(\mathbf{C})$ is well-defined on morphisms because J is identity-on-objects, and it is bifunctional by bifunctionality of hom and functorality of J . The structure maps are:

- $\text{idt}: (1, 1) \rightarrow \mathfrak{F}(\mathbf{C})(a, a)$ is $\star \mapsto \text{id}$;
- $\text{seq}: \mathfrak{F}(\mathbf{C})(a, b) \times \mathfrak{F}(\mathbf{C})(b, c) \rightarrow \mathfrak{F}(\mathbf{C})(a, c)$ is $(f, g) \mapsto g.f$;
- $\text{zero}: (1, 1) \rightarrow \mathfrak{F}(\mathbf{C})(e, e)$ is $\star \mapsto \text{id}$;
- $\text{par}: \mathfrak{F}(\mathbf{C})(a_1, b_1) \otimes \mathfrak{F}(\mathbf{C})(a_2, b_2) \rightarrow \mathfrak{F}(\mathbf{C})(a_1 \oplus a_2, b_1 \oplus b_2)$ is $(f_1, f_2) \mapsto f_1 \otimes f_2$;
this is well-defined whether (f_1, f_2) is in $J(\mathbf{M}(a_1, b_1)) \times \mathbf{C}(a_2, b_2)$ or is in $\mathbf{C}(a_1, b_1) \times J(\mathbf{M}(a_2, b_2))$ as J preserves centrality of $\mathbf{M} = Z(\mathbf{M})$.

The (extra)naturality of the structure maps comes from the extranaturality of composition, functorality of \mathbf{M} 's monoidal product, and J being a strict premonoidal functor preserving centrality. Axioms i and ii are true by \mathbf{C} 's composition, axioms iii and iv follow from the strict premonoidality of J and the naturality of unitors and associators, and axioms v and viii are trivial. Finally, axioms vi and viii follow from \mathbf{C} 's premonoidal structure.

Finally, it is easy to check that $\mathfrak{F}(F) = F$ is well-defined and functorial.

Proposition 2. *There is a functor $\mathfrak{U}: \mathbf{Subset-Freyd} \rightarrow \mathbf{Freyd}$ that sends an object $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{Subset}$ to the functor $J: \mathbf{M} \rightarrow \mathfrak{U}(\mathbf{C})$ defined as follows:*

- the category $\mathfrak{U}(\mathbf{C})$ has the same objects as \mathbf{M} but homsets $\mathfrak{U}(\mathbf{C})(a, b) = A$ where $(X, A) := \mathbf{C}(a, b)$, with composition $g.f = \text{seq}(f, g)$, and identity $\text{id}_a = \text{idt}(\star)$;
- the functor J is the identity on objects and $J(f) = \mathbf{C}(\text{id}_a, f)(\text{idt}(\star))$ on morphisms;
- the binoidal structure on $\mathfrak{U}(\mathbf{C})$ is $a \rtimes b = a \ltimes b = a \oplus_{\mathbf{M}} b$ on objects and $a \rtimes f = \text{par}(\text{idt}(\star), f)$ and $f \ltimes b = \text{par}(f, \text{idt}(\star))$ on morphisms.

Proof (Proof sketch). It is mechanical to check that $\mathfrak{U}(\mathbf{C})$ is a well-defined Freyd category. Given a morphism $F = (F_0, F_1)$ from $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{Subset}$ to $\mathbf{C}' : \mathbf{M}'^{\text{op}} \times \mathbf{M}' \rightarrow \mathbf{Subset}$, we must define a morphism $\mathfrak{U}(F) : J_{\mathfrak{U}(\mathbf{C})} \rightarrow J_{\mathfrak{U}(\mathbf{C}')}$. We define $\mathfrak{U}(F)_0$ to be the strong monoidal functor F_0 , and define $\mathfrak{U}(F)_1$ as F_0 on objects and as F_1 on homsets. This is a well-defined morphism of Freyd categories. It is straightforward to verify that \mathfrak{U} is functorial.

Theorem 1. *The functors of Propositions 1 and 2 form an adjunction $\mathfrak{F} \dashv \mathfrak{U}$.*

Proof (Proof sketch). For the unit η of the adjunction we may take the identity as a short calculation shows that $\mathfrak{U}\mathfrak{F} = \text{Id}_{\mathbf{Freyd}}$. A second calculation shows that for a **Subset-Freyd** category $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{Subset}$, we have $\mathfrak{F}\mathfrak{U}(\mathbf{C})(a, b) = (\mathbf{C}(\text{id}, \mathbf{M}(a, b))(\text{idt}(\star)), \mathbf{C}(a, b))$, and so each component $\epsilon_{\mathbf{C}}: \mathfrak{F}\mathfrak{U}(\mathbf{C}) \rightarrow \mathbf{C}$ of the counit can be defined as $\epsilon_{\mathbf{C}0} = \text{Id}_{\mathbf{M}}$ and $\epsilon_{\mathbf{C}1} = \text{id}_{\mathbf{C}(a, b)}: \mathfrak{F}\mathfrak{U}(\mathbf{C})(a, b) \rightarrow \mathbf{C}(a, b)$. Note that the underlying **Set** map for $\epsilon_{\mathbf{C}1}$ is the identity map, but this is not an identity in **Subset**. This counit is natural, and this unit and counit satisfy the zig-zag identities for an adjunction.

Recall that an adjunction $F \dashv G$ with unit $\eta: \text{Id} \rightarrow GF$ and counit $\epsilon: FG \rightarrow \text{Id}$ is *idempotent* if any of $F\eta$, ϵF , ηG , or $G\epsilon$ are invertible [9, Section 3.8]. In the case of the previous theorem, clearly $\mathfrak{F}\eta$ is invertible as η is the identity, so this adjunction is idempotent. This leads to the following theorem detailing just how **Subset-Freyd** generalises **Freyd**.

Theorem 2. *The full coreflective subcategory of **Subset-Freyd** consisting of objects $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{Subset}$ for which $\mathbf{C}(a, b)$ has the distinguished subset $\mathbf{C}(\text{id}, \mathbf{M}(a, b))(\text{idt}(\star))$ is equivalent to **Freyd**.*

Proof (Proof sketch). The following is a general fact about idempotent adjunctions [9, Section 3.8]: if $F \dashv G$ is an idempotent adjunction with associated monad $T = GF$ and comonad $S = FG: \mathbf{A} \rightarrow \mathbf{A}$, then the category of algebras of T is equivalent to the category of coalgebras of S , and the category of coalgebras of S is a full coreflective subcategory of \mathbf{A} given by the objects of \mathbf{A} for which $\epsilon: SA \rightarrow A$ is invertible.

The category of algebras for the monad $\mathfrak{U}\mathfrak{F} = \text{Id}$ is equivalent to **Freyd**, which is therefore a full coreflective subcategory of **Subset-Freyd**. Furthermore, we can characterize the objects of this subcategory as **Subset-Freyd** categories \mathbf{C} for which $\epsilon: \mathfrak{F}\mathfrak{U}(\mathbf{C}) \rightarrow \mathbf{C}$ is invertible. Concretely, this means $\epsilon_{\mathbf{C}1}$ must be invertible in **Subset**. But the underlying **Set** map is the identity, establishing the claim.

5 Abstract characterisation

Definition 6 is a very concrete way to specify a **V-Freyd** category, involving a nontrivial amount of data and axioms. Yet it fits together, as we show in this subsection by giving a characterisation in the style of [12]. Recall that a natural transformation between lax monoidal functors is *monoidal* when it respects

the coherence maps μ and η . Write $\mathbf{MonCat}_{\text{lax}}(\mathbf{C}, \mathbf{D})$ for the category of lax monoidal functors from \mathbf{C} to \mathbf{D} and monoidal natural transformations between them. If \mathbf{A} and \mathbf{B} are monoidal categories, so are \mathbf{A}^{op} and $\mathbf{A} \times \mathbf{B}$, with componentwise structure. Thus we may consider $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V})$ for the monoidal category $(\mathbf{V}, *, J)$. We will lift the other monoidal structure (\mathbf{V}, \circ, I) to $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V})$ and prove that a \mathbf{V} -Freyd category is exactly a monoid with respect to this monoidal structure, under additional assumptions on \mathbf{V} . Most proofs are deferred to Appendix B.

Definition 10. A duoidal category \mathbf{V} is a cocomplete duoidal category if \mathbf{V} is cocomplete and $*$ and \circ are cocontinuous in each argument. In a cocomplete duoidal category, the following diagrams and their symmetric versions commute:

$$\begin{array}{ccc} J * \text{colim}(D) & \xleftarrow{\sim} & \text{colim}(J * D) \\ \swarrow \cong \quad \nearrow \cong & & \\ \text{colim}(D) & & \end{array} \qquad \begin{array}{ccc} I \circ \text{colim}(D) & \xleftarrow{\sim} & \text{colim}(I \circ D) \\ \swarrow \cong \quad \nearrow \cong & & \\ \text{colim}(D) & & \end{array}$$

where the top isomorphism is colimit preservation and the others are induced by unitors.

The rest of this subsection assumes that \mathbf{V} is a cocomplete duoidal category; importantly, this is satisfied for presheaf categories. This restriction will be mitigated in Section 6.2 for small \mathbf{V} . We also assume that \mathbf{M} is small. All laxness is with respect to $(\mathbf{V}, *, J)$. We now lift (\mathbf{V}, \circ, I) ; first the unit, then composition.

Proposition 3. There is a lax monoidal functor $\underline{\text{hom}}_{\mathbf{M}}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{V}$ defined on objects as $\underline{\text{hom}}_{\mathbf{M}}(a, b) = \coprod_{\sigma \in \text{hom}_{\mathbf{M}}(a, b)} I$.

Proposition 4. If $S, T: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{V}$ are lax monoidal functors, the functor $S \hat{\diamond} T: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{V}$ defined using coends as $(S \hat{\diamond} T)(a, c) = \int^b T(a, b) \circ S(b, c)$ is lax monoidal.

Proposition 5. $(\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V}), \hat{\diamond}, \underline{\text{hom}}_{\mathbf{M}})$ is a monoidal category.

Proof. Lemmas 5 to 7 in Appendix B show that the \circ -composition is functorial, associative, and has $\underline{\text{hom}}_{\mathbf{M}}$ as left and right unit. That leaves only the triangle and pentagon identities, which follow from cocontinuity and the equivalent identities for \circ .

With these preparations we can characterise \mathbf{V} -Freyd categories abstractly.

Theorem 3. Let \mathbf{V} be a cocomplete duoidal category. Then a \mathbf{V} -Freyd category $\mathbf{C}: \mathbf{M} \times \mathbf{M}^{\text{op}} \rightarrow \mathbf{V}$ is exactly a monoid in $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V})$.

Proof (Proof sketch). A monoid \mathbf{C} in $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V})$ consists of two maps $e: \underline{\text{hom}}_{\mathbf{M}} \rightarrow \mathbf{C}$ and $m: \mathbf{C} \hat{\diamond} \mathbf{C} \rightarrow \mathbf{C}$, inducing idt and seq satisfying unit and associativity conditions. The lax monoidal structure of \mathbf{C} gives zero and par respectively, so identity and associativity conditions follow. Finally, the components of e and m are monoidal natural transformations, ensuring that idt and seq respect zero and par .

We note that by Fujii's observations [7], PROs and PROPs are equivalent to **Set**-Freyd categories over **N** and **P** respectively because $(\mathbf{Set}, \times, \times)$ is a cocomplete duoidal category.

6 Change of enrichment

After defining enriched categories, a natural next step is to consider a change of enrichment. Any monoidal functor $\mathbf{V} \rightarrow \mathbf{W}$ induces a functor $\mathbf{V}\text{-Cat} \rightarrow \mathbf{W}\text{-Cat}$. We will show that the same holds for the appropriate type of functors between duoidal categories and enriched Freyd categories (in Section 6.1). We will then use that to alleviate the restriction of duoidal cocompleteness on the abstract characterisation of Section 5 (in Section 6.2) at the cost of losing a direction of the correspondence. Finally, changing enrichment along a forgetful functor gives an underlying (unenriched) Freyd category $J: \mathbf{M} \rightarrow \mathbf{C}$ with \mathbf{C} monoidal, which we show recovers the pure computations in the examples of Section 3 (in Section 6.3).

6.1 Lifting duoidal functors

To talk about change of enrichment, we first need to define the appropriate type of functor between the enriching categories along which to change.

Definition 11. [2, Definition 6.54] Take duoidal categories $(\mathbf{V}, *_\mathbf{V}, J_\mathbf{V}, \circ_\mathbf{V}, I_\mathbf{V})$ and $(\mathbf{W}, *_\mathbf{W}, J_\mathbf{W}, \circ_\mathbf{W}, I_\mathbf{W})$. A functor $F: \mathbf{V} \rightarrow \mathbf{W}$ is a double lax monoidal functor when equipped with η_* , μ_* , η_\circ , and μ_\circ such that (F, η_*, μ_*) is lax monoidal for $*_\mathbf{V}$ and $*_\mathbf{W}$, $(F, \eta_\circ, \mu_\circ)$ is lax monoidal for $\circ_\mathbf{V}$ and $\circ_\mathbf{W}$, and the following diagrams commute:

$$\begin{array}{ccc}
 (F(A) \circ_\mathbf{W} F(B)) *_\mathbf{W} (F(C) \circ_\mathbf{W} F(D)) & \xrightarrow{\zeta} & (F(A) *_\mathbf{W} F(C)) \circ_\mathbf{W} (F(B) *_\mathbf{W} F(D)) \\
 \downarrow \mu_\circ * \mu_\circ & & \downarrow \mu_* * \mu_* \\
 F(A \circ_\mathbf{V} B) *_\mathbf{W} F(C \circ_\mathbf{V} D) & & F(A *_\mathbf{V} C) \circ_\mathbf{W} F(B *_\mathbf{V} D) \\
 \downarrow \mu_* & & \downarrow \mu_\circ \\
 F((A \circ_\mathbf{V} B) *_\mathbf{V} (C \circ_\mathbf{V} D)) & \xrightarrow{F\zeta} & F((A *_\mathbf{V} C) \circ_\mathbf{V} (B *_\mathbf{V} D))
 \end{array}$$

$$\begin{array}{ccc}
 J_\mathbf{W} & \xrightarrow{\eta_*} & F(J_\mathbf{V}) \xrightarrow{F\Delta} F(J_\mathbf{V} \circ_\mathbf{V} J_\mathbf{V}) & I_\mathbf{W} & \xrightarrow{\eta_\circ} & F(I_\mathbf{V}) \xleftarrow{F\nabla} F(I_\mathbf{V} *_\mathbf{V} I_\mathbf{V}) \\
 \downarrow \Delta & & \uparrow \mu_\circ & \nabla & & \uparrow \mu_* \\
 J_\mathbf{W} \circ_\mathbf{W} J_\mathbf{W} & \xrightarrow{\eta_* * \eta_*} & F(J_\mathbf{V}) \circ_\mathbf{W} F(J_\mathbf{V}) & I_\mathbf{W} *_\mathbf{W} I_\mathbf{W} & \xrightarrow{\eta_\circ * \eta_\circ} & F(I_\mathbf{V}) *_\mathbf{W} F(I_\mathbf{V})
 \end{array}$$

Here now is the change-of-enrichment theorem for duoidally enriched Freyd categories.

Theorem 4. Let $F: \mathbf{V} \rightarrow \mathbf{W}$ be a double lax monoidal functor. For a \mathbf{V} -Freyd category $\mathbf{C}: \mathbf{M}^{\text{op}} \times \mathbf{M} \rightarrow \mathbf{V}$, define $\overline{F}(\mathbf{C})(a, b) := F(\mathbf{C}(a, b))$ with structure maps $\text{id}_F := F\text{idt}.\eta_\circ$, $\text{seq}_F := F\text{seq}.\mu_\circ$, $\text{zero}_F := F\text{zero}.\eta_*$, and $\text{par}_F := F\text{par}.\mu_*$. For a map $G = (G_0, G_1): \mathbf{C} \rightarrow \mathbf{C}'$, define $\overline{F}(G) := (G_0, FG_1)$. This \overline{F} is a functor $\mathbf{V}\text{-Freyd} \rightarrow \mathbf{W}\text{-Freyd}$.

Proof. See Appendix C.

Example 8. Let M and N be separated monoids and $\phi: M \rightarrow N$ a homomorphism such that $\phi(m) \parallel \phi(m')$ implies $m \parallel m'$. Then ϕ induces a double lax monoidal functor $\phi_*: \mathbf{Label}_M \rightarrow \mathbf{Label}_N$ given by $\ell \mapsto \phi.\ell$ on objects and $f \mapsto f$ on morphisms. The maps η_* , μ_* , and η_\circ are all identities, while $\mu_\circ: \{(a, a') \mid \phi.\ell(a) \parallel \phi.\ell'(a')\} \rightarrow \{(a, a') \mid \ell(a) \parallel \ell'(a')\}$ is the inclusion, and so ϕ_* is clearly double lax monoidal. Apply Theorem 4 to the example from Section 3.1 along the map $\mathcal{P}_f(!): \mathcal{P}_f(R) \rightarrow \mathcal{P}_f(1)$, which is a homomorphism such that $\mathcal{P}_f(!)(P) \cap \mathcal{P}_f(!)(Q) = \emptyset$ implies $P \cap Q = \emptyset$. We get $\mathcal{P}_f(!)_*(\mathbf{C})(a, b) = \sum_{Q \in \mathcal{P}_f(R)} (\mathbf{Set}(a \times \Pi_Q, b \times \Pi_Q)) \rightarrow \mathcal{P}_f(1)$, $(Q, f) \mapsto \emptyset$ if $Q = \emptyset$, else 1. This change of enrichment alters the example to only allowing maps to be put in parallel if at least one of them requires no resources.

Example 9. We can use change of enrichment for the indexed state example of Section 3.2. Consider Example 6 for $(\mathbf{Set}, \times, 1)$ (using universes for this example to avoid size issues). There, the definition of Day convolution \times_{Day} simplifies to $(P \times_{\text{Day}} Q)(a, b) = \int^{b_2, b_2} \mathbf{Set}(b_1 \times b_2, b) \times P(a, b_1) \times Q(a, b_2)$ and its unit becomes $k(a, b) = b$. The Kleisli construction turns a finitary endofunctor on \mathbf{Set} into a profunctor as follows. Define $\text{Kl}: [\mathbf{Set}, \mathbf{Set}]_f \rightarrow \mathbf{Prof}(\mathbf{Set})$ by $\text{Kl}(F)(a, b) = \mathbf{Set}(a, Fb)$, and coherence maps:

$$\begin{array}{ll} \eta_*: k \rightarrow \text{Kl}(\text{Id}) & \mu_*: \text{Kl}(F_1) \times_{\text{Day}} \text{Kl}(F_2) \rightarrow \text{Kl}(F_1 \times_{\text{Day}} F_2) \\ b \mapsto \text{cst}_b & (k, f_1, f_2) \mapsto \lambda a. (k, f_1(a), f_2(a)) \\ \\ \eta_\circ: \text{hom} \rightarrow \text{Kl}(\text{Id}) & \mu_\circ: \text{Kl}(F) \diamond \text{Kl}(G) \rightarrow \text{Kl}(F \circ G) \\ f \mapsto f & (f, g) \mapsto Fg.f \end{array}$$

This makes Kl a double lax monoidal functor. Theorem 4 then gives a $\mathbf{Prof}(\mathbf{Set})$ -Freyd category defined by $\text{Kl}(\mathbf{C})(a, b)(x, y) := \mathbf{Set}(x, (b \times y)^a)$.

6.2 Yoneda embedding

The Yoneda embedding of a small monoidal category is a strong monoidal functor with respect to Day convolution. This extends to small duoidal categories.

Proposition 6. *The Yoneda embedding $\mathbf{V} \rightarrow [\mathbf{V}^{\text{op}}, \mathbf{Set}]$ is a double lax monoidal functor from small $(\mathbf{V}, *, J, \circ, I)$ to $([\mathbf{V}^{\text{op}}, \mathbf{Set}], *_{\text{Day}}, \mathbf{V}(-, J), \circ_{\text{Day}}, \mathbf{V}(-, I))$.*

Proof. See [11] for the fact that it is lax monoidal for each monoidal structure separately. The diagrams of Definition 11 are verified straightforwardly.

It follows from Theorem 4 that every \mathbf{V} -Freyd category for small \mathbf{V} induces a $[\mathbf{V}^{\text{op}}, \mathbf{Set}]$ -Freyd category. But $[\mathbf{V}^{\text{op}}, \mathbf{Set}]$ is duoidally cocomplete, so the setting in which the abstract characterisation of Theorem 3 applies. We conclude that the characterisation extends beyond the duoidally cocomplete setting in the sense that every \mathbf{V} -Freyd category for small \mathbf{V} induces a monoid in $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, [\mathbf{V}^{\text{op}}, \mathbf{Set}])$.

6.3 Forgetful functors

Any category enriched in a monoidal category \mathbf{V} has an underlying (unenriched) category, got by changing the enrichment along the ‘forgetful’ monoidal functor $\mathbf{V}(I, -): \mathbf{V} \rightarrow \mathbf{Set}$. A similar process plays out for duoidal categories.

Proposition 7. *Let $(\mathbf{V}, *, J, \circ, I)$ be a duoidal category and write $\phi: J \rightarrow J * J$ for the inverse of the unitors. Then $\mathbf{V}(J, -): \mathbf{V} \rightarrow \mathbf{Set}$ is a double lax monoidal functor with coherence maps:*

$$\begin{array}{ll} \eta_*: 1 \rightarrow \mathbf{V}(J, J) & \mu_*: \mathbf{V}(J, A_1) \times \mathbf{V}(J, A_2) \rightarrow \mathbf{V}(J, A_1 * A_2) \\ \star \mapsto \text{id} & (f_1, f_2) \mapsto (f_1 * f_2). \phi \\ \eta_\circ: 1 \rightarrow \mathbf{V}(J, I) & \mu_\circ: \mathbf{V}(J, A_1) \times \mathbf{V}(J, A_2) \rightarrow \mathbf{V}(J, A_1 \circ A_2) \\ \star \mapsto \epsilon & (f_1, f_2) \mapsto (f_1 \circ f_2). \Delta \end{array}$$

Applying Theorem 4 along the forgetful functor of the previous proposition in the case of the examples of Section 3 will show that this recovers the underlying pure computations. Note that a **Set**-Freyd category \mathbf{C} has a trivial instance of the exchange axiom, axiom viii, and so \mathbf{C} is a monoidal category with identity-on-objects monoidal functor $J: \mathbf{M} \rightarrow \mathbf{C}$.

Example 10. Applying the forgetful functor to the stateful function example of Section 3.1 results in the (unenriched) category with $\mathbf{Label}_{\mathcal{P}_f(R)}(\text{cst}_\emptyset, \mathbf{C}(a, b))$ as the homsets. Because labels are preserved, the morphisms in this (unenriched) category are exactly the elements of $\mathbf{C}(a, b)$ which have label \emptyset , i.e. maps $a \times 1 \rightarrow b \times 1$ which are pure functions.

Example 11. Changing the enrichment of the indexed state example from Section 3.2 along the forgetful functor gives the (unenriched) category with homsets $[\mathbf{Set}, \mathbf{Set}]_f(\text{Id}, \mathbf{C}(a, b))$. If $\phi: \text{Id} \rightarrow (b \times (-))^a$ is such a natural transformation, then the function $\phi_1: 1 \rightarrow (b \times 1)^a$, which is equivalent to choosing a function $f: a \rightarrow b$, completely determines ϕ , because for any set X and $x \in X$ by naturality $1 \xrightarrow{x} X \xrightarrow{\phi_X} (b \times X)^a = 1 \xrightarrow{\phi_1} (b \times 1)^a \xrightarrow{(\text{id} \times x).-} (b \times X)^a$, whence $\phi_X(x)(a) = (f(a), x)$. Therefore the morphisms in this (unenriched) category are all functions $a \rightarrow b$.

Example 12. Changing the enrichment of the Kleisli categories of Lawvere theories example from Section 3.3 along the forgetful functor gives the (unenriched) category with homsets $[\mathbf{Law}, \mathbf{Set}](\mathbb{1}, \mathbf{C}(a, b))$. Consider such a natural transformation $\phi: \mathbb{1} \rightarrow T(-)(b)^a$. It is completely determined by its component at \mathcal{S} . For any \mathcal{L} let $\iota: \mathcal{S} \rightarrow \mathcal{L}$ be the unique map, then naturality implies $\phi_{\mathcal{L}} = T(\iota)\phi_{\mathcal{S}}$. Furthermore, $\phi_{\mathcal{S}}(\star) \in T(\mathcal{S})(b)^a = b^a$. So the morphisms in this (unenriched) category again are all functions $a \rightarrow b$.

7 Conclusion

We have defined a version of Freyd categories enriched over any duoidal category \mathbf{V} , and morphisms between them. We used various duoidal categories to give examples based on separation of resources, parameterised monads, and the Kleisli construction for Lawvere theories. By enriching with **Subset**, we have proven that the category of Freyd categories **Freyd** is a full coreflective subcategory of **Subset-Freyd**, thus establishing that \mathbf{V} -Freyd categories indeed generalise Freyd categories. Additionally, we proved an abstract characterisation of \mathbf{V} -Freyd categories over small \mathbf{M} for duoidally cocomplete \mathbf{V} , they are monoids in $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V})$. Finally, we provided change of enrichment and examples thereof.

Future work There are several directions for further investigation:

- The abstract characterisation of Section 5 may be part of a larger structure, namely a *bicategory with proarrow equipment*, whose objects are monoidal categories, arrows are strong monoidal functors, proarrows are lax monoidal profunctors, and cells are lax monoidal natural transformations. In this setting, a \mathbf{V} -Freyd category would be a monad and the vertical monad morphisms would be a \mathbf{V} -Freyd morphism. This would enable applying general constructions for monads in a bicategory.
- Relatedly, an *fc-multicategory* structure on $\mathbf{MonCat}_{\text{lax}}(\mathbf{M}^{\text{op}} \times \mathbf{M}, \mathbf{V})$ may bypass cocompleteness in characterising \mathbf{V} -Freyd categories as monoids.
- The abstract characterisation of Section 5 also uses the free \mathbf{V} -category on \mathbf{M} . It may be fruitful to change the definition of a \mathbf{V} -Freyd category to be a \mathbf{V} -functor $J: \mathbf{M} \rightarrow \mathbf{C}$ where we extend \mathbf{V} -categories in a way similar to Morrison and Penneys [16].
- Freyd categories can have the property of being *closed*. In this case they induce a strong monad. A similar definition may be possible for \mathbf{V} -Freyd categories. This could determine a higher-order semantics for effectful programs based on duoidal categories. A nontrivial definition of closure may require a \mathbf{V} -category \mathbf{M} that is not free.
- Our original motivation stemmed from the desire for semantics combining differentiable and probabilistic programming, in particular, the possibility of having a linear structure for the probabilistic fragment and a cartesian one for differentiable terms. **Prof**-Freyd categories may provide a useful separation to aid the desired distinction between linear and cartesian properties.

Acknowledgments We would like to thank Robin Kaarsgaard, Ohad Kammar, and Matthew Di Meglio for their input and encouragement, as well as the reviewers of all versions of this work.

References

1. Adamek, J., Rosicky, J.: Locally presentable and accessible categories. Cambridge University Press (1994). <https://doi.org/10.1017/CBO9780511600579>

2. Aguiar, M., Mahajan, S.: Monoidal Functors, Species and Hopf Algebras. American Mathematical Society (2010). <https://doi.org/10.1090/crmm/029>
3. Atkey, R.: Algebras for parameterised monads. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) Algebra and Coalgebra in Computer Science, Lecture Notes in Computer Science, vol. 5728, pp. 3–17. Springer (2009). https://doi.org/10.1007/978-3-642-03741-2_2
4. Batanin, M., Markl, M.: Centers and homotopy centers in enriched monoidal categories. Advances in Mathematics **230**, 1811–1858 (2012). <https://doi.org/10.1016/j.aim.2012.04.011>
5. Day, B.: On closed categories of functors. In: Midwest Category Seminar. Lecture Notes in Mathematics, vol. 137, pp. 1–38 (1970)
6. Forcey, S.: Enrichment over iterated monoidal categories. Algebraic & Geometric Topology **4**, 95–119 (2004). <https://doi.org/10.2140/agt.2004.4.95>
7. Fujii, S.: A unified framework for notions of algebraic theory. Theory and Applications of Categories **34**(40), 1246–1316 (2019)
8. Garner, R., López Franco, I.: Commutativity. Journal of Pure and Applied Algebra **220**(5), 1707–1751 (2016). <https://doi.org/10.1016/j.jpaa.2015.09.003>
9. Grandis, M.: Category theory and applications: a textbook for beginners. World Scientific, 2 edn. (Apr 2021). <https://doi.org/10.1142/12253>
10. Hyland, M., Plotkin, G., Power, J.: Combining effects: sum and tensor. Theor. Comput. Sci. **357**(1–3), 70–99 (2006). <https://doi.org/10.1016/j.tcs.2006.03.013>
11. Im, G.B., Kelly, G.M.: A universal property of the convolution monoidal structure. Journal of Pure and Applied Algebra **43**, 75–88 (1986). [https://doi.org/10.1016/0022-4049\(86\)90005-8](https://doi.org/10.1016/0022-4049(86)90005-8)
12. Jacobs, B., Heunen, C., Hasuo, I.: Categorical semantics for Arrows. Journal of Functional Programming **19**(3–4), 403–438 (2009). <https://doi.org/10.1017/S0956796809007308>
13. Lack, S.: Composing PROPs. Theory and Applications of Categories **13**(9), 147–163 (2004)
14. Levy, P.B., Power, J., Thielecke, H.: Modelling environments in call-by-value programming languages. Information and Computation **185**(2), 182–210 (2003). [https://doi.org/10.1016/S0890-5401\(03\)00088-9](https://doi.org/10.1016/S0890-5401(03)00088-9)
15. Moggi, E.: Notions of computation and monads. Information and Computation **93**, 55–92 (1991). [https://doi.org/10.1016/0890-5401\(91\)90052-4](https://doi.org/10.1016/0890-5401(91)90052-4)
16. Morrison, S., Penneys, D.: Monoidal categories enriched in braided monoidal categories. International Mathematical Research Notes **11**, 3527–3579 (2019). <https://doi.org/10.1093/imrn/rnx217>
17. Plotkin, G., Power, J.: Computational effects and operations: an overview. In: Domains. Electronic Notes in Theoretical Computer Science, vol. 73, pp. 149–163 (2004). <https://doi.org/10.1016/j.entcs.2004.08.008>
18. Power, J., Robinson, E.: Premonoidal categories and notions of computation. Mathematical Structures in Computer Science **7**(5), 453–468 (1997). <https://doi.org/10.1017/S0960129597002375>
19. Power, J.: Premonoidal categories as categories with algebraic structure. Theoretical Computer Science **278**(1-2), 303–321 (May 2002). [https://doi.org/10.1016/S0304-3975\(00\)00340-6](https://doi.org/10.1016/S0304-3975(00)00340-6), publisher: Elsevier
20. Staton, S.: Freyd categories are enriched Lawvere theories. In: Proceedings of the Workshop on Algebra, Coalgebra and Topology. Electronic

Notes in Theoretical Computer Science, vol. 303, pp. 197–206 (2014).
<https://doi.org/10.1016/j.entcs.2014.02.010>

A Definition of V-Freyd category

This appendix spells out the type diagrams of Definition 6 of V-Freyd categories.

Extranaturality of idt:

$$\begin{array}{ccc} I & \xrightarrow{\text{idt}} & \mathbf{C}(b, b) \\ \text{idt} \downarrow & & \downarrow \mathbf{C}(f, \text{id}) \\ \mathbf{C}(a, a) & \xrightarrow[\mathbf{C}(\text{id}, f)]{} & \mathbf{C}(a, b) \end{array}$$

Extranaturality of seq:

$$\begin{array}{ccc} \mathbf{C}(a, b) \circ \mathbf{C}(b', c) & \xrightarrow{\mathbf{C}(\text{id}, f) \circ \text{id}} & \mathbf{C}(a, b') \circ \mathbf{C}(b', c) \\ \text{id} \circ \mathbf{C}(f, \text{id}) \downarrow & & \downarrow \text{seq} \\ \mathbf{C}(a, b) \circ \mathbf{C}(b, c) & \xrightarrow[\text{seq}]{} & \mathbf{C}(a, c) \end{array}$$

idt is the identity for seq:

$$\begin{array}{ccc} I \circ \mathbf{C}(a, b) & \xrightarrow{\text{idt} \circ \text{id}} & \mathbf{C}(a, a) \circ \mathbf{C}(a, b) & \mathbf{C}(a, b) \circ I & \xrightarrow{\text{id} \circ \text{idt}} & \mathbf{C}(a, b) \circ \mathbf{C}(b, b) \\ & \searrow \lambda & \downarrow \text{seq} & & \searrow \rho & \downarrow \text{seq} \\ & & \mathbf{C}(a, b) & & & \mathbf{C}(a, b) \end{array}$$

seq is associative:

$$\begin{array}{ccc} (\mathbf{C}(a, b) \circ \mathbf{C}(b, c)) \circ \mathbf{C}(c, d) & \xrightarrow{\alpha} & \mathbf{C}(a, b) \circ (\mathbf{C}(b, c) \circ \mathbf{C}(c, d)) \\ \text{seq} \circ \text{id} \downarrow & & \downarrow \text{id} \circ \text{seq} \\ \mathbf{C}(a, c) \circ \mathbf{C}(c, d) & & \mathbf{C}(a, b) \circ \mathbf{C}(b, d) \\ \searrow \text{seq} & & \swarrow \text{seq} \\ & \mathbf{C}(a, d) & \end{array}$$

zero is the identity for par:

$$\begin{array}{ccc} J * \mathbf{C}(a, b) & \xrightarrow{\text{zero} * \text{id}} & \mathbf{C}(e, e) * \mathbf{C}(a, b) & \mathbf{C}(a, b) * J & \xrightarrow{\text{id} * \text{zero}} & \mathbf{C}(a, b) * \mathbf{C}(e, e) \\ \lambda \downarrow & & \downarrow \text{par} & \rho \downarrow & & \downarrow \text{par} \\ \mathbf{C}(a, b) & \xleftarrow[\mathbf{C}(\lambda^{-1}, \lambda)]{} & \mathbf{C}(e \oplus a, e \oplus b) & \mathbf{C}(a, b) & \xleftarrow[\mathbf{C}(\rho^{-1}, \rho)]{} & \mathbf{C}(b \oplus e, b \oplus e) \end{array}$$

par is associative:

$$\begin{array}{ccc}
 (\mathbf{C}(a_1, b_1) * \mathbf{C}(a_2, b_2)) * \mathbf{C}(a_3, b_3) & \xrightarrow{\alpha} & \mathbf{C}(a_1, b_1) * (\mathbf{C}(a_2, b_2) * \mathbf{C}(a_3, b_3)) \\
 \downarrow \text{par*id} & & \downarrow \text{id*par} \\
 \mathbf{C}(a_1 \oplus a_2, b_1 \oplus b_2) * \mathbf{C}(a_3, b_3) & & \mathbf{C}(a_1, b_1) * \mathbf{C}(a_2 \oplus a_3, b_2 \oplus b_3) \\
 \downarrow \text{par} & & \downarrow \text{par} \\
 \mathbf{C}((a_1 \oplus a_2) \oplus a_3, (b_1 \oplus b_2) \oplus b_3) & \xrightarrow[\mathbf{C}(\alpha^{-1}, \alpha)]{} & \mathbf{C}(a_1 \oplus (a_2 \oplus a_3), b_1 \oplus (b_2 \oplus b_3))
 \end{array}$$

idt respects zero:

$$\begin{array}{ccc}
 J & \xrightarrow{\epsilon} & I \\
 \searrow \text{zero} & & \downarrow \text{idt} \\
 & & \mathbf{C}(e, e)
 \end{array}$$

idt respects par:

$$\begin{array}{ccc}
 I * I & \xrightarrow{\text{idt}*idt} & \mathbf{C}(a, a) * \mathbf{C}(b, b) \\
 \nabla \downarrow & & \downarrow \text{par} \\
 I & \xrightarrow[\text{idt}]{} & \mathbf{C}(a \oplus b, a \oplus b)
 \end{array}$$

seq respects zero:

$$\begin{array}{ccc}
 J & \xrightarrow{\Delta} & J \circ J \\
 \downarrow \text{zero} & & \downarrow \text{zero} \circ \text{zero} \\
 \mathbf{C}(e, e) & \xleftarrow[\text{seq}]{} & \mathbf{C}(e, e) \circ \mathbf{C}(e, e)
 \end{array}$$

seq respects par:

$$\begin{array}{ccc}
 (\mathbf{C}(a_1, b_1) \circ \mathbf{C}(b_1, c_1)) * (\mathbf{C}(a_2, b_2) \circ \mathbf{C}(b_2, c_2)) & \xrightarrow{\zeta} & (\mathbf{C}(a_1, b_1) * \mathbf{C}(a_2, b_2)) \circ (\mathbf{C}(b_1, c_1) * \mathbf{C}(b_2, c_2)) \\
 \downarrow \text{seq*seq} & & \downarrow \text{par*par} \\
 \mathbf{C}(a_1, c_1) * \mathbf{C}(a_2, c_2) & & \mathbf{C}(a_1 \oplus a_2, b_1 \oplus b_2) \circ \mathbf{C}(b_1 \oplus b_2, c_1 \oplus c_2) \\
 \searrow \text{par} & & \swarrow \text{seq} \\
 & \mathbf{C}(a_1 \oplus a_2, c_1 \oplus c_2) &
 \end{array}$$

B Proofs for abstract characterisation

This appendix contains proofs of the abstract characterisation of \mathbf{V} -Freyd categories of Section 5. They rely on properties of \mathbf{V} -Freyd categories listed in the following four lemmas, that are mechanical to verify.

Lemma 1. *The unitors of \circ respect zero and par:*

$$\begin{array}{ll}
 \rho.\text{zero} = (\text{zero} \circ \epsilon).\Delta & \text{zero}.\lambda = (\epsilon \circ \text{zero}).\Delta \\
 \rho.\text{par} = (\text{par} \circ \nabla).\zeta.(\rho * \rho) & \text{par}.\lambda = (\text{par} \circ \nabla).\zeta.(\lambda * \lambda)
 \end{array}$$

Lemma 2. *The associator of \circ respects zero and par:*

$$\begin{aligned}\alpha.(\text{zero} \circ (\text{zero} \circ \text{zero})).(\text{id} \circ \Delta).\Delta &= ((\text{zero} \circ \text{zero}) \circ \text{zero}).(\Delta \circ \text{id}).\Delta \\ \alpha.(\text{par} \circ (\text{par} \circ \text{par})).(\text{id} \circ \zeta).\zeta &= ((\text{par} \circ \text{par}) \circ \text{par}).(\zeta \circ \text{id}).\zeta.(\alpha * \alpha)\end{aligned}$$

Lemma 3. *The unitors of $*$ respect zero and par:*

$$\begin{aligned}\text{id} &= (\text{par} \circ \text{par}).\zeta.(\text{id} * ((\text{zero} \circ \text{zero}).\Delta)).\rho \\ \text{id} &= (\text{par} \circ \text{par}).\zeta.(((\text{zero} \circ \text{zero}).\Delta) * \text{id}).\lambda\end{aligned}$$

Lemma 4. *The associator of $*$ respects par:*

$$\begin{aligned}((\text{par}.(\text{par} * \text{id})) \circ (\text{par}.(\text{par} * \text{id}))).\zeta.(\zeta * \text{id}) &= \\ ((\text{par}.(\text{id} * \text{par})) \circ (\text{par}.(\text{id} * \text{par}))).\zeta.(\text{id} * \zeta).\alpha &\end{aligned}$$

The previous lemmas require all the axioms of a duoidal category between them, except for \circ being a monoid in $(\mathbf{V}, *, J)$. This latter property is used in the abstract characterisation.

Proof (Proof of Proposition 3). Bifunctionality is inherited from $\underline{\text{hom}}_{\mathbf{M}}$. The coherence morphisms making it lax monoidal are $\eta: J \xrightarrow{\epsilon} I \xrightarrow{\iota_{\text{id}_0}} \coprod_{\sigma} I \cong \underline{\text{hom}}_{\mathbf{M}}(e, e)$ and

$$\mu: (\coprod_{\sigma_1} I) * (\coprod_{\sigma_2} I) \cong \coprod_{\sigma_1, \sigma_2} I * I \xrightarrow{\coprod \nabla} \coprod_{\sigma_1, \sigma_2} I \xrightarrow{[\iota_{\sigma_1} \oplus \sigma_2]_{\sigma_1, \sigma_2}} \coprod_{\sigma} I.$$

The coherence diagrams commute by cocontinuity and the monoidal structure (I, ∇, ϵ) .

Proof (Proof of Proposition 4). The coherence morphisms are:

$$\begin{aligned}\eta_{S \hat{\circ} T}: J \xrightarrow{\Delta} J \circ J \xrightarrow{\eta_S \circ \eta_T} T(e, e) \circ S(e, e) &\rightarrow \int^b T(e, b) \circ S(b, e) \cong (S \hat{\circ} T)(e, e) \\ \mu_{S \hat{\circ} T}: (S \hat{\circ} T)(a, c) * (S \hat{\circ} T)(a', c') &\simeq \int^{b, b'} (T(a, b) \circ S(b, c)) * (T(a', b') \circ S(b', c')) \\ &\xrightarrow{\int \zeta} \int^{b, b'} (T(a, b) * T(a', b')) \circ (S(b, c) * S(b', c')) \\ &\xrightarrow{\int \mu_T \circ \mu_S} \int^{b, b'} T(a \oplus a', b \oplus b') \circ S(b \oplus b', c \oplus c') \\ &\rightarrow \int^b T(a \oplus a', b) \circ S(b, c' \oplus c') \simeq (S \hat{\circ} T)(a \oplus a', c \oplus c')\end{aligned}$$

Cocontinuity and Lemmas 3 and 4 finish the proof.

Lemma 5. *The \circ -composition of Proposition 4 is functorial.*

Proof. It is easy to see that $\hat{\circ}$ is well-defined on objects. Bifunctionality for morphisms then follows from bifunctionality of \circ and functoriality of coends.

Lemma 6. *The functor $\underline{\text{hom}}_M$ of Proposition 3 is the left and right identity of the \circ -composition of Proposition 4.*

Proof. The isomorphism on objects involves cocontinuity, the unitors of \circ , left Kan extending along the identity. Naturality is inherited from the naturality of the constructions involved. The unitors must also be monoidal natural transformations, which is true via cocontinuity and Lemma 1.

Lemma 7. *The \circ -composition of Proposition 4 is associative.*

Proof. The isomorphism uses cocontinuity and the associator of \circ . Naturality is inherited from the naturality of the constructions involved. The associator is a monoidal natural transformation by cocontinuity and Lemma 2.

C Proofs for change of enrichment

Proof (Proof of Theorem 4). Axioms i to iv hold by the axioms for lax monoidal functors for the same reason lax monoidal functors preserve monoids. Axioms v to viii each require the use of an axiom of double lax monoidal functors as shown below.

$$\begin{aligned} \text{idt}_F.\epsilon &= F\text{idt}.\eta_\circ.\epsilon \\ &= F\text{idt}.F\epsilon.\eta_* \\ &= F\text{zero}.\eta_* \\ &= \text{zero}_F \end{aligned}$$

$$\begin{aligned} \text{idt}_F.\nabla &= F\text{idt}.\eta_\circ.\nabla \\ &= F\text{idt}.F\nabla.\mu_*.(\eta_\circ * \eta_\circ) \\ &= F\text{par}.F(\text{idt} * \text{idt}).\mu_*.(\eta_\circ * \eta_\circ) \\ &= F\text{par}.\mu_*.(F\text{idt} * F\text{idt}).(\eta_\circ * \eta_\circ) \\ &= \text{par}_F.(\text{idt}_F * \text{idt}_F) \end{aligned}$$

$$\begin{aligned} \text{seq}_F.(\text{zero}_F \circ \text{zero}_F).\Delta &= F\text{seq}.\mu_\circ.(F\text{zero} \circ F\text{zero}).(\eta_* \circ \eta_*).\Delta \\ &= F\text{seq}.F(\text{zero} \circ \text{zero}).\mu_\circ.(\eta_* \circ \eta_*).\Delta \\ &= F\text{seq}.F(\text{zero} \circ \text{zero}).F\Delta.\eta_* \\ &= F\text{zero}.\eta_* \\ &= \text{zero}_F \end{aligned}$$

$$\begin{aligned} \text{seq}_F.(\text{par}_F \circ \text{par}_F).\zeta &= F\text{seq}.\mu_\circ.(F\text{par} \circ F\text{par}).(\mu_* \circ \mu_*).\zeta \\ &= F\text{seq}.F(\text{par} \circ \text{par}).\mu_\circ.(\mu_* \circ \mu_*).\zeta \\ &= F\text{seq}.F(\text{par} \circ \text{par}).F\zeta.\mu_\circ.(\mu_* \circ \mu_*) \\ &= F\text{par}.F(\text{seq} * \text{seq}).\mu_\circ.(\mu_* \circ \mu_*) \\ &= F\text{par}.\mu_\circ.(F\text{seq} * F\text{seq}).(\mu_* \circ \mu_*) \\ &= \text{par}_F.(\text{seq}_F * \text{seq}_F) \end{aligned}$$

Similar checks show that $\overline{F}(G)$ is a \mathbf{W} -Freyd map. \overline{F} is functorial by functoriality of F .

Canonicity of Proofs in Constructive Modal Logic \star

Matteo Acclavio¹, Davide Catta², and Federico Olimpieri³

¹ University of Southern Denmark, Odense, Denmark

² Università degli studi di Napoli, Federico II, Naples, Italy

³ University of Leeds, Leeds, UK

Abstract. In this paper we investigate the Curry-Howard correspondence for constructive modal logic in light of the gap between the proof equivalences enforced by the lambda calculi from the literature and by the recently defined winning strategies for this logic.

We define a new lambda-calculus for a minimal constructive modal logic by enriching the calculus from the literature with additional reduction rules and we prove normalization and confluence for our calculus. We then provide a typing system in the style of focused proof systems allowing us to provide a unique proof for each term in normal form, and we use this result to show a one-to-one correspondence between terms in normal form and winning innocent strategies.

Keywords: Constructive Modal Logic, Lambda Calculus, Game Semantics

1 Introduction

Proof theory is the branch of mathematical logic whose aim is studying the properties of logical arguments (i.e., proofs) as well as the structure of proofs and their invariants. For this purpose, the most used representations of proofs are based on tree-like data structures inductively defined using inference rules of a proof system.⁴ *Natural deduction* and *sequent calculus* are among the most used proof systems due to their intuitive representation. Both these proof systems were originally devised by Gentzen in order to prove the consistency of first-order arithmetic. Their versatility resulted in their employment for a wide variety of logics.

However, having formalisms able to represent proofs is not enough to define “what is a proof” since different derivations, or derivations in different proof systems, could represent the same abstract object. A notion of *proof identity* is therefore required to define a proof as a proper mathematical entity [18]. Such a notion of identity is provided by delineating the conditions under which two distinct formal representations of a proof represent the same logical argument. The definition of these conditions are often driven

[∗] The first author is supported by Villum Fonden, grant no. 50079. The second author is supported by the PRIN project RIPER (No. 20203FFYLK). The third author is supported by the US Air Force Office for Scientific Research under award number FA9550-21-1-0007.

⁴ It is worth noting that some proof systems (in the sense of [12]) allows to represent proofs using structures such as infinite trees (for non-well-founded proof systems, see, e.g., [15]), graphs (see proof nets [22,23], combinatorial proofs [27,27] or proof diagrams [3]) or structures defined in a compositional way (see open deduction [24] and deep inference [50])

by semantic considerations (by performing specific transformations on two derivations, they can be transformed to the same object) or intuitive ones (two derivations only differ for the order in which the same rules are applied to the same formulas).

Natural deduction is often considered a satisfactory formalism since it allows to define a more canonical representation of proofs with respect to sequent calculus: sequent calculus derivations differing because of some rules permutations are represented (*via* a standard translation) by the same natural deduction derivation. Moreover, natural deduction provides a one-to-one correspondence between derivations and lambda-terms, called the *Curry-Howard correspondence* [48].

Constructive Modal Logic. Classical modal logics are obtained by extending *classical logic* with unary operators, called *modalities*, that qualify the truth of a judgment. The most used modalities are the \Box (called *box*) and its dual operator \Diamond (called *diamond*) which are usually interpreted as *necessity* and *possibility*. According to the interpretation of such modalities, modal logics find applications, for example, in knowledge representation [51], artificial intelligence [40] and the formal verification of computer programs [45,19,36]. The work of Fitch [21] initiated the investigation of the proof theory of modal logics extending intuitionistic logic, leading to numerous results on the topic [46,26,39,20,35].

In particular, the Curry-Howard correspondence has been extended to various constructive modal logics [6,9,31,32,44,16]. Intuitionistic logic can be extended with modalities in different ways (for an overview see [47]): while in classical logic axioms involving only \Box provide also description of the behavior of \Diamond , for intuitionistic logic this is no more the case since the duality of the two modalities does not hold anymore. This leads to different approaches. *Constructive modal logics* consider minimal sets of axioms to guarantee the definition of the behaviors of the \Box and \Diamond modalities. A second approach, referred to as *intuitionistic modal logic*, considers additional axioms in order to validate the Gödel-Gentzen translation [14]. In this work we consider a minimal fragment of the constructive modal logic CK only containing the implication \rightarrow and the modality \Box . This fragment is enough to define types for a λ -calculus with a Let constructor [6] which can be interpreted as an explicit substitution and, for this reason, we more concisely denote by $N[M_1, \dots, M_n/x_1, \dots, x_n]$ instead of Let M_1, \dots, M_n be x_1, \dots, x_n in N .

Recent works on the the proof equivalence of constructive modal logics [5] expose a complexity gap between the proof equivalences induced by the natural deduction ([9]) and winning innocent strategies ([4]) for this logic. This discrepancy cannot be observed in intuitionistic propositional logic where there are one-to-one correspondences between natural deduction derivations, lambda terms and innocent winning strategies. In particular, in the logic CK we observe sequent calculus proofs which correspond to the same winning strategy but which cannot be represented by the same natural deduction derivation in the systems provided in [9,31] (or equivalently corresponding to different modal λ -terms). By means of example, consider the terms $x[z/x]$ and $x[z, w/x, y]$ and their (unique) typing derivations shown in Figure 1 (see Figure 3 for the typing system). Intuitively, the two terms $x[z/x]$ and $x[z, w/x, y]$ should be semantically equivalent since the explicit substitution of the variable y in the term x is vacuous. Said differently, if we explicit the substitution encoded by the constructor Let, both terms $x[z/x]$ and $x[z, w/x, y]$ should reduce to the term z .

$$\begin{array}{c}
 \frac{\text{Id} \quad \text{Id}}{z : \square a, w : \square b \vdash z : \square a} \quad \frac{\text{Id}}{x : a, y : b \vdash x : a} \\
 \frac{}{\square\text{-subst} \quad \frac{}{z : \square a, w : \square b \vdash z : \square a} \quad \frac{}{z : \square a, w : \square b \vdash w : \square b} \quad \frac{}{x : a, y : b \vdash x : a}}{z : \square a, w : \square b \vdash x[z/x]_\blacksquare : \square a} \\
 \\
 \frac{\text{Id} \quad \text{Id} \quad \text{Id}}{z : \square a, w : \square b \vdash z : \square a} \quad \frac{\text{Id}}{z : \square a, w : \square b \vdash w : \square b} \quad \frac{\text{Id}}{x : a, y : b \vdash x : a} \\
 \frac{}{\square\text{-subst} \quad \frac{}{z : \square a, w : \square b \vdash z : \square a} \quad \frac{}{z : \square a, w : \square b \vdash w : \square b} \quad \frac{}{x : a, y : b \vdash x : a}}{z : \square a, w : \square b \vdash x[z, w/x, y]_\blacksquare : \square a}
 \end{array}$$

Fig. 1. The typing derivations of the modal λ -terms $x[z/x]_\blacksquare$ and $x[z, w/x, y]_\blacksquare$.

In fact, this undesirable behavior disappear when considering the Winning Innocent Strategies for CK defined in [4]. In this syntax, both the above natural deduction derivations correspond to the same strategy below.

$$S = \{\epsilon, a^\circ, a^\circ a^\bullet\} \quad \text{over the arena} \quad \llbracket \square a, \square b \vdash \square a \rrbracket = \quad \text{Diagram} \quad (1)$$

Contribution. In this paper we define a new modal λ -calculus for CK by considering additional rewriting rules that allow us to retrieve a one-to-one correspondence between terms in normal form and winning innocent strategies, that is, providing more canonical representatives for proofs with respect to natural deduction and modal λ -terms defined in the literature. From the technical point-of-view, we obtain this result by extending the operational semantics of the modal λ -calculus with the appropriate new reduction rules for the explicit substitution encoded by the Let, dealing with contraction and weakening operating on the variables bound by the Let. We call this set of rules the κ -reduction, which we show to be strongly normalizing using elementary combinatorial methods. In order to deal with the interaction of the η -reduction with β -reduction, we define a restricted η -reduction following an approach similar to the one used in [42,17,30]. We prove strong normalization and confluence for our new operational semantics.

After proving confluence and strong normalization for our modal λ -calculus, we provide a canonical typing system inspired by focused sequent calculi (see, e.g., [7]) providing a unique typing derivation for each term in normal form. We conclude by establishing a one-to-one correspondence between the winning strategies defined in [4] and proofs of this calculi, therefore with terms in normal form.

Related Work. To the best of our knowledge, the first paper proposing a Curry-Howard correspondence for the logic CK is [9]. In this work, the authors provide a natural deduction system for the logic CK by enriching the standard system for intuitionistic propositional logic with a generalized elimination rule capable of taking into account the behavior of the \square -modality. At the level of lambda calculus, they enrich the syntax of terms by adding a new constructor Let defined as follows:

$$\text{Let } x_1, \dots, x_n \text{ be } N_1, \dots, N_n \text{ in } M \quad (\text{which we denote } M[N_1, \dots, N_n/x_1, \dots, x_n]_\blacksquare) \quad (2)$$

providing a notation which can be interpreted as an explicit substitution of the variable x_i with the term N_i for all occurrences of x_1, \dots, x_n inside a term M . For this calculus,

the authors only consider the usual η and β reductions plus the following reduction:

$$\text{Let } y \text{ be } P \text{ in } (\text{Let } x \text{ be } N \text{ in } M) \rightsquigarrow \text{Let } x \text{ be } (\text{Let } y \text{ be } P \text{ in } N) \text{ in } (\text{Let } x \text{ be } x \text{ in } M) \\ (\text{in our syntax this reduction is written as } M[N/x]_\blacksquare[P/y]_\blacksquare \rightsquigarrow M[x/x]_\blacksquare[N[P/y]_\blacksquare/x]_\blacksquare)$$

In [31] the author considers the usual η and β reduction with an the following additional β -reduction rule specifically designed to handle the explicit substitution construct.

$$M[\vec{P}, R[\vec{N}/\vec{z}]_\blacksquare, \vec{Q}/\vec{x}, \textcolor{red}{y}, \vec{w}]_\blacksquare \rightsquigarrow_{\beta_2} M\{R/y\}[\vec{P}, \vec{N}, \vec{Q}/\vec{x}, \vec{z}, \vec{w}]_\blacksquare \quad (3)$$

In the same paper, the author provides a detailed proof of strong normalization and confluence for modal lambda terms with respect to the standard η and β reduction, plus this new β_2 reduction. However, also this calculus does not manage to fix the aforementioned problem with canonicity.

An alternative natural deduction system (and λ -calculus) is proposed in [32], where the symmetry between elimination and introduction rules typical of natural deduction is restored. However, this result requires to define a sequent calculus where sequents have a more complex structure (dual-contexts), and lacks an in-depth study of the operational semantics because the η -expansion is not considered in the calculus.

Outline of the paper. In Section 2 we recall the definition of the fragment of the logic CK we consider in this paper, as well as the main results on the proof theory for this logic, its natural deduction and lambda calculus. In Section 3 we define the modal λ -calculus we consider in this paper, proving its strong normalization and confluence properties. In Section 4 we provide a typing system in the style of focused sequent calculi, where we are able to narrow the proof search of the type assignment of our normal terms to a single derivation. In Section 5 we recall the definition of the game semantics for the logic we consider and we prove the one-to-one correspondence between terms in normal form and winning strategies.

2 Preliminaries

In this section we recall the definition of the (fragment of the) constructive modal logic CK we consider in this paper, and we recall the definition and some terminology for modal λ -terms. We are interested in a minimal constructive modal logic whose *formulas* are defined from a countable set of propositional variables $\mathcal{A} = \{a, b, c, \dots\}$ using the following grammar:

$$A := a \mid (A \rightarrow A) \mid \square A \quad (4)$$

We say that a formula is *modality-free* if it contains no occurrences of the modality \square . A formula is a \rightarrow -formula if it is of the form $A \rightarrow B$. In the following we use Krivine's convention [37] and write $(A_1, \dots, A_n) \rightarrow C$ as a shortcut for $(A_1 \rightarrow (\dots \rightarrow (A_n \rightarrow C) \dots))$. A *sequent* is an expression $\Gamma \vdash C$ where Γ is a finite (possibly empty) list of formulas and C is a formula. If $\Gamma = A_1, \dots, A_n$ and σ a permutation over $\{1, \dots, n\}$, then we may write $\sigma(\Gamma)$ to denote $A_{\sigma(1)}, \dots, A_{\sigma(n)}$.

In this paper we consider the logic CK defined by extending the conjunction-free and disjunction-free fragment of intuitionistic propositional logic with the modality \square

$$\begin{array}{c}
\text{ax} \frac{}{a \vdash a} \quad \text{ex} \frac{\Gamma \vdash C}{\sigma(\Gamma) \vdash C} \quad \rightarrow^R \frac{\Gamma, A \vdash C}{\Gamma \vdash A \rightarrow C} \quad \rightarrow^L \frac{\Gamma \vdash A \quad B, A \vdash C}{\Gamma, A, A \rightarrow B \vdash C} \\
\kappa^\square \frac{\Gamma \vdash A}{\square \Gamma \vdash \square A} \quad \text{w} \frac{\Gamma \vdash C}{\Gamma, A \vdash C} \quad \text{c} \frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C} \quad \text{cut} \frac{\Gamma \vdash A \quad A, A \vdash C}{\Gamma, A \vdash C}
\end{array}$$

Fig. 2. Sequent calculus rules of the sequent system SCK, where σ is a permutation over $\{1, \dots, n\}$

$$\begin{array}{c}
\text{Id} \frac{}{x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i}^{i \in \{1, \dots, n\}} \quad \text{Abs} \frac{\Gamma, x : A \vdash M : C}{\Gamma \vdash \lambda x. M : A \rightarrow C} \quad \text{App} \frac{\Gamma \vdash N : A \quad \Gamma \vdash M : A \rightarrow C}{\Gamma \vdash MN : C} \\
\text{□-subst} \frac{\Gamma \vdash N_1 : \square A_1 \quad \dots \quad \Gamma \vdash N_n : \square A_n \quad x_1 : A_1, \dots, x_n : A_n \vdash M : C}{\Gamma \vdash M [N_1, \dots, N_n / x_1, \dots, x_n]_\blacksquare : \square C}^{x_1, \dots, x_n \text{ do not occur in } \Gamma}
\end{array}$$

Fig. 3. Typing rules in the natural deduction system ND_{CK} for modal λ -terms.

whose behavior is defined by the *necessitation rule* and the axiom K₁ below.

$$\text{Nec} := \text{if } A \text{ is provable, then also } \square A \text{ is} \quad K_1 := \square(A \rightarrow B) \rightarrow (\square A \rightarrow \square B)$$

The sequent calculus SCK, whose rules are provided in Figure 2, is a sound and complete proof system for the logic CK. This system have been extracted from the one presented in [38] and satisfies cut-elimination.

2.1 A Lambda Calculus for CK

The set of (untyped) *modal λ -terms* is defined inductively from a countable set of *variables* $\mathcal{V} = \{x, y, \dots\}$ using the following grammar:

$$M, N := x \mid \lambda x. M \mid (MN) \mid M [\vec{N} / \vec{x}]_\blacksquare \text{ where } \begin{cases} \vec{N} = N_1, \dots, N_n \text{ is a list of terms and} \\ \vec{x} = x_1, \dots, x_n \text{ is a list of distinct variables.} \end{cases}$$

modulo the standard α -equivalence (denoted $=_\alpha$, see [8]) and modulo the equivalence generated by the following permutations (for any σ permutation over the set $\{1, \dots, n\}$) over the order of substitutions in the $[\cdot / \cdot]_\blacksquare$ constructor:

$$[\vec{N} / \vec{x}]_\blacksquare := [N_1, \dots, N_n / x_1, \dots, x_n]_\blacksquare = [N_{\sigma(1)}, \dots, N_{\sigma(n)} / x_{\sigma(1)}, \dots, x_{\sigma(n)}]_\blacksquare =: [\sigma(\vec{N}) / \sigma(\vec{x})]_\blacksquare \text{ for any } \sigma \text{ permutation over } \{1, \dots, n\}.$$

As usual, application associates to the left, and has higher precedence than abstraction. For example, $\lambda xyz. xyz := \lambda x. (\lambda y. (\lambda z. ((xy)z)))$. A modal λ -term is a (*explicit*) *substitution* if it is of the form $M [\vec{N} / \vec{x}]_\blacksquare$, an *application* if of the form MN , and a λ -*abstraction* if of the form $\lambda x. M$.

The set of *subterms* of a term M (denoted SUB(M)) is defined as follows:

$$\begin{aligned} \text{Sub}(x) &= \{x\}, & \text{Sub}(\lambda x. M) &= \text{Sub}(M) \cup \{\lambda x. M\}, & \text{Sub}(MN) &= \text{Sub}(M) \cup \text{Sub}(N) \cup \{MN\}, \\ \text{Sub}(M [N_1, \dots, N_n / x_1, \dots, x_n]_\blacksquare) &= \text{Sub}(M) \cup \left(\bigcup_{i \in \{1, \dots, n\}} \text{Sub}(N_i) \right) \cup \{M [N_1, \dots, N_n / x_1, \dots, x_n]_\blacksquare\}. \end{aligned}$$

Its *length* $|M|$ and its set of *free variables* $\text{FV}(M)$ are defined as:

$$|M| = \begin{cases} 0 & \text{if } M = x \\ |N| + 1 & \text{if } M = \lambda x.N \\ \max\{|N|, |P|\} + 1 & \text{if } M = NP \\ \max\{|N|, |P_1|, \dots, |P_n|\} + 1 & \text{if } M = N[\vec{P}/\vec{x}] \end{cases}$$

$$\text{FV}(M) = \begin{cases} \{x\} & \text{if } M = x \\ \text{FV}(N) \setminus \{x\} & \text{if } M = \lambda x.N \\ \text{FV}(N) \cup \text{FV}(P) & \text{if } M = NP \\ \bigcup_i \text{FV}(P_i) & \text{if } M = N[\vec{P}/\vec{x}] \end{cases}$$

We denote $|M|_x$ the number of the occurrences of the free variable x in a term M and we may write $|M|_x = 0$ if $x \notin \text{FV}(M)$ and we say that a term M is *linear* in the variables x_1, \dots, x_n if $|M|_{x_i} = 1$ for all $i \in \{1, \dots, n\}$. We denote by $M[N_1, \dots, N_n/x_1, \dots, x_n]$ the result of the standard capture avoiding substitution of the occurrences of the variable x_1, \dots, x_n in M with the term N_1, \dots, N_n respectively (see, e.g., [49]).

A *variable declaration* is an expression $x : A$ where x is a variable and A is a *type*, that is, a formula as defined in Equation (4). A (*typing*) *context* is a finite list $\Gamma := x_1 : A_1, \dots, x_n : A_n$ of distinct variable declarations. Given a context $\Gamma = x_1 : A_1, \dots, x_n : A_n$, we say that a variable x *appears* in Γ if $x = x_i$ for a $i \in \{1, \dots, n\}$ and we denote by $\Gamma, y : B$ the context $x_1 : A_1, \dots, x_n : A_n, y : B$ implicitly assuming that y does not appear in Γ . A *type assignment* is an expression of the form $\Gamma \vdash M : A$ where Γ is a context, M a modal λ -term and A a type.

Definition 1. Let $\Gamma \vdash M : A$ be an type assignment. A typing derivation (or derivation for short) of $\Gamma \vdash M : A$ in ND_{CK} is a finite tree of type assignment constructed using the rules in Figure 3 in such a way it has root $\Gamma \vdash M : A$ and each leaf is the conclusion of a Id -rule. A type assignment is derivable (in ND_{CK}) if there is a derivation with conclusion the given type assignment.

We denote by Λ (resp. by Λ^\square and Λ^λ) the set of modal λ -terms (resp. the set of substitutions and λ -abstractions in Λ) admitting a derivable type assignment in ND_{CK} .

3 A New Modal Lambda Calculus

In this section we define a new modal lambda calculus by enriching the operational semantics of the previous calculi with additional reduction rules aiming at recovering canonicity, proving confluence and strong normalization properties.

To define our term rewriting rules, we require special care when they are applied in a proper sub-term. This is due to the fact that the explicit substitution encoded by $[\cdot/\cdot]_\blacksquare$ could capture free variables. For this reason, we introduce the notion of *term with a hole* as a term of the form $\mathbf{C}[\circ]$ containing a single occurrence of a special variable \circ . More precisely, the set CwH of terms with a hole and the two sets CwH_{η_1} and CwH_{η_2} of specific terms with a hole are defined by the following grammars:

$$\begin{aligned} \text{CwH} : \mathbf{C}[\circ] &:= \circ \mid \lambda x. \mathbf{C}[\circ] \mid M \mathbf{C}[\circ] \mid \mathbf{C}[\circ] M \mid \mathbf{C}[\circ] [\vec{M}/\vec{x}] \mid M[\vec{N}_1, \mathbf{C}[\circ], \vec{N}_2/\vec{x}_1, x, \vec{x}_2]_\blacksquare \\ \text{CwH}_{\eta_1} : \mathbf{E}[\circ] &:= \circ \mid \lambda x. \mathbf{E}[\circ] \mid M \mathbf{E}[\circ] \mid \mathbf{E}'[\circ] M \mid \mathbf{E}[\circ] [\vec{M}/\vec{x}]_\blacksquare \mid M[\vec{N}_1, \mathbf{E}, \vec{N}_2/\vec{x}_1, x, \vec{x}_2]_\blacksquare \\ \text{CwH}_{\eta_2} : \mathbf{D}[\circ] &:= \circ \mid \lambda x. \mathbf{D}[\circ] \mid M \mathbf{D}[\circ] \mid \mathbf{D}[\circ] M \mid \mathbf{D}[\circ] [\vec{M}/\vec{x}]_\blacksquare \mid M[\vec{N}_1, \mathbf{D}'[\circ], \vec{N}_2/\vec{x}_1, x, \vec{x}_2]_\blacksquare \\ &\quad \text{with } \mathbf{E}'[\circ] \neq [\circ] \neq \mathbf{D}'[\circ] \end{aligned}$$

We denote by $\mathbf{C}[M]$ the term obtained by replacing the hole \circ in $\mathbf{C}[\circ]$ with the term M . By means of example, if $\mathbf{C}[\circ] = \circ$ then $\mathbf{C}[M] = M$ and if $\mathbf{E}[\circ] = (\lambda x. xN)[\circ/x]_\blacksquare$

Ground Steps:

$$\begin{aligned}
& (\lambda x.M)N \rightsquigarrow_{\beta_1} M[N/x] \\
M[\vec{P}, R[\vec{N}/\vec{z}], \vec{Q}/\vec{x}, \vec{y}, \vec{w}] & \rightsquigarrow_{\beta_2} M[R/y][\vec{P}, \vec{N}, \vec{Q}/\vec{x}, \vec{z}, \vec{w}] \\
& M \rightsquigarrow_{\eta_1} \lambda x.Mx \quad \text{if } \Gamma \vdash M : A \rightarrow B, x \notin \text{FV}(M) \text{ and } M \notin \Lambda^\lambda \\
& M \rightsquigarrow_{\eta_2} x[M/x] \quad \text{if } \Gamma \vdash M : \Box A, x \notin \text{FV}(M) \text{ and } M \notin \Lambda^\Box \\
M[\vec{P}, N, \vec{Q}/\vec{x}, \vec{y}, \vec{z}] & \rightsquigarrow_{\kappa_1} M[\vec{P}, \vec{Q}/\vec{x}, \vec{z}] \quad \text{if } |M|_y = 0 \\
M[\vec{P}, N, \vec{N}, \vec{Q}/\vec{x}, \vec{y}_1, \vec{y}_2] & \rightsquigarrow_{\kappa_2} M[v, v/y_1, y_2][\vec{P}, N, \vec{Q}/\vec{x}, v, \vec{z}] \quad \text{with } v \text{ fresh}
\end{aligned}$$

Reduction Steps in Contexts:

$$\frac{\begin{array}{c} M \rightsquigarrow_{\beta_i} N \\ \mathbf{C}[M] \rightsquigarrow_{\beta_i} \mathbf{C}[N] \end{array}}{\mathbf{C}[M] \rightsquigarrow_{\beta_i} \mathbf{C}[N]}_{i \in \{1,2\}} \quad \frac{\begin{array}{c} M \rightsquigarrow_{\kappa_i} N \\ \mathbf{C}[M] \rightsquigarrow_{\kappa_i} \mathbf{C}[N] \end{array}}{\mathbf{C}[M] \rightsquigarrow_{\kappa_i} \mathbf{C}[N]}_{i \in \{1,2\}} \quad \text{and} \quad \frac{M \rightsquigarrow_{\eta_1} N}{\mathbf{E}[M] \rightsquigarrow_{\eta} \mathbf{E}[N]} \quad \text{and} \quad \frac{M \rightsquigarrow_{\eta_2} N}{\mathbf{D}[M] \rightsquigarrow_{\eta} \mathbf{D}[N]}$$

with $\mathbf{C}[\circ] \in \mathbf{CwH}$ and $\mathbf{E}[\circ] \in \mathbf{CwH}_{\eta_1}$ and $\mathbf{D}[\circ] \in \mathbf{CwH}_{\eta_2}$

Fig. 4. Definition of the ground steps of the reduction relations, and the rules for their extension to terms with holes.

then $\mathbf{E}[M] = (\lambda x.xN)[M/x]$. The reduction relations of our calculus are provided in Figure 4, where the ground steps and the rules for extending them to specific contexts are provided.

Remark 1. The term constructor **Let** (i.e., $[\cdot/\cdot]$) from Equation (2)) plays no role in the standard η and β reduction rules from the literature, where it behaves as a black-box during reduction. The inertness of this constructor with respect to normalization is indeed what makes the lambda calculus in [9,31] unable to identify terms whose expected behavior is the same as, for example, the following pairs of terms:

$$x[v/x] \quad \text{and} \quad x[v, w/x, y] \quad \mid \quad xyz[v, v/y, z] \quad \text{and} \quad xyy[v/y] \quad (5)$$

Our operational semantics extends the one provided in [31]. The novelty of our approach is the definition of the κ -reduction and the restriction of the η -reduction. The former is needed to being able to identify modal λ -terms with the same expected computational meaning, as the ones in Equation (5). The latter is carefully defined to avoid η -redexes that would make the reduction non-terminating, using a well-known technique in term rewriting theory (see, e.g., [42,30]).

The need of these restrictions can be observed in the two following (unrestricted) η -reduction chains, which are both forbidden by our restricted rule from Figure 4.

$$M \rightsquigarrow_{\eta} \lambda x.Mx \rightsquigarrow_{\eta} \lambda x.(\lambda y.My)x \rightsquigarrow_{\eta} \dots \quad \text{and} \quad M \rightsquigarrow_{\eta} x[M/x] \rightsquigarrow_{\eta} x[y[M/y]/x] \rightsquigarrow_{\eta} \dots$$

whenever $\Gamma \vdash M : A \rightarrow B$ whenever $\Gamma \vdash M : \Box A$

Moreover, our definition rules out interactions between the η and β reductions which could lead to infinite chains, as the ones shown below.

$$\begin{aligned}
\lambda x.M & \rightsquigarrow_{\eta} \lambda y.(\lambda x.M)y \rightsquigarrow_{\beta} \lambda y.(M\{x/y\}) =_{\alpha} \lambda x.M \quad \text{or} \\
x[M/x] & \rightsquigarrow_{\eta} x[y[M/y]/x] \rightsquigarrow_{\beta} y[M/y] =_{\alpha} x[M/x].
\end{aligned}$$

Definition 2. We define the following reduction relations:

$$\rightsquigarrow_{\beta\eta} = \rightsquigarrow_{\beta} \cup \rightsquigarrow_{\eta} \quad \rightsquigarrow_{\beta\kappa} = \rightsquigarrow_{\beta} \cup \rightsquigarrow_{\kappa} \quad \rightsquigarrow_{\beta\eta\kappa} = \rightsquigarrow_{\beta} \cup \rightsquigarrow_{\eta} \cup \rightsquigarrow_{\kappa} \quad (6)$$

For any $\xi \in \{\beta, \eta, \kappa, \beta\eta, \beta\kappa, \beta\eta\kappa\}$, we denote by \rightsquigarrow_ξ^+ its transitive closure, by $\rightsquigarrow_\xi^\equiv$ its reflexive closure, by \rightsquigarrow_ξ^* its reflexive and transitive closure, and by \equiv_ξ the equivalence relation it enforces over terms, that is, its reflexive, symmetric and transitive closure. Given a term M , we denote by $\text{nf}_\xi(M)$ the set of its \rightsquigarrow_ξ -normal form. A term M is strongly normalizable for \rightsquigarrow_ξ if it admits no infinite \rightsquigarrow_ξ -chains. A reduction \rightsquigarrow_ξ is strongly normalizing if every term M is strongly normalizable for it. A reduction \rightsquigarrow_ξ is confluent if given $M \rightsquigarrow_\xi^* N_1$ and $M \rightsquigarrow_\xi^* N_2$ there exists a term N such that $N_1 \rightsquigarrow_\xi^* N$ and $N_2 \rightsquigarrow_\xi^* N$.

The substitution lemma and subject reduction theorem holds for the reduction $\rightsquigarrow_{\beta\eta\kappa}$.

Lemma 1. [Substitution Lemma] Let $\Gamma, x : B \vdash M : C$ and $\Gamma \vdash N : B$ be derivable type assignments. Then $\Gamma, x : B \vdash M \{N/x\} : C$ is a derivable type assignment.

Proof. The proof is by induction on $|M|$.

- If $|M| = 1$ is a variable z we either have that $M = x$ or $M = y$ for some $y \neq x$ that appears in Γ . Then either $M \{N/x\} = N$ and $C = B$, or $M \{N/x\} = M$. In both cases $\Gamma \vdash M \{N/x\} : C$ is derivable by hypothesis.
- If $|M| \geq 1$ and M is an abstraction or an application, then the proof is the same as in standard λ -calculus [49]. If $M = P[T_1, \dots, T_n/x_1, \dots, x_n]$, then $C = \square C'$ and $M \{N/x\} = P[T_1 \{N/x\}, \dots, T_n \{N/x\}/x_1, \dots, x_n]$. Then, by definition of derivability, we have $x \notin \{x_1, \dots, x_n\}$ and the type assignments $x_1 : A_1, \dots, x_n : A_n \vdash M : C$ and $\Gamma, x : B \vdash N_i : \square A_i$ are derivable for all $i \in \{1, \dots, n\}$ and for some A_1, \dots, A_n . We can apply inductive hypothesis on $\Gamma, x : B \vdash N_i : \square A_i$ to deduce that the type assignment $\Gamma \vdash T_i \{N/x\} : \square A_i$ is derivable for all $i \in \{1, \dots, n\}$. We conclude the existence the desired type assignment with bottom-mots rule a \square -subst with premises $x_1 : A_1, \dots, x_n : A_n \vdash \{M/x\} : C$ and $\Gamma \vdash T_i \{N/x\} : \square A_i$ for all $i \in \{1, \dots, n\}$.

Theorem 1. Let $\Gamma \vdash M : C$ be derivable. If $M \rightsquigarrow_{\beta\eta\kappa} N$, then $\Gamma \vdash N : C$.

Proof. Because of Lemma 1, it suffices to check the cases when M reduces to N in one ground step of $\rightsquigarrow_{\beta\eta\kappa}$:

- if $M \rightsquigarrow_{\beta_1} N$, then $M = (\lambda x.P)Q$ and $N = P\{Q/x\}$. The case where $M \rightsquigarrow_{\beta_2} N$ uses a similar argument. The result follows the fact that if $\Gamma, x : B \vdash M : C$ and $\Gamma \vdash N : B$ are derivable type assignment, then $\Gamma, x : B \vdash M \{N/x\} : C$ by Lemma 1.
- if $M \rightsquigarrow_{\eta_1} N$, then $C = A \rightarrow B$ and $N = \lambda x.Mx$. The result follows by applying the rule **Abs**. The case where $M \rightsquigarrow_{\eta_2} N$ uses a similar argument;
- if $M \rightsquigarrow_{\kappa_1} N_1$, then $M = M'[P_1, \dots, P_k, \textcolor{red}{N}, P_{k+1}, \dots, P_n/x_1, \dots, x_k, \textcolor{red}{x}, x_{k+1}, \dots, x_n]$ such that x is not free in M , $C = \square B$, and $N_1 = M'[\vec{P}, \vec{Q}/\vec{x}, \vec{y}]$. Then there are derivations for $\Gamma \vdash P_i : A_i$ for all $i \in \{1, \dots, n\}$ (for some A_i) and a derivation for $x_1 : A_1, \dots, x_k : A_k, x : A, x_{k+1} : A_{k+1}, \dots, x_n : A_n \vdash M' : B$. Therefore we have a derivation for $x_1 : A_1, \dots, x_n : A_n \vdash M' : B$ since weakening is admissible (that is, whenever $\Gamma, x : A \vdash M : C$ is derivable and x does not occur free in M , then

$\Gamma \vdash M : C$ is also derivable⁵. Then we have a derivation of $\Gamma \vdash N : C$ with bottom-most rule a $\square\text{-subst}$ with right-most premise $x_1 : A_1, \dots, x_n : A_n \vdash M' : B$. and a premise $\Gamma \vdash P_i : A_i$ for each $i \in \{1, \dots, n\}$;

- if $M \rightsquigarrow_{\kappa_2} N_1$, then we conclude similarly to the previous point since we have

$$M = M' [\vec{P}, \textcolor{red}{N}, \textcolor{red}{N}, \vec{Q}/\vec{x}, \textcolor{red}{y}_1, \textcolor{red}{y}_2, \vec{z}]_\blacksquare \quad \text{and} \quad N_1 = M \{y, y/y_1, y_2\} [\vec{P}, \textcolor{red}{N}, \vec{Q}/\vec{x}, \textcolor{red}{y}, \vec{z}]_\blacksquare.$$

We can prove local confluence of $\rightsquigarrow_{\beta\eta\kappa}$ by case analysis of the critical pairs using the following lemma.

Lemma 2. *Let P, P' and Q modal λ -terms. If $P \rightsquigarrow_{\beta\eta\kappa} P'$, then $P \{Q/x\} \rightsquigarrow_{\beta\eta\kappa}^* P' \{Q/x\}$. Moreover, there is a N_Q such that $Q \{P/x\} \rightsquigarrow_{\beta\eta\kappa}^* N_Q$ and $Q \{P'/x\} \rightsquigarrow_{\beta\eta\kappa}^* N_Q$.*

Proof. To prove that $P \{Q/x\} \rightsquigarrow_{\beta\eta\kappa}^* P' \{Q/x\}$ we first prove that the result holds for the ground reduction steps:

- If $P = (\lambda y.M)N \rightsquigarrow_{\beta_1} M \{N/y\} = P'$, then we have $P \{Q/x\} = ((\lambda y.M)N) \{Q/x\} = (\lambda y.M \{Q/x\})(N \{Q/x\})$. We conclude by associativity of substitution.
- If $P \rightsquigarrow_{\beta_1} P'$, we conclude similarly to the previous case.
- If $P \rightsquigarrow_{\eta_1} \lambda y.Py = P'$, then $(\lambda y.Py) \{Q/x\} = \lambda y.(P \{Q/x\})y$. We conclude since, definition of $\rightsquigarrow_{\eta_1}$, we have that $P \{Q/x\} \{q/x\} \rightsquigarrow_{\eta_1} \lambda y.(P \{Q/x\})y$.
- If $P \rightsquigarrow_{\eta_2} y[P/y]_\blacksquare = P'$, then $P' \{Q/x\} = y[(P \{Q/x\})/y]_\blacksquare$. We conclude since, by definition of $\rightsquigarrow_{\eta_2}$, we have that $P \{Q/x\} \rightsquigarrow_{\eta_2} y[(P \{Q/x\})/y]_\blacksquare$.
- If $P = M' [\vec{P}, N, \vec{Q}/\vec{x}, y, \vec{z}]_\blacksquare \rightsquigarrow_{\kappa_1} P' = M' [\vec{P}, \vec{Q}/\vec{x}, \vec{z}]_\blacksquare$, then $P \{Q/x\} = M' \{Q/x\} [\vec{P} \{Q/x\}, N \{Q/x\}, \vec{Q} \{Q/x\}/\vec{x}, y, \vec{z}]_\blacksquare$. We conclude since $P' \{Q/x\} = M' \{Q/x\} [\vec{P} \{Q/x\}, \vec{Q} \{Q/x\}/\vec{x}, \vec{z}]_\blacksquare$, thus $P \{Q/x\} \rightsquigarrow_{\kappa_1} P' \{Q/x\}$.
- If $P = M' [\vec{P}, N, N, \vec{Q}/\vec{x}, y_1, y_2, \vec{z}]_\blacksquare \rightsquigarrow_{\kappa_2} P' = M' \{v, v/y_1, y_2\} [\vec{P}, N, \vec{Q}/\vec{x}, v, \vec{z}]_\blacksquare$, then $P \{Q/x\} = M' \{Q/x\} [\vec{P} \{Q/x\}, N \{Q/x\}, N \{Q/x\}, \vec{Q} \{Q/x\}/\vec{x}, y_1, y_2, \vec{z}]_\blacksquare$ and $P' \{Q/x\} = (M' \{v, v/y_1, y_2\}) \{Q/x\} [\vec{P} \{Q/x\}, N \{Q/x\}, \vec{Q} \{Q/x\}/\vec{x}, v, \vec{z}]_\blacksquare$. We conclude since $(M' \{v, v/y_1, y_2\}) \{Q/x\} = (M' \{Q/x\}) \{v, v/y_1, y_2\}$, thus $P \{Q/x\} \rightsquigarrow_{\kappa_2} P' \{Q/x\}$.

Then we conclude by showing that it also holds when reductions are applied in a context.

- If $P = \lambda y.M$ for a M such that $M \rightsquigarrow_{\beta\eta\kappa} M'$, then $P' = \lambda y.M'$ and we conclude by inductive hypothesis since $P \{Q/x\} = \lambda y.M \{Q/x\} \rightsquigarrow_{\beta\eta\kappa} \lambda y.M' \{Q/x\}$.
- If $P = MN$, then $P \{Q/x\} = M \{Q/x\} N \{Q/x\}$. In this case, either $M \rightsquigarrow_{\beta\eta\kappa} M'$ and $P' = M'N$, or $N \rightsquigarrow_{\beta\eta\kappa} N'$ and $P' = MN'$. We conclude taking into account the restriction of the possible application of the reduction steps in a context. Note that without the restriction on $\rightsquigarrow_{\eta_1}$ we could have had $M \{Q/x\} \rightsquigarrow_{\eta_1} \lambda y.(M \{Q/x\})y$, and therefore $M \{Q/x\} N \rightsquigarrow_{\eta_1} \lambda y.(M \{Q/x\})yN \rightsquigarrow_{\beta_1} M \{Q/x\} N$;

⁵ The admissibility of weakening is easily proven by induction on the size of a derivation.

- If $P = M[N/y]$, then $P\{Q/x\} = M\{Q/x\}[N\{Q/x\}/y]$. In this case, either $M \rightsquigarrow_{\beta\eta\kappa} M'$ and $P' = M'[N/y]$, or $N \rightsquigarrow_{\beta\eta\kappa} N'$ and $P' = M[N'/y]$. We conclude taking into account the restriction of the possible application of the reduction steps in a context. Note that without the restriction on $\rightsquigarrow_{\eta_2}$ we could have had $N\{Q/x\} \rightsquigarrow_{\eta_2} y[N\{Q/x\}/y]$, and therefore the following sequence of reductions.

$$M[N\{Q/x\}/z] \rightsquigarrow_{\eta_2} M[y[N\{Q/x\}/y]/z] \rightsquigarrow_{\beta_2} M[N\{Q/x\}/z]$$

The fact that, for each Q , there is a N_Q such that $Q\{P/x\} \rightsquigarrow_{\beta\eta\kappa}^* N_Q$ and $Q\{P'/x\} \rightsquigarrow_{\beta\eta\kappa}^* N_Q$ is proven by induction on the structure of Q and considering the restrictions on the definition of the rewriting steps in a context.

Proposition 1. *The reduction $\rightsquigarrow_{\beta\eta\kappa}$ is locally confluent.*

Proof. We show that if there are M, N_1 and N_2 with $N_1 \neq N_2$ such that $M \rightsquigarrow_{\beta\eta\kappa} N_1$ and $M \rightsquigarrow_{\beta\eta\kappa} N_2$, then there exists N such that $N_1 \rightsquigarrow_{\beta\eta\kappa}^* N$ and $N_2 \rightsquigarrow_{\beta\eta\kappa}^* N$. Without loss of generality we have the following cases:

1. if $M \rightsquigarrow_{\beta_1} N_1$ with $M = (\lambda x.P)Q$ and $N_1 = P\{Q/x\}$, then N_2 can only be obtained by applying $\rightsquigarrow_{\beta\eta\kappa}$ the subterms P and Q of M . We conclude by Lemma 2;
2. if $M \rightsquigarrow_{\beta_2} N_1$ with $M = M'[\vec{P}, R[\vec{N}/\vec{z}], \vec{Q}/\vec{x}, y, \vec{w}]$ and with $N_1 = M'\{R/y\}[\vec{P}, \vec{N}, \vec{Q}/\vec{x}, \vec{z}, \vec{w}]$, then N_2 must be a term obtained by applying $\rightsquigarrow_{\beta\eta\kappa}$ on R or on one of the terms in \vec{P}, \vec{N} or \vec{Q} . We conclude again by Lemma 2;
3. if $M \rightsquigarrow_{\eta_1} N_1$, then $\Gamma \vdash M : A \rightarrow B$ and $N_1 = \lambda x.Mx$. Therefore, for any N_2 such that $M \rightsquigarrow_{\beta\eta\kappa} N_2$ we have that $\Gamma \vdash N_2 : A \rightarrow B$ (by subject reduction). Then
 - either N_2 is not an abstraction and we conclude by letting $N = \lambda x.N_2x$.
 - otherwise $N_2 = \lambda y.M'$ and we conclude since $N_1 \rightsquigarrow_{\eta_1} \lambda x.N_2x \rightsquigarrow_{\beta_1} N_2$.
4. if $M \rightsquigarrow_{\eta_2} N_1$ with $\Gamma \vdash M : \square A$ and $N_1 = x[M/x]$, then we conclude with a similar argument with respect to the previous point by letting $N = x[N_2/x]$.
5. if $M \rightsquigarrow_{\kappa} N_1$, $M = M'[\vec{P}, \textcolor{red}{N}, \vec{Q}/\vec{x}, \textcolor{red}{x}, \vec{y}] \rightsquigarrow_{\kappa_1} M'[\vec{P}, \vec{Q}/\vec{x}, \vec{y}] = N_1$, or $M = M'[\vec{P}, \textcolor{red}{N}, \textcolor{red}{N}, \vec{Q}/\vec{x}, \textcolor{red}{y}_1, \textcolor{red}{y}_2, \vec{z}] \rightsquigarrow_{\kappa_2} M\{y, y/y_1, y_2\}[\vec{P}, \textcolor{red}{N}, \vec{Q}/\vec{x}, \textcolor{red}{y}, \vec{z}] = N_1$. In both cases we conclude with an argument similar to the one in Case (2).

In order to prove the termination of $\rightsquigarrow_{\beta\eta\kappa}$, we define the following measures.

Definition 3. *Let M be a modal λ -term. We define the following multisets of derivable type assignments:*

$$\begin{aligned} \text{Est}_1(M) &= \left\{ B \rightarrow C \mid P \in \text{Sub}(M) \setminus \Lambda^\lambda \text{ such that } M \neq PQ \text{ and } \Gamma \vdash P : B \rightarrow C \right\} \\ \text{Est}_2(M) &= \left\{ \square B \mid P \in \text{Sub}(M) \setminus \Lambda^\square \text{ such that } M \neq Q[\vec{N}_1, P, \vec{N}_2/\vec{x}_1, x, \vec{x}_2] \text{ and } \Gamma \vdash P : \square B \right\} \end{aligned}$$

We then define $\|M\|_\eta := \|M\|_\eta^1 + \|M\|_\eta^2$ with

$$\|M\|_\eta^1 := \sum_{A \in \text{Est}_1(M)} \|A\|_\eta^1 \quad \text{and} \quad \|M\|_\eta^2 := \sum_{A \in \text{Est}_2(M)} \|A\|_\eta^2$$

$$\text{where} \quad \begin{array}{lll} \|a\|_\eta^1 = 0 & \|A \rightarrow B\|_\eta^1 = \|A\|_\eta^1 + \|B\|_\eta^1 + 1 & \|\square A\|_\eta^1 = \|A\|_\eta^1 \\ \|a\|_\eta^2 = 0 & \|A \rightarrow B\|_\eta^2 = \|A\|_\eta^2 + \|B\|_\eta^2 & \|\square A\|_\eta^2 = \|A\|_\eta^2 + 1 \end{array}$$

We also define $\|M\|_k$ as the size of substitution subterms of M as follows:

$$\begin{aligned}\|x\|_k &= 0 & \|\lambda x M\|_k &= \|M\|_k & \|MN\|_k &= \|M\|_k + \|N\|_k \\ \|M[N_1, \dots, N_n/x_1, \dots, x_n]_\blacksquare\|_k &= \|M\|_k + \|N\|_k + n\end{aligned}$$

Example 1. Intuitively, the measure $\|\cdot\|_\eta$ does not take into account all the subterms of M , but only the ones on which we can apply the restricted \rightsquigarrow_η . For an example, consider the modal λ -term $M = (\lambda z^{a \rightarrow a}.z)y$ with $\|M\|_\eta = 3$ because all four subterms of M are of type $a \rightarrow a$ -formula, but the subterm $\lambda z.z$ is an abstraction, therefore no \rightsquigarrow_η can be applied on it. If $M \rightsquigarrow_\eta N$, because of the restrictions on \rightsquigarrow_η , we have that

- either $N = (\lambda z.z)(\lambda v.yv)$ with $\|N\|_\eta = 2$ because no \rightsquigarrow_η can be applied to the subterms y and $\lambda z.z$ (they occur on the left of an application) or $\lambda v.yv$ (it is an abstraction), but only to the subterms z and the whole term N ;
- or $N = \lambda v^a.((\lambda z.z)y)v$ with $\|N\|_\eta = 2$ because \rightsquigarrow_η can only be applied to the subterms z and y .

Lemma 3. Let M and N be modal λ -terms. If $M \rightsquigarrow_\eta N$, either $\|N\|_\eta < \|M\|_\eta$ or there is N' such that $N \rightsquigarrow_\eta N'$ and $\|N'\|_\eta < \|M\|_\eta$.

Proof. We only discuss the two following cases, since the others are direct consequence of the definitions of $\|\cdot\|_\eta^1$ and $\|\cdot\|_\eta^2$.

1. If $\Gamma \vdash M : C$ with $C = A \rightarrow B$ and $M \rightsquigarrow_{\eta_1} N$, then $N = \lambda x.Mx$. Therefore, $\|C\|_\eta^1 = \|A \rightarrow B\|_\eta^1 > 0$ and

$$\|M\|_\eta^1 = \left(\sum_{C' \in \text{Est}_1(M)} \|C'\|_\eta^1 \right) > \sum_{C' \in \text{Est}_1(M) \setminus \{C\}} \|C'\|_\eta^1 > \|\lambda x.Mx\|_\eta^1 = \|N\|_\eta^1$$

Now consider $\|N\|_\eta^2$. We reason by cases on the structure of the type A . If A is not a box, we can conclude by setting $N' = N$. If A is of the shape $\Box A'$ then $\|N\|_\eta^2 = \sum_{C' \in \text{Est}_2(M) \cup \{A\}} \|C'\|_\eta^2$. This means that we can perform a step $\lambda x.Mx \rightsquigarrow_{\eta_2} \lambda x.(M(z[x/z]_\blacksquare))$ with z fresh. Now we set $N' = \lambda x.(M(z[x/z]_\blacksquare))$ since $\|N'\|_\eta^2 = \sum_{C' \in \text{Est}_2(M)} \|C'\|_\eta^2 = \|M\|_\eta^2$. We remark that $\|N'\|_\eta^1 = \|N\|_\eta^1$. Hence $\|N'\|_\eta = \|N\|_\eta^1 + \|M\|_\eta^2 < \|M\|_\eta^1 + \|M\|_\eta^2 = \|M\|_\eta$ since $\|N\|_\eta^1 < \|M\|_\eta^1$ for what we said before.

2. If $\Gamma \vdash M : C$ with $C = \Box A$ and $M \rightsquigarrow_{\eta_2} N$, then $N = x[M/x]_\blacksquare$. Therefore, $\|C\|_\eta^2 = \|\Box A\|_\eta^2 > 0$ and

$$\|M\|_\eta^2 = \left(\sum_{C' \in \text{Est}_1(M)} \|C'\|_\eta^2 \right) > \sum_{C' \in \text{Est}_1(M) \setminus \{C\}} \|C'\|_\eta^1 > \|\lambda x.Mx\|_\eta^2 = \|x[M/x]_\blacksquare\|_\eta^2 = \|N\|_\eta^2$$

Now consider $\|N\|_\eta^1$. We reason by cases on the structure of the type A . If A is not an implication, we can conclude by setting $N' = N$. If A is of the shape $A' \rightarrow B$ then $\|N\|_\eta^1 = \sum_{C' \in \text{Est}_1(M) \cup \{A\}} \|C'\|_\eta^2$. This means that we can perform a step $x[M/x]_\blacksquare \rightsquigarrow_{\eta_2} (\lambda y.xy)[M/x]_\blacksquare$ with y fresh. Now we set $N' = (\lambda y.xy)[M/x]_\blacksquare$ since $\|N'\|_\eta^1 = \sum_{C' \in \text{Est}_1(M)} \|C'\|_\eta^2 = \|M\|_\eta^1$. We remark that $\|N'\|_\eta^2 = \|N\|_\eta^2$. Hence $\|N'\|_\eta = \|N\|_\eta^2 + \|M\|_\eta^1 < \|M\|_\eta^2 + \|M\|_\eta^1 = \|M\|_\eta$ since $\|N\|_\eta^2 < \|M\|_\eta^2$ for what we said before.

Lemma 4. Let P, Q, N be modal λ -terms. If $P \{Q/x\} \rightsquigarrow_{\eta} N$, then there are N_1 and N_2 such that $P \rightsquigarrow_{\eta}^* N_1$ and $Q \rightsquigarrow_{\eta}^* N_2$ with $N_1 \{N_2/x\} = N$.

Proof. We prove it by induction on the structure of P .

- If $P = x$ then $P \{Q/x\} = Q \rightsquigarrow_{\eta} N$. Then $N_1 = x$ and $N_2 = N$.
- If $P = \lambda y.P'$, then $P \{Q/x\} = \lambda y.P' \{Q/x\}$. By definition, if $P \{Q/x\} \rightsquigarrow_{\eta} N$ then $\Gamma \vdash P : A \rightarrow B$ and then $N = \lambda z.P''$ with $P' \{Q/x\} \rightsquigarrow_{\eta} P''$. In this case we apply inductive hypothesis to get N'_1 and N'_2 such that $P' \rightsquigarrow_{\eta} N'_1$ and $Q \rightsquigarrow_{\eta} N'_2$ with $N'_1 \{N'_2/x\} = P''$. We conclude by letting $N_1 = \lambda y.N'_1$ and $N_2 = N'_2$;
- If $P = ST$, then $P \{Q/x\} = S \{Q/x\} T \{Q/x\}$. By definition of \rightsquigarrow_{η} , if $P \{Q/x\} \rightsquigarrow_{\eta} N$ then:
 - if $S \{Q/x\} T \{Q/x\} \rightsquigarrow_{\eta_1} \lambda y.(S \{Q/x\} T \{Q/x\})y = N$, then $N_1 = \lambda y(ST)y$ and $N_2 = Q$;
 - if $S \{Q/x\} T \{Q/x\} \rightsquigarrow_{\eta_2} y[S \{Q/x\} T \{Q/x\} / y]_{\blacksquare} = N$, then $N_1 = y[ST/y]_{\blacksquare}$ and $N_2 = Q$;
 - otherwise, the step $N \rightsquigarrow_{\eta} N'$ must be applied in a context. In this case, we have that $N = P_1 P_2$ and either $S \{Q/x\} \rightsquigarrow_{\eta} P_1$ or $T \{Q/x\} \rightsquigarrow_{\eta} P_2$. In the case $S \{Q/x\} \rightsquigarrow_{\eta} P_1$, by definition of the contextual step, $\{Q/x\} \rightsquigarrow_{\eta} P_1$ cannot be a step of $\rightsquigarrow_{\eta_1}$. By inductive hypothesis there are $N_{1,2}$ and $N_{2,2}$ such that $S \rightsquigarrow_{\eta} N_{1,2}$ and $Q \rightsquigarrow_{\eta} N_{2,2}$ with $P_1 = N_{1,1} \{N_{2,2}/x\}$. Then $N_1 = N_{1,1}(T \{Q/x\})$ and $N_2 = N_{2,2}$. The other case is similar.
- If $P = S \left[\vec{T} / \vec{y} \right]_{\blacksquare}$, then $P \{Q/x\} = S \{Q/x\} \left[\vec{T} \{Q/x\} / \vec{y} \right]_{\blacksquare}$. By definition of \rightsquigarrow_{η} , if $P \{Q/x\} \rightsquigarrow_{\eta} N$ then:
 - if $S \{Q/x\} \left[\vec{T} \{Q/x\} / \vec{y} \right]_{\blacksquare} \rightsquigarrow_{\eta_1} \lambda y.(S \{Q/x\} \left[\vec{T} \{Q/x\} / \vec{y} \right]_{\blacksquare})y = N$, then $N_1 = \lambda y.(S \left[\vec{T} / \vec{y} \right]_{\blacksquare})y$ and $N_2 = Q$;
 - otherwise, the step $N \rightsquigarrow_{\eta} N'$ must be applied in a context. In this case, we have that $N = P_1 \left[\vec{P}_2 / \vec{y} \right]_{\blacksquare}$ and either $S \{Q/x\} \rightsquigarrow_{\eta} P_1$ or $\vec{T} \{Q/x\} \rightsquigarrow_{\eta} \vec{P}_2$. We do the case $S \{Q/x\} \rightsquigarrow_{\eta} P_1$. By inductive hypothesis there exists $N_{1,2}$ and $N_{2,2}$ such that $S \rightsquigarrow_{\eta} N_{1,2}$ and $Q \rightsquigarrow_{\eta} N_{2,2}$ with $P_1 = N_{1,1} \{N_{2,2}/x\}$. Then $N_1 = N_{1,1} \left[\vec{T} \{Q/x\} / \vec{y} \right]_{\blacksquare}$ and $N_2 = N_{2,2}$. The other case is similar.

Lemma 5. The following commutations between \rightsquigarrow_{β} , \rightsquigarrow_{η} and $\rightsquigarrow_{\kappa}$ hold:

- if $M \rightsquigarrow_{\kappa} N \rightsquigarrow_{\beta} N'$, then there is M' such that $M \rightsquigarrow_{\beta} M'$ and $M' \rightsquigarrow_{\kappa}^* N'$;
- if $M \rightsquigarrow_{\eta} N \rightsquigarrow_{\kappa} N'$, then there is M' such that $M \rightsquigarrow_{\kappa} M'$ and $M' \rightsquigarrow_{\eta}^* N'$;
- if $M \rightsquigarrow_{\beta} N \rightsquigarrow_{\eta} N'$, then there is M' such that $M \rightsquigarrow_{\eta} M'$ and $M' \rightsquigarrow_{\beta}^* N'$.

Proof. After Lemmas 1 and 4, we can reason on the structure of M only considering its possible shape according to the ground steps of $\rightsquigarrow_{\beta\eta\kappa}$.

- If $M \rightsquigarrow_{\eta_1} N = \lambda x.Mx$, then, by lemma 4 we have N_1 and N_2 such that $P \rightsquigarrow_{\eta} N_1$ and $Q \rightsquigarrow_{\eta} N_2$ with $N_1 \{N_2/x\} = N$. We conclude by letting $M' = (\lambda x.N_1)N_2$. A similar argument is applied if $M \rightsquigarrow_{\eta_2} N$.
- If $M = (\lambda x.P)Q \rightsquigarrow_{\beta_1} P \{Q/x\} = N$, then we have N_1 and N_2 such that $P \rightsquigarrow_{\eta} N_1$ and $Q \rightsquigarrow_{\eta} N_2$ with $N_1 \{N_2/x\} = N$. We conclude by letting $M' = (\lambda x.N_1)N_2$. A similar argument is applied if $M \rightsquigarrow_{\beta_2} N$.

- If $M = M[\vec{P}, N, \vec{Q}/\vec{x}, y, \vec{y}] \rightsquigarrow_{\kappa_1} M[\vec{P}, \vec{Q}/\vec{x}, \vec{y}] = N$, then we can conclude since any β -redex of N is also a β -redex of M . A similar argument is applied if $M \rightsquigarrow_{\kappa_2} N$.

Therefore, the following corollary trivially holds.

Corollary 1. *The following hold.*

- if $M \rightsquigarrow_{\kappa} N$, then $\text{nf}_{\beta}(M) \rightsquigarrow_{\kappa}^* \text{nf}_{\beta}(N)$;
- if $M \rightsquigarrow_{\eta} N$, then $\text{nf}_{\kappa}(M) \rightsquigarrow_{\eta} \text{nf}_{\kappa}(N)$;
- if $M \rightsquigarrow_{\beta} N$, then $\text{nf}_{\eta}(M) \rightsquigarrow_{\beta}^* \text{nf}_{\eta}(N)$.

Theorem 2. *The reduction relation $\rightsquigarrow_{\beta\eta\kappa}$ is strongly normalizing and confluent.*

Proof. After Proposition 1, it suffices to prove that $\rightsquigarrow_{\beta\eta\kappa}$ is strongly normalizing to conclude by Newman's lemma that $\rightsquigarrow_{\beta\eta\kappa}$ is also confluent.

To prove strong normalization we use the fact that the reductions \rightsquigarrow_{β} , \rightsquigarrow_{η} and $\rightsquigarrow_{\kappa}$ are strongly normalizing: for \rightsquigarrow_{β} the proof can be found in [31], for \rightsquigarrow_{η} the proof is by induction on $\|\cdot\|_{\eta}$ using Lemma 3, and for $\rightsquigarrow_{\kappa}$ it follows the fact that, by definition of $\|\cdot\|_{\kappa}$, we have that $\|M\|_{\kappa} > \|N\|_{\kappa}$ whenever $M \rightsquigarrow_{\kappa} N$. To conclude that $\rightsquigarrow_{\beta\eta\kappa}$ also is strongly normalizing, the standard result (see, e.g., [49]) in rewriting theory ensuring that given two strongly normalizing reduction relations \rightsquigarrow_1 and \rightsquigarrow_2 with \rightsquigarrow_1 confluent, if $M \rightsquigarrow_2 N$ implies the existence of a reduction $\text{nf}_1(M) \rightsquigarrow_2^+ \text{nf}_1(N)$ for any M and N , then $\rightsquigarrow_1 \cup \rightsquigarrow_2$ is strongly normalizing. In our case, the fact that $M \rightsquigarrow_2 N$ implies $\text{nf}_1(M) \rightsquigarrow_2^+ \text{nf}_1(N)$ is a corollary of Lemma 5.

Definition 4. *The set $\widehat{\Lambda}$ is the set of modal λ -terms defined inductively as follows:*

- if x is a variable, $T_1, \dots, T_n \in \widehat{\Lambda}$, and there are derivations for the types assignments $\Gamma \vdash x : (A_1, \dots, A_n) \rightarrow C$ with C atomic and $\Gamma \vdash T_i : A_i$ for all $i \in \{1, \dots, n\}$, then $xT_1 \cdots T_n \in \widehat{\Lambda}$. Variables are the special case with $n = 0$;
- if $T \in \widehat{\Lambda}$ and there is a derivation of $\Gamma, x : A \vdash T : C$, then $\lambda x^A.T \in \widehat{\Lambda}$;
- if $M \in \widehat{\Lambda}$, $\text{FV}(M) = \{x_1, \dots, x_n\}$ and the type assignment $x_1 : B_1, \dots, x_n : B_n \vdash M : C$ is derivable, and if there are n distinct terms $T_1, \dots, T_n \in \Lambda$ of the shape $T_i = y_i U_{i1} \cdots U_{ik_i}$ with $U_{ij} \in \widehat{\Lambda}$ for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, k_i\}$, such that the type assignment $\Gamma \vdash T_i : \square B_i$ is derivable for all $i \in \{1, \dots, n\}$, then $M[T_1, \dots, T_n/x_1, \dots, x_n] \in \widehat{\Lambda}$.

Proposition 2. *The set $\widehat{\Lambda}$ is the set of modal λ -terms in $\beta\eta\kappa$ -normal form $\text{nf}_{\beta\eta\kappa}(\Lambda)$.*

Proof. By definition, every $\widehat{\Lambda} \subseteq \text{nf}_{\beta\eta\kappa}(\Lambda)$ is $\rightsquigarrow_{\beta\eta\kappa}$ -normal. To prove the converse we proceed by induction on the structure of $M \in \text{nf}_{\beta\eta\kappa}(\Lambda)$:

- if $M = x$, then $M \in \widehat{\Lambda}$ by definition;
- if $M = \lambda x.M' \in \text{nf}_{\beta\eta\kappa}(\Lambda)$, then also $M' \in \text{nf}_{\beta\eta\kappa}(\Lambda)$. By inductive hypothesis, this implies $M' \in \widehat{\Lambda}$. Therefore $\lambda x.M' \in \widehat{\Lambda}$;

$$\begin{array}{c}
\text{ax } \frac{}{\Gamma, x : c \vdash x : c} \quad \text{ex } \frac{\Gamma \vdash M : C}{\sigma(\Gamma) \vdash M : C}^* \quad \text{K}^\square \frac{x_1 : A_1, \dots, x_n : A_n \vdash M : C}{\Delta, y_1 : \square A_1, \dots, y_n : \square A_n \vdash M[x_1, \dots, x_n / y_1, \dots, y_n]_\blacksquare : \square C}^* \\
\rightarrow^\text{ax} \frac{\{ \Gamma, y : B \vdash N_i : A_i \}_{i \in \{1, \dots, n\}}}{\Gamma, y : (\underbrace{A_1, \dots, A_n \rightarrow c}_B) \vdash y N_1 \cdots N_n : c} \$ \quad \rightarrow^\ast_R \frac{\Gamma, x_1 : A_1, \dots, x_n : A_n \vdash M : C}{\Gamma \vdash \lambda x_1^{A_1} \cdots x_n^{A_n}. M : (A_1, \dots, A_n) \rightarrow C} \\
\rightarrow^\text{K} \frac{\{ \Gamma, \Delta \vdash T_{i,j} : A_{i,j} \}_{i \in \{1, \dots, n\}, j \in \{1, \dots, k_i\}} \quad \Gamma, \Delta, x_1 : \square B_1, \dots, x_n : \square B_n \vdash M[x_1, \dots, x_n, \vec{z} / y_1, \dots, y_n, \vec{w}]_\blacksquare : \square C}{\Gamma, f_1 : (A_{1,1}, \dots, A_{1,k_1}) \rightarrow \square B_1, \dots, f_n : (A_{n,1}, \dots, A_{n,k_n}) \rightarrow \square B_n \vdash M[N_1, \dots, N_n, \vec{z} / y_1, \dots, y_n, \vec{w}]_\blacksquare : \square C}^{\dagger, \$} \\
* := \text{permutation over } \{1, \dots, n\} \quad \star := FV(M) = \{x_1, \dots, x_n\} \text{ and } y_1, \dots, y_n \text{ fresh} \\
\$:= \text{each } N_i = f_i T_{i,1} \cdots T_{i,k_i} \text{ for } i \in \{1, \dots, n\} \quad \dagger := \Gamma \text{ contains no formula of the shape } (A_1 \cdots A_n) \rightarrow \square B
\end{array}$$

Fig. 5. Typing rules of the typing system CK^F .

- if $M = PQ \in \text{nf}_{\beta\eta\kappa}(\Lambda)$, then both P and Q are in $\text{nf}_{\beta\eta\kappa}(\Lambda)$ and there is a derivable type assignment $\Gamma \vdash M : C$, and derivable type assignments $\Gamma \vdash P : A \rightarrow C$ and $\Gamma \vdash Q : A$. We have that no \rightsquigarrow_η -rule can be applied to C because $M \in \text{nf}_\eta(\Lambda)$; thus C must be atomic. We know that P cannot be in Λ^\perp since $M \in \text{nf}_\beta(\Lambda)$ and P cannot be in Λ^\square because $\Gamma \vdash P : A \rightarrow C$ is derivable. Then by inductive hypothesis we have that $P = xT_1, \dots, T_n$ for some $T_1, \dots, T_n \in \widehat{\Lambda}$. We conclude that $PQ \in \widehat{\Lambda}$;
- if $M = P[Q_1, \dots, Q_n/x_1, \dots, x_n]_\blacksquare \in \text{nf}_{\beta\eta\kappa}(\Lambda)$, then there is a derivable type assignment $x_1 : B_1, \dots, x_n : B_n \vdash P : C$ and derivable type assignments $\Gamma \vdash Q_i : \square B_i$ for all $i \in \{1, \dots, n\}$. Since $M \in \text{nf}_{\beta\eta\kappa}(\Lambda)$, then no $\rightsquigarrow_{\beta\eta\kappa}$ -rule can be applied to M , nor to P ; thus $P \in \text{nf}_{\beta\eta\kappa}(\Lambda)$. Similarly, since $M \in \text{nf}_{\beta\eta\kappa}(\Lambda)$, then $Q_i \notin \Lambda^\square$ (otherwise we could apply \rightsquigarrow_β^2), $Q_i \in \text{nf}_\beta(\Lambda)$ (since no \rightsquigarrow_β -rule can be applied to Q_i) and Q_i cannot be in $\text{nf}_\eta(\Lambda)$ (because $Q_i : \square B_i$ and otherwise \rightsquigarrow_η -steps could be applied on M) for all $i \in \{1, \dots, n\}$. We conclude that $M \in \widehat{\Lambda}$.

4 A Canonical Type System for CK

In this section we present an alternative typing system for modal λ -terms where each term in $\widehat{\Lambda}$ admits exactly one typing derivation. The rules of this system (we call CK^F) are provided in Figure 5 and are conceived to reduce the non-determinism of the typing process, following the same approach used in designing focused sequent calculi [7,41,11]. Derivations and derivability in CK^F are defined analogously to Definition 1, using rules in CK^F instead of rules in ND_{CK} . We remark that the structural rules of weakening and contraction are admissible in the system.

We can now prove a result of *canonicity* of CK^F with respect to typing derivations of modal λ -terms in $\text{nf}_{\beta\eta\kappa}(\Lambda)$.

Theorem 3. *Let $T \in \widehat{\Lambda}$ and $\Gamma \vdash T : A$ be a derivable type assignment. Then there is a unique (up to ex-rules) derivation of $\Gamma \vdash T : A$ in CK^F .*

Proof. The proof of this theorem follows from the correspondence between the inductive definition of terms in $\widehat{\Lambda}$ (definition 4) and the shape of the typing rules of CK^F .

By definition of $\widehat{\Lambda}$, we have the following cases:

- if $M = x$ is a variable, then $A = a$ must be an atomic formula. Since the sequent $\Gamma \vdash x : a$ is derivable in ND_{CK} , then it can only be the conclusion of a Id -rule if $x : a \in \Gamma$. We conclude since the rule Id is also in CK^F and such a derivation is unique;
- if $T = xN_1 \cdots N_m$, then $x : C = (A_1, \dots, A_n) \rightarrow A$ is in Γ and $N_1, \dots, N_m \in \widehat{\Lambda}$ with size smaller than $|M|$. By inductive hypothesis, for each $i \in \{1, \dots, m\}$ the sequent $\Gamma \vdash N_i : A_i$ is derivable in ND_{CK} , therefore admits a unique derivation \mathcal{D}_i in CK^F . We conclude since we have a unique derivation of $\Gamma \vdash M : A$ starting with a by $\rightarrow_L^{\text{ax}}$ whose premises are the conclusions $\mathcal{D}_1, \dots, \mathcal{D}_n$;
- if $T = \lambda x_1^{A_1} \cdots \lambda x_n^{A_n}.M$ for a $M \neq \lambda y.N$, then $A = (A_1, \dots, A_n) \rightarrow C$ for some types A_1, \dots, A_n, C . Applying n times the rule Abs we know that $\Gamma \vdash \lambda x^{A_1} \cdots \lambda x^{A_n}.M : (A_1, \dots, A_n) \rightarrow C$ iff $\Gamma, x : A_1, \dots, x_n : A_n \vdash M : C$. By induction, we know that there is a unique derivation \mathcal{D}_1 of the sequent $\Gamma, x : A_1, \dots, x_n : A_n \vdash M : C$ in CK^F . We conclude since we have a unique derivation of $\Gamma \vdash M : A$ starting with a by \rightarrow_R^* whose premise is the conclusion of \mathcal{D}_1 ;
- if $T = M[N_1, \dots, N_n/y_1, \dots, y_n]_\blacksquare$, then, we have two cases:
 - $N_i = x_i$ is a variable for all $i \in \{1, \dots, n\}$. In this case a ND_{CK} derivation of $\Gamma \vdash x_i : \square A_i$ can only be made of a single Id -rule. This implies that $x_i : \square A_i \in \Gamma$ for all $i \in \{1, \dots, n\}$; thus $\Gamma = \Delta, x_1 : \square A_1, \dots, x_n : \square A_n$ for a context Δ . Moreover we must have a ND_{CK} -derivation of $x_1 : A_1, \dots, x_n : A_n \vdash M : C$ for a C is such that $\square C = A$; thus, since $|M| < |T|$, there is a unique derivation \mathcal{D}_1 of this latter sequent in CK^F . We conclude since we have a unique derivation of $\Gamma \vdash M : A$ starting with a by K^\square whose premise is the unique derivation of $\Delta, x : A_1, \dots, x_n : A_n \vdash M : C$ in CK^F ;
 - there are some $i \in \{1, \dots, n\}$ such that N_i is of the form $f_i T_{i,1} \cdots T_{i,k_i}$ with $f_i : (A_{i,1}, \dots, A_{i,k_i}) \rightarrow \square B_i$ and $T_{i,1}, \dots, T_{i,k_i} \in \widehat{\Lambda}$. For any $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, k_i\}$ the sequent $\Gamma \vdash T_{i,j} : A_{i,j}$ is derivable in ND_{CK} , then, since the weakening rule is admissible, also $\Gamma, f_1 : (A_{1,1}, \dots, A_{1,k_1}) \rightarrow \square B_1, \dots, f_n : (A_{n,1}, \dots, A_{n,k_n}) \rightarrow \square B_n \vdash N_{i,j} : A_{i,j}$ is derivable. Since $|N_{i,j}| < |M|$, we can conclude by induction the existence of a unique derivation $\mathcal{D}_{i,j}$ in CK^F for this latter sequent. By similar argument, there is also a unique derivation for $\Gamma, \Delta, x_1 : \square B_1, \dots, x_n : \square B_n \vdash M[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_\blacksquare : \square C$ allowing us to conclude the existence of a unique derivation of $\Gamma \vdash M : A$ starting with a by \rightarrow_L^K whose rightmost premise is the conclusion of \mathcal{D}' and whose other premises are the derivations $\mathcal{D}_{i,j}$ with $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, k_i\}$;

5 Game Semantics for CK

In this section we recall definitions and results on the winning innocent strategies for the logic CK defined in [4]. For this purpose, we first recall the construction extending Hyland-Ong arenas [28,43] for intuitionistic propositional formulas to represent formulas containing modalities, and then we recall the characterization of the winning innocent strategies representing proofs in CK. We conclude by proving the full-completeness result between for those strategies by showing a one-to-one correspondence between strategies for type assignments of terms in normal forms and their (unique) typing derivations in CK^F .

5.1 Arenas with Modalities

We recall the definition of arenas with modalities from [4] extending the encoding of arenas from [29,25]. For this purpose, we assume the reader familiar with the definition of *two-color directed graph* (or *2-dag's* for short), i.e., directed acyclic graphs with two disjoint sets of directed edges \rightarrow and \rightsquigarrow (details can be found in [4,25]).

Definition 5. *The arena of a formula F is the 2-dag $\llbracket F \rrbracket$ with vertices are labeled by elements in $\mathcal{L} = \mathcal{A} \cup \{\square\}$ inductively defined as follows:*

$$\llbracket a \rrbracket = a \quad \llbracket A \rightarrow B \rrbracket = \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket \quad \llbracket \square A \rrbracket = \square \rightsquigarrow \llbracket A \rrbracket \quad (7)$$

where a and \square denote the graphs consisting of a single vertex labeled by a and \square respectively, and where the binary operation \rightarrow and \rightsquigarrow on 2-dag's are defined as follows:

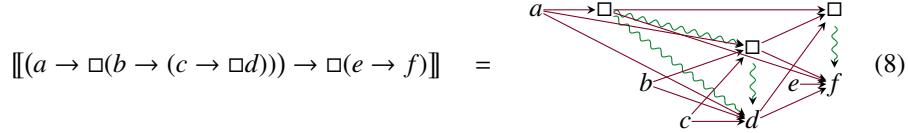
$$\mathcal{G} \rightarrow \mathcal{H} = \langle V_{\mathcal{G}} \uplus V_{\mathcal{H}}, \overset{\mathcal{G} \uplus \mathcal{H}}{\rightarrow} \cup (\vec{R}_{\mathcal{G}} \curvearrowright \vec{R}_{\mathcal{H}}), \overset{\mathcal{G} \uplus \mathcal{H}}{\rightsquigarrow} \rangle \quad \text{and} \quad \mathcal{G} \rightsquigarrow \mathcal{H} = \langle V_{\mathcal{G}} \uplus V_{\mathcal{H}}, \overset{\mathcal{G} \uplus \mathcal{H}}{\rightarrow}, \overset{\mathcal{G} \uplus \mathcal{H}}{\rightsquigarrow} \cup (\vec{R}_{\mathcal{G}} \curvearrowright \vec{R}_{\mathcal{H}}) \rangle \quad \text{with}$$

$$\begin{aligned} V_{\mathcal{G}} \uplus V_{\mathcal{H}} &= \{(v_i, i) \mid i \in \{0, 1\} \text{ and } v_0 \in V_{\mathcal{G}} \text{ and } v_1 \in V_{\mathcal{H}}\} \quad \text{and} \quad \ell((v_i, i)) = \ell(v_i) \\ \overset{\mathcal{G} \uplus \mathcal{H}}{\curvearrowright} &= \left\{ ((v_i, i), (w_j, j)) \mid i \in \{0, 1\} \text{ and } (v_0, w_0) \in \overset{\mathcal{G}}{\curvearrowright} \text{ and } (v_1, w_1) \in \overset{\mathcal{H}}{\curvearrowright} \right\} \quad \text{for each } \curvearrowright \in \{\rightarrow, \rightsquigarrow\} \\ (\vec{R}_{\mathcal{G}} \curvearrowright \vec{R}_{\mathcal{H}}) &= \left\{ ((v, 0), (w, 1)) \mid v \in \vec{R}_{\mathcal{G}}, w \in \vec{R}_{\mathcal{H}} \right\} \quad \text{where} \quad \vec{R}_X := \{v \in V_X \mid v \xrightarrow{X} w \text{ for no } w \in V_X\} \end{aligned}$$

The arena of a sequent $A_1, \dots, A_n \vdash C$ is the arena \mathbf{A} of $\llbracket (A_1, \dots, A_n) \rightarrow C \rrbracket$.

Remark 2. By construction, an arena \mathcal{G} of a formula or a sequent $\Gamma \vdash C$ always admits a unique non \square -labeled vertex in $\vec{R}_{\mathcal{G}}$, i.e., a unique vertex v with $\ell(v) \neq \square$ such that there is no $w \in V_{\mathcal{G}}$ such that $v \xrightarrow{\mathcal{G}} w$.

We draw 2-dag's by representing a vertex v by its label $\ell(v)$. If v and w are vertices of an 2-dag, then we draw $v \rightarrow w$ if $v \rightarrow w$ and $v \rightsquigarrow w$ if $v \rightsquigarrow w$. By means of example, consider the arena below.



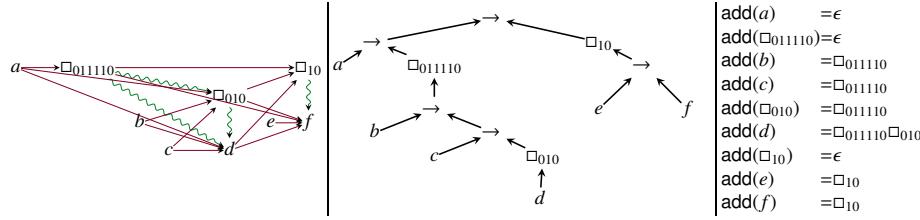
Remark 3. All arenas of the form $\llbracket (A_{\sigma(1)}, \dots, A_{\sigma(n)}) \rightarrow C \rrbracket$ have the same representation for any σ permutation over $\{1, \dots, n\}$. More in general, it can be shown that the arena of any two equivalent formulas modulo Currying $A \rightarrow (B \rightarrow C) \sim B \rightarrow (A \rightarrow C)$ can be depicted by the same arena. However, whenever there may be ambiguity because of the presence of two vertices with the same label, we may represent the vertex $v = ((\dots (v', i_1) \dots), i_n)$ (where $i_1, \dots, i_n \in \{0, 1\}$) by $\ell(v)_{i_1, \dots, i_n}$ instead of simply $\ell(v) = \ell(v')$ (see Example 2).

Definition 6. Let $\llbracket F \rrbracket$ be an arena and v one of its vertices. The depth of v is the number $d(v)$ of vertices in a \rightarrow -path from v to a vertex in $\vec{R}_{\llbracket F \rrbracket}$ ⁶. The address of v is defined as

⁶ As proven in [25,5], arenas are *stratified*, that is, all the \rightarrow -path from a vertex v to any vertex in $\vec{R}_{\llbracket F \rrbracket}$ have the same length. Therefore the number $d(v)$ is well-defined.

the unique sequence of modal vertices $\text{add}(v) = m_1, \dots, m_h$ in $V_{\llbracket F \rrbracket}$ corresponding to the sequence of modalities in the path in the formula tree of F connecting the node of v to the root. If $\text{add}(v) = m_1, \dots, m_h$, we denote by $\text{add}^k(v) = m_k$ its k^{th} element and we call the height of v (denoted h_v) the number of elements in $\text{add}(v)$.

Example 2. Below an alternative representation of its arena of the formula $(a \rightarrow \square(b \rightarrow (c \rightarrow \square d))) \rightarrow \square(e \rightarrow f)$ in Equation (8) where the ambiguity of the vertex representation is avoided by the use of indices, the corresponding formula-tree, and the complete list of the addresses of all vertices in this arena.



5.2 Games and Winning Innocent Strategies

In this subsection, we briefly recall the definitions of games and winning strategies from [4] required to make the paper self-contained. Note that differently from the previous works, we here include the additional information of the *pointer function* in the definition of views. This information is crucial for the results in Section 4 where we provide a one-to-one correspondence between our winning strategies and modal λ -terms.

Definition 7. Let \mathbf{A} be an arena. We call a move an occurrence of a vertex v of \mathbf{A} with $\ell(v) \neq \square$. The polarity of a move v is the parity of $d(v)$: a move is a \circ -move (resp. a \bullet -move) if $d(v)$ is even (resp. odd).

A pointed sequence in \mathbf{A} is a pair $\mathbf{p} = \langle \mathbf{s}, f \rangle$ where $\mathbf{s} = s_0, \dots, s_n$ is a finite sequences of moves in \mathbf{A} and a pointer function $f: \{1, \dots, n\} \rightarrow \{0, \dots, n-1\}$ such that $f(i) < i$ and $s_i \xrightarrow{\mathbf{A}} s_{f(i)}$. The length of \mathbf{p} (denoted $|\mathbf{p}|$) is defined as the length of \mathbf{s} , that is, $|\mathbf{p}| = n + 1$. Note that we also use ϵ to denote the empty pointed sequence $\langle \epsilon, \emptyset \rangle$.

Remark 4. It follows by definition of view that the player \circ (resp. \bullet) can only play vertices whose $d(v)$ is even (resp. odd). For this reason, for each $v \in V_{\mathcal{G}}$ we write v° (resp. v^\bullet) if the parity of $d(v)$ even (resp. odd).

Note that the parity of a modality in the address of a move may not be the same as the parity of the move itself. By means of example, consider the vertex c in Example 2 which belongs in the scope of two modalities $\square 011110$ and $\square 010$ with odd parity.

Given two pointed sequences $\mathbf{p} = \langle \mathbf{s}, f \rangle$ and $\mathbf{p}' = \langle \mathbf{s}', f' \rangle$ in \mathbf{A} , we write $\mathbf{p} \sqsubseteq \mathbf{p}'$ whenever \mathbf{s} is a prefix of \mathbf{s}' (thus $|\mathbf{s}| \leq |\mathbf{s}'|$) and $f(i) = f'(i)$ for all $i \in \{1, \dots, |\mathbf{p}'|\}$ and we say that \mathbf{p} is a predecessor of \mathbf{p}' if $\mathbf{p} \sqsubseteq \mathbf{p}'$ and $|\mathbf{p}| = |\mathbf{p}'| - 1$.

Definition 8. Let \mathbf{A} be an arena. A play on \mathbf{A} is a pointed sequence $\mathbf{p} = \langle \mathbf{s}, f \rangle$ such that, either $\mathbf{s} = \epsilon$, or \mathbf{s}_i and \mathbf{s}_{i+1} have opposite polarities for all $i \in \{0, \dots, |\mathbf{p}| - 1\}$.

Arena	$\llbracket (\Box a) \rightarrow a \rrbracket = \begin{array}{c} \square^* \\ \swarrow \quad \searrow \\ a^* \end{array}$	$\llbracket (\Box a \rightarrow \Box b) \rightarrow \Box(a \rightarrow b) \rrbracket = \begin{array}{c} \square^* \\ \swarrow \quad \searrow \\ a^* \end{array} \quad \begin{array}{c} \square^* \\ \swarrow \quad \searrow \\ b^* \end{array} \quad \begin{array}{c} \square^* \\ \swarrow \quad \searrow \\ a^* \end{array}$
WIS	$S_1 = \{\epsilon, a^*, a^*a^*\}$	$S_2 = \{\epsilon, b^*, b^*b^*, b^*b^*a^*, b^*b^*a^*a^*\}$
(failed) Derivation	$\frac{\text{FAIL}}{\vdash^R \Box a \vdash a \quad \vdash \Box a \rightarrow a}$	$\frac{\text{FAIL} \quad \frac{\text{ax } b \vdash b}{w \frac{b, a \vdash b}{\vdash b \vdash a \rightarrow b}} \quad \frac{\Box^0 \vdash a \quad K^0 \vdash \Box^0 a \quad \Box^* b \vdash \Box^0(a \rightarrow b)}{\vdash b \vdash a \rightarrow b}}{\frac{\Box^0 a \rightarrow \Box^* b \vdash \Box^0(a \rightarrow b)}{\vdash (\Box^0 a \rightarrow \Box^* b) \rightarrow \Box^0(a \rightarrow b)}}$

Fig. 6. Examples of WISs for arenas not corresponding to proofs.

The game of A (denoted \mathcal{G}_A) is the set of prefix-closed sets of plays over A .

A view is a play $p = \langle s, f \rangle$ such that either $p = \epsilon$ or the following properties hold:

- p is \circ -shortsighted : $f(2k) = 2k - 1$ for every $2k \in \{2, \dots, |p|\}$;
- p is \bullet -uniform : $\ell(s_{2k+1}) = \ell(s_{2k})$ for every $2k + 1 \in \{0, \dots, |p|\}$.

A winning innocent strategy (or WIS for short) for the game \mathcal{G}_A is a finite non-empty prefix-closed set S of views in \mathcal{G}_A such that:

- S is \circ -complete: if $p \in S$ and p has odd length,
then every successor of p (in \mathcal{G}_A) is also in S ;
- p is \bullet -total: if $p \in S$ and p has even length,
then exactly one successor of p (in \mathcal{G}_A) is in S ;

A view is maximal in S if it is not prefix of any other view in S . S is trivial if $S = \{\epsilon\}$. We say that S is a WIS for a sequent $A_1, \dots, A_n \vdash C$ if S is a WIS for $\llbracket A_1, \dots, A_n \vdash C \rrbracket$.

The definition of WIS above is a reformulation of the one in the literature of game semantics for intuitionistic propositional logic [28,13,25]. In presence of modalities, this definition requires to be refined to guarantee the possibility of gather modalities in batches corresponding to the modalities introduced by a single application of the K^\Box (see Figure 2). By means of example, consider the following arenas and their corresponding WISs, which cannot represent valid proofs in CK because of the impossibility of applying rules handling the modalities in a correct way.

Example 3. Consider the formulas $F_1 = (\Box a) \rightarrow a$ and $F_2 = (\Box a \rightarrow \Box b) \rightarrow \Box(a \rightarrow b)$ and their arenas in Figure 6. The set of views S_1 and S_2 are WISs for F_1 and F_2 respectively. However, these formulas are not provable in SCK because the proof search fails (see Figure 6). In particular, in the first case, no K^\Box can be applied because only there is a mismatch between the modalities on the left-hand side and on the right-hand side of the sequent; in the second case the problem is more subtle and, intuitively, is related to the fact that each K^\Box can remove only a single \Box° at a time, corresponding to the modality of the unique formula on the right-hand side of the sequent.

Therefore, in order to capture provability in CK, the notion of winning strategies has to be refined as follows.

Definition 9. Let $\mathbf{p} = (\mathbf{s}, f)$ be a view in a strategy \mathcal{S} on an arena \mathbf{A} , and let $\mathbf{h}_{\mathbf{p}} = 1 + \max\{\mathbf{h}_v \mid v \in \mathbf{p}\}$. We define the batched view of \mathbf{p} as the $\mathbf{h}_{\mathbf{p}} \times n$ matrix $\mathcal{F}(\mathbf{p}) = (\mathcal{F}(\mathbf{p})_0, \dots, \mathcal{F}(\mathbf{p})_n)$ with elements in $V_{\mathcal{G}} \cup \{\epsilon\}$ such that the each column $\mathcal{F}(\mathbf{p})_i$ is defined as follows:

$$\mathcal{F}(\mathbf{p})_i = \begin{pmatrix} \mathcal{F}(\mathbf{p})_i^{\mathbf{h}_{\mathbf{p}}} \\ \vdots \\ \mathcal{F}(\mathbf{p})_i^0 \end{pmatrix} \quad \text{where} \quad \begin{cases} \mathcal{F}(\mathbf{p})_i^{\mathbf{h}_{\mathbf{p}}} = \text{add}^{\mathbf{h}_{\mathbf{p},i}}(\mathbf{p}_i), \dots, \mathcal{F}(\mathbf{p})_i^{\mathbf{h}_{\mathbf{p}} - \mathbf{h}_{\mathbf{p},i} + 1} = \text{add}^1(\mathbf{p}_i) \\ \mathcal{F}(\mathbf{p})_i^{\mathbf{h}_{\mathbf{p}} - \mathbf{h}_{\mathbf{p},i}} = \epsilon, \dots, \mathcal{F}(\mathbf{p})_i^1 = \epsilon \\ \mathcal{F}(\mathbf{p})_i^0 = \mathbf{p}_i \end{cases}$$

We say that \mathbf{p} is well-batched if $|\text{add}(\mathbf{s}_{2k})| = |\text{add}(\mathbf{s}_{2k+1})|$ for every $2k \in \{0, \dots, |\mathbf{p}| - 1\}$. Each well-batched view \mathbf{p} induces an equivalence relation $\overset{\mathcal{G}_{\mathbf{p}}}{\sim}$ over $V_{\mathcal{G}}$ generated by:

$$u \overset{\mathcal{G}_{\mathbf{p}}}{\sim} w \quad \text{iff} \quad u = \mathcal{F}(\mathbf{p})_{2k}^h \text{ and } w = \mathcal{F}(\mathbf{p})_{2k+1}^h \text{ for a } 2k < n - 1 \text{ and a } h \leq \mathbf{h}_{\mathbf{p}} \quad (9)$$

A WIS \mathcal{S} is linked if it contains only well-batched views and if for every $\mathbf{p} \in \mathcal{S}$ the $\overset{\mathcal{G}_{\mathbf{p}}}{\sim}$ -classes are of the shape $\{v_1^\bullet, \dots, v_n^\bullet, w^\circ\}$.

A CK-winning innocent strategy (or CK-WIS for short) is a linked WIS \mathcal{S} .⁷

Example 4. Consider the arenas in Figure 6. The batched view of the (unique) maximal views in \mathcal{S}_1 and \mathcal{S}_2 are $\begin{pmatrix} \epsilon & \square^\bullet \\ a^\circ & a^\bullet \end{pmatrix}$ and $\begin{pmatrix} \square^\circ & \square^\bullet & \square^\circ & \square^\circ \\ b^\circ & b^\bullet & a^\circ & a^\bullet \end{pmatrix}$ respectively. The first is not well-batched because a° has height 0 while a^\bullet has height 1, while the second, even if well-batched, is not linked because the $\overset{\mathcal{G}_{\mathbf{p}}}{\sim}$ -class $\{\square_{10}^\circ, \square_{010}^\bullet, \square_{000}^\circ\}$ contains two \square° .

The definition of CK-WISs allows us to obtain a full-completeness result with respect to CK which, together with the good compositionality properties of CK-WISs shown in [4,10], provides a full-complete denotational semantics for the logic CK. That is, every given CK-WIS is the encoding of a derivation in CK, and if a derivation \mathcal{D} reduces via cut-elimination to a derivation \mathcal{D}' , then they are encoded by the same CK-WIS.

Theorem 4 ([4]). The set of CK-WISs is a full-complete denotational model for CK.

5.3 Full Completeness for Modal Lambda Terms in Normal Form

We can prove the full completeness result using the type system CK^F and relying on Theorem 3. For this purpose, we have to extend the definition of α -equivalence from terms to type assignments in order to avoid technicality in our proofs, since in arenas we keep no track of variable names. For example, consider the α -equivalent terms $\lambda x.x$ and $\lambda y.y$ whose derivation should be considered non-equivalent due to the fact that α -equivalence does not extends to type assignments, therefore the two occurrence of the axiom rule with conclusion $x : a \vdash x : a$ and $y : a \vdash y : a$ should be considered distinct.⁸

⁷ We here provide a simpler definition of CK-WISs w.r.t. the one in [4]. In fact, we are able here to simplify this definition because we are considering the \diamond -free fragment of CK.

⁸ Note that another possible way to deal with this problem is to label non-modal vertices of arenas by pairs of propositional atoms and variables instead of propositional variables only.

$$\begin{aligned}
& \left\{ \left\{ \text{ax} \frac{}{\Gamma, x : c^* \vdash x : c^\circ} \right\} \right\} = \{\epsilon, c^\circ, c^\circ c^*\} \quad \left\{ \left\{ \frac{\mathcal{D}' \mathbb{T}}{\Gamma, x_1 : A_1, \dots, x_n : A_n \vdash M : C} \right\} \right\} = \{\mathcal{D}'\} \\
& \left\{ \left\{ \frac{\mathcal{D}_i \mathbb{T}}{\Gamma, (A_1, \dots, A_n) \rightarrow c \vdash N_i : A_i}_{i \in \{1, \dots, n\}} \right\} \right\} = \{\epsilon, c^\circ, c^\circ c^*\} \cup \{c^\circ c^* p \mid \epsilon \neq p \in \{\mathcal{D}_i\} \text{ for a } i \in \{1, \dots, n\}\} \\
& \left\{ \left\{ \frac{\mathcal{D}' \mathbb{T}}{\text{ex} \frac{\sigma(\Gamma) \vdash M : C}{\Gamma \vdash M : C}} \right\} \right\} = \{f_\sigma(p) \mid p \in \{\mathcal{D}'\}, f_\sigma \text{ isomorphism between } \llbracket \Gamma \vdash M : C \rrbracket \text{ and } \llbracket \sigma(\Gamma) \vdash M : C \rrbracket\} \\
& \text{where } f_\sigma(p) \text{ is the view obtained by applying } f_\sigma \text{ to each move in } p \text{ (and updating its pointer accordingly)} \\
& \left\{ \left\{ \frac{\mathcal{D}' \mathbb{T}}{\kappa^\square \frac{x_1 : A_1, \dots, x_n : A_n \vdash M[x_1, \dots, x_n/y_1, \dots, y_n]_\blacksquare : \square C}{\Gamma, y_1 : \square A_1, \dots, y_n : A_n \vdash M[x_1, \dots, x_n/y_1, \dots, y_n]_\blacksquare : \square C}} \right\} \right\} = \{\mathcal{D}'\} \\
& \left\{ \left\{ \frac{\mathcal{D}_{i,j} \mathbb{T}}{\text{ex} \frac{\Gamma, \Delta \vdash T_{i,j} : A_{i,j}}{\Gamma, f_1 : (A_{1,1}, \dots, A_{1,k_1}) \rightarrow \square B_1, \dots, f_n : (A_{n,1}, \dots, A_{n,k_n}) \rightarrow \square B_n \vdash M[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_\blacksquare : \square C}} \right\} \right\} = \{\mathcal{D}_0\} \cup \left(\bigcup_{i \in \{1, \dots, n\}} \{c^\circ b_i^* p \mid \epsilon \neq p \in \{\mathcal{D}_{i,j}\} \text{ for a } j \in \{1, \dots, k_i\}\} \right) \\
& \text{where } c^\circ \text{ (resp. } b_i^* \text{) is the unique non-}\square \text{ vertex in } \vec{R}_{\llbracket \square C \rrbracket} \text{ (resp. in } \llbracket \square B_i \rrbracket).
\end{aligned}$$

Fig. 7. Rules to construct a CK-WIS from a type derivation in CK^F . For reasons of readability, we assume there is an implicit map identifying the moves in the arenas of the type assignment in the premises with the moves in the arena of the type assignment in the conclusion. Note that c° and c^* are occurrences of the same atom c , but we have decorated them to improve readability.

Definition 10. Let $A_1, \dots, A_n \vdash C$ be a sequent. We define $\Lambda(\Gamma \vdash C)$ as the set of terms M such that the typing derivation $x_1 : A_1, \dots, x_n : A_n \vdash M : C$ is derivable, that is,

$$\Lambda(\Gamma \vdash C) = \{M \in \Lambda \mid x_1 : A_1, \dots, x_n : A_n \vdash M : C \text{ is derivable for some } x_1, \dots, x_n\}.$$

If $M, N \in \Lambda(\Gamma \vdash C)$, we define $M =_{\alpha}^{\Gamma;C} N$ as the smallest equivalence relation generated by the rule $\frac{M \{z_1, \dots, z_n/x_1, \dots, x_n\} = N \{z_1, \dots, z_n/y_1, \dots, y_n\}}{M =_{\alpha}^{\Gamma;C} N}$ z_i is fresh.

From now on, we consider derivations up the α -equivalence defined above, that is, we consider derivations up to renaming of the variables occurring in a typing context.

Lemma 6. Let \mathcal{S} be a non-trivial CK-WIS over the arena $\llbracket \Gamma \vdash C \rrbracket$. Then there is a canonically defined $T_{\mathcal{S}} \in \widehat{\Lambda} \cap \Lambda(\Gamma \vdash C)$ admitting a unique typing derivation in CK^F .

Proof. We define a term $T_{\mathcal{S}}$ and a derivation $\mathcal{D}_{\mathcal{S}}(\Gamma)$ in CK^F of the type assignment $\Gamma \vdash T_{\mathcal{S}} : C$ by induction on the lexicographic order over the pairs $(|\mathcal{S}|, |C|)$:

1. if $C = c$ is atomic, then \mathcal{S} must contain the CK-WIS $\{\epsilon, c^\circ, c^\circ c^*\}$ because c° is the unique sink of $\llbracket \Gamma \vdash C \rrbracket$ and c^* is the unique (by \bullet -totality) \bullet -move justified by the unique previous \circ -move in \mathcal{S} . Note that by the well-batched condition we must have $\text{add}(c^\circ) = \text{add}(c^*) = \epsilon$. Then

- (a) either $\mathcal{S} = \{\epsilon, c, cc\}$, then $\Gamma = \Delta, c$ for a sequent Δ such that no move in Δ occurs in \mathcal{S} (because of \circ -completeness). In this case $T = x$ and

$$\mathcal{D}_{\mathcal{S}}(T) = \text{ax } \frac{}{\Delta, x : c \vdash x : c}$$

- (b) or, since \mathcal{S} is prefix-closed and well-batched, $\mathcal{S} = (\{\epsilon, c^\circ, c^\circ c^\bullet\} \cup_{i=0}^n cc\mathcal{S}_i)$ for some CK-WISs \mathcal{S}_i for a sequent $\Gamma \vdash A_i$ for each $i \in \{1, \dots, n\}$. Then $\Gamma = \Delta, (A_0, \dots, A_n) \rightarrow c$. In this case $T = yN_1 \cdots N_n$ and

$$\mathcal{D}_{\mathcal{S}}(T) = \text{ax}_{\text{L}} \frac{\frac{\mathcal{D}_{\mathcal{S}_1}(N_1) \parallel}{\Delta, y : (A_1, \dots, A_n) \rightarrow c \vdash N_1 : A_1} \cdots \frac{\mathcal{D}_{\mathcal{S}_n}(N_n) \parallel}{\Delta, y : (A_1, \dots, A_n) \rightarrow c \vdash N_n : A_n}}{\Delta, y : (A_1, \dots, A_n) \rightarrow c \vdash c}$$

2. if $C = (A_1, \dots, A_n) \rightarrow B$, then $T = \lambda x_1 \cdots \lambda x_n. T'$ and

$$\mathcal{D}_{\mathcal{S}}(T) = \text{ax}_{\text{R}}^* \frac{\frac{\mathcal{D}_{\mathcal{S}(T')} \parallel}{\Gamma, x_1 : A_1, \dots, x_n : A_n \vdash T' : B}}{\Gamma \vdash T : (A_1, \dots, A_n) \rightarrow B}$$

3. $C = \square^\circ A$ is a \square -formula, then, if a sink v of $\llbracket \Gamma \rrbracket$ occurs as a move in \mathcal{S} , then it must be justified by a sink of $\llbracket \square A \rrbracket$. Therefore, by the well-batched condition, v must be in the scope of a \square . We have two cases:

- (a) either $\Gamma = \Sigma, \square \Delta$ and no move in Σ occurs in \mathcal{S} . In this case we have that $T = M[x_1, \dots, x_n/y_1, \dots, y_n]_\blacksquare$ and

$$\mathcal{D}_{\mathcal{S}}(T) = \text{K}^\square \frac{x_1 : A_1, \dots, x_n : A_n \vdash M : C}{\Delta, y_1 : \square A_1, \dots, y_n : A_n \vdash M[x_1, \dots, x_n/y_1, \dots, y_n]_\blacksquare : \square C}$$

- (b) or $\Gamma = \Delta, (A_{1,1}, \dots, A_{1,k_1}) \rightarrow \square B_1, \dots, (A_{n,1}, \dots, A_{n,k_n}) \rightarrow \square B_n$ for some k_1, \dots, k_n such that $k_1 + \dots + k_n > 0$ and a sequent Δ such that if $\square^\bullet D$ or $(A_1, \dots, A_n) \rightarrow \square^\bullet D$ is in Δ , then $\square^\bullet \not\approx_{\mathcal{S}} \square^\circ$. In this case, by similar reasoning of (1.1b), there are for some CK-WISs \mathcal{S}_i for the sequent $\Gamma \vdash A_i$ for each $i \in \{1, \dots, n\}$ and a CK-WIS \mathcal{S}_0 for the sequent $\Gamma, \Delta, \square B_1, \dots, \square B_n \vdash \square C$ such that $\mathcal{S} = (\mathcal{S}_0 \cup (\bigcup_{i=0}^n \sigma_i \mathcal{S}_i))$. Therefore $T = M[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_\blacksquare$ and $\mathcal{D}_{\mathcal{S}}(T)$ is the following derivation

$$\text{K}^\square \frac{\left\{ \begin{array}{c} \mathcal{D}_{\mathcal{S}_{i,j}}(T_{i,j}) \parallel \\ \Gamma, \Delta \vdash T_{i,j} : A_{i,j} \end{array} \right\}_{\substack{i \in \{1, \dots, n\}, \\ j \in \{1, \dots, k_i\}}} \quad \mathcal{D}_{\mathcal{S}_0}(M[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_\blacksquare) \parallel}{\Gamma, \Delta, \Delta' \vdash M[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_\blacksquare : \square C}$$

where $\Delta = f_1 : (A_{1,1}, \dots, A_{1,k_1}) \rightarrow \square B_1, \dots, f_n : (A_{n,1}, \dots, A_{n,k_n}) \rightarrow \square B_n$ and $\Delta' = x_1 : \square B_1, \dots, x_n : \square B_n$.

Theorem 5. *There is a one-to-one correspondence between terms in $\widehat{\Lambda} \cap \Lambda(\Gamma \vdash C)$ and CK-WIS for $\Gamma \vdash C$.*

Proof. Lemma 6 ensures that one CK-WIS \mathcal{S} for $\Gamma \vdash C$, we can define a (unique) typing derivation $\mathcal{D}_{\mathcal{S}}$ in CK^F of a term $T_{\mathcal{S}} \in \widehat{\Lambda} \cap \Lambda(\Gamma \vdash C)$.

Conversely, given a type assignment $\Gamma \vdash T : C$ for a $T \in \widehat{\Lambda}$, then, we can uniquely define is a derivation \mathcal{D}_T in CK^F . Thus, by Theorem 3, the type assignment $\Gamma \vdash T : C$ is derivable. We define \mathcal{S}_T as the CK-WIS defined by induction on the number of rules in \mathcal{D}_T using the rules in Figure 7.

We conclude since we have that $\mathcal{S}_{T_S} = \mathcal{S}$ and $T_{\mathcal{S}_T} = T$ by definition.

6 Conclusion

In this paper we introduced a new modal λ -calculus for the \Diamond -free fragment of the constructive modal logic CK (without conjunction or disjunction). This lambda calculus builds on the work in [31], by adding a restricted η -reduction as well as two new reduction rules dealing with the explicit substitution constructor used to model the modality \Box . We proved normalization and confluence for this calculus and we provide a one-to-one correspondence between the set of terms in normal form and the set of winning strategies for the logic CK introduced in [4].

We foresee the possibility of extending the result presented in this paper to the entire disjunction-free fragment of CK, for which winning strategies are already defined in [4] are a fully complete denotational semantics. For this purpose, we should consider additional term constructors for terms whose type is a conjunction, as well as a new Let-like operator to model terms whose type is the modality \Diamond -formula similar to the one proposed in [9]. For this reason, in future works we plan to reformulate our lambda-calculus in the light of the novel line of research on calculi with explicit substitutions [33,34,2,1]. This approach would allow us to simplify some of the technicalities and achieve a more elegant operational semantics. Another interesting prospective is to extend our approach to operational semantics to the Fitch-style modal λ -calculus studied in [52].

At the same time, we plan to make explicit that our game semantics provides a concrete model for the *cartesian closed categories* provided with a *strong monoidal endofunctor* [9,32]. Indeed, categorical semantics of the calculus in [9] is modeled by means of *cartesian closed categories* equipped with a *strong monoidal endofunctor* taking into account the proof-theoretical behavior of the \Box -modality. We further conjecture that the syntactic category obtained via the quotient of modal terms modulo the relations we introduce in this paper is indeed a *free cartesian closed category* on a set of atoms with a *strong monoidal endofunctor*.

References

1. Accattoli, B.: Exponentials as substitutions and the cost of cut elimination in linear logic. In: Baier, C., Fisman, D. (eds.) LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022. pp. 49:1–49:15. ACM (2022). <https://doi.org/10.1145/3531130.3532445>, <https://doi.org/10.1145/3531130.3532445>

2. Accattoli, B., Bonelli, E., Kesner, D., Lombardi, C.: A nonstandard standardization theorem. Association for Computing Machinery, New York, NY, USA (2014). <https://doi.org/10.1145/2535838.2535886>, <https://doi.org/10.1145/2535838.2535886>
3. Acclavio, M.: Proof diagrams for multiplicative linear logic: Syntax and semantics. *Journal of Automated Reasoning* **63**(4), 911–939 (2019). <https://doi.org/10.1007/s10817-018-9466-4>, <https://doi.org/10.1007/s10817-018-9466-4>
4. Acclavio, M., Catta, D., Straßburger, L.: Game semantics for constructive modal logic. In: International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. pp. 428–445. Springer (2021)
5. Acclavio, M., Straßburger, L.: Combinatorial Proofs for Constructive Modal Logic. In: Advances in Modal Logic 2022. Rennes, France (Aug 2022), <https://hal.inria.fr/hal-03909538>
6. Alechina, N., Mendler, M., de Paiva, V., Ritter, E.: Categorical and kripke semantics for constructive S4 modal logic. In: Fribourg, L. (ed.) Computer Science Logic, 15th International Workshop, CSL 2001. 10th Annual Conference of the EACSL, Paris, France, September 10-13, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2142, pp. 292–307. Springer (2001). https://doi.org/10.1007/3-540-44802-0_21, https://doi.org/10.1007/3-540-44802-0_21
7. Andreoli, J.M.: Focussing and proof construction. *Annals of Pure and Applied Logic* **107**, 131–163 (2001)
8. Barendregt, H.P., Dekkers, W., Statman, R.: Lambda Calculus with Types. Perspectives in logic, Cambridge University Press (2013), <http://www.cambridge.org/de/academic/subjects/mathematics/logic-categories-and-sets/lambda-calculus-types-perspectives-in-logic>
9. Bellin, G., De Paiva, V., Ritter, E.: Extended Curry-Howard correspondence for a basic constructive modal logic. In: In Proceedings of Methods for Modalities (05 2001)
10. Catta, D.: Les preuves vues comme des jeux et réciprocement: sémantique dialogique de langages naturel ou logiques. (Proofs as games and games as proofs: dialogical semantics of logical and natural languages). Ph.D. thesis, University of Montpellier, France (2021), <https://tel.archives-ouvertes.fr/tel-03588308>
11. Chaudhuri, K., Marin, S., Straßburger, L.: Modular focused proof systems for intuitionistic modal logics. In: Kesner, D., Pientka, B. (eds.) 1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal. LIPIcs, vol. 52, pp. 16:1–16:18. Leibniz-Zentrum fuer Informatik (2016)
12. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic* **44**(1), 36–50 (1979)
13. Danos, V., Herbelin, H., Regnier, L.: Game semantics & abstract machines. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996. pp. 394–405. IEEE Computer Society (1996). <https://doi.org/10.1109/LICS.1996.561456>, <https://doi.org/10.1109/LICS.1996.561456>
14. Das, A., Marin, S.: Brouwer meets kripke: constructivising modal logic. <https://prooftheory.blog/2022/08/19/brouwer-meets-kripke-constructivising-modal-logic/>, posted on August 19 2022
15. Das, A., Pous, D.: Non-wellfounded proof theory for (Kleene+action)(algebras+lattices). In: Computer Science Logic (CSL). Birmingham, United Kingdom (Sep 2018). <https://doi.org/10.4230/LIPIcs.CSL.2018.19>, <https://hal.archives-ouvertes.fr/hal-01703942>

16. Davies, R., Pfenning, F.: A modal analysis of staged computation. *J. ACM* **48**(3), 555–604 (2001). <https://doi.org/10.1145/382780.382785>, <https://doi.org/10.1145/382780.382785>
17. Di Cosmo, R., Kesner, D.: Combining algebraic rewriting, extensional lambda calculi, and fixpoints. *Theoretical Computer Science* **169**(2), 201–220 (1996). [https://doi.org/https://doi.org/10.1016/S0304-3975\(96\)00121-1](https://doi.org/https://doi.org/10.1016/S0304-3975(96)00121-1)
18. Došen, K.: Identity of proofs based on normalization and generality **9**, 477–503 (2003)
19. Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.* **2**(3), 241–266 (1982). [https://doi.org/10.1016/0167-6423\(83\)90017-5](https://doi.org/10.1016/0167-6423(83)90017-5), [https://doi.org/10.1016/0167-6423\(83\)90017-5](https://doi.org/10.1016/0167-6423(83)90017-5)
20. Fairtlough, M., Mendler, M.: Propositional lax logic. *Information and Computation* **137**(1), 1–33 (1997)
21. Fitch, F.: Intuitionistic modal logic with quantifiers. *Portugaliae Mathematica* **7**(2), 113–118 (1948)
22. Girard, J.Y.: Linear logic. *Theoretical Computer Science* **50**, 1–102 (1987). [https://doi.org/10.1016/0304-3975\(87\)90045-4](https://doi.org/10.1016/0304-3975(87)90045-4)
23. Girard, J.Y.: Proof-nets : the parallel syntax for proof-theory. In: Ursini, A., Agliano, P. (eds.) *Logic and Algebra*. Marcel Dekker, New York (1996)
24. Guglielmi, A., Gundersen, T., Parigot, M.: A Proof Calculus Which Reduces Syntactic Bureaucracy. In: Lynch, C. (ed.) *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*. LIPIcs, vol. 6, pp. 135–150. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2010). <https://doi.org/10.4230/LIPIcs.RTA.2010.135>, <http://drops.dagstuhl.de/opus/volltexte/2010/2649>
25. Heijltjes, W., Hughes, D., Straßburger, L.: Intuitionistic proofs without syntax. In: LICS 2019 - 34th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 1–13. IEEE, Vancouver, Canada (Jun 2019). <https://doi.org/10.1109/LICS.2019.8785827>, <https://hal.inria.fr/hal-02386878>
26. Heilala, S., Pientka, B.: Bidirectional decision procedures for the intuitionistic propositional modal logic IS4. In: International Conference on Automated Deduction. pp. 116–131. Springer (2007)
27. Hughes, D.: Proofs Without Syntax. *Annals of Mathematics* **164**(3), 1065–1076 (2006). <https://doi.org/10.4007/annals.2006.164.1065>
28. Hyland, J.M.E., Ong, C.L.: On full abstraction for PCF: i, ii, and III. *Inf. Comput.* **163**(2), 285–408 (2000). <https://doi.org/10.1006/inco.2000.2917>, <https://doi.org/10.1006/inco.2000.2917>
29. Hyland, J.M.E., Ong, C.H.L.: On full abstraction for PCF: I. Models, observables and the full abstraction problem, II. Dialogue games and innocent strategies, III. A fully abstract and universal game model. *Information and Computation* **163**, 285–408 (2000)
30. Jay, C.B., Ghani, N.: The virtues of eta-expansion. *J. Funct. Program.* **5**(2), 135–154 (1995). <https://doi.org/10.1017/S0956796800001301>, <https://doi.org/10.1017/S0956796800001301>
31. Kakutani, Y.: Call-by-name and call-by-value in normal modal logic. In: Shao, Z. (ed.) *Programming Languages and Systems*. pp. 399–414. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
32. Kavvos, G.A.: Dual-context calculi for modal logic. *Log. Methods Comput. Sci.* **16**(3) (2020). [https://doi.org/10.23638/LMCS-16\(3:10\)2020](https://doi.org/10.23638/LMCS-16(3:10)2020), <https://lmcs.episciences.org/6722>

33. Kesner, D.: The theory of calculi with explicit substitutions revisited. In: Duparc, J., Henzinger, T.A. (eds.) Computer Science Logic. pp. 238–252. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
34. Kesner, D.: A theory of explicit substitutions with safe and full composition. *Log. Methods Comput. Sci.* **5**(3) (2009), <http://arxiv.org/abs/0905.2539>
35. Kojima, K.: Semantical study of intuitionistic modal logics. Ph.D. thesis, Kyoto University (2012)
36. Kozen, D.: Results on the propositional mu-calculus. *Theor. Comput. Sci.* **27**, 333–354 (1983). [https://doi.org/10.1016/0304-3975\(82\)90125-6](https://doi.org/10.1016/0304-3975(82)90125-6), [https://doi.org/10.1016/0304-3975\(82\)90125-6](https://doi.org/10.1016/0304-3975(82)90125-6)
37. Krivine, J.: Lambda-calculus, types and models. Ellis Horwood series in computers and their applications, Masson (1993)
38. Kuznets, R., Marin, S., Straßburger, L.: Justification logic for constructive modal logic. *Journal of Applied Logics: IfCoLog Journal of Logics and their Applications* **8**(8), 2313–2332 (2021), <https://hal.inria.fr/hal-01614707>
39. Mendler, M., Scheele, S.: Cut-free Gentzen calculus for multimodal CK. *Information and Computation* **209**(12), 1465–1490 (2011)
40. Meyer, J.J., Veltmanw, F.: Intelligent agents and common sense reasoning. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) *Handbook of Modal Logic*, Studies in Logic and Practical Reasoning, vol. 3, pp. 991 – 1029. Elsevier (2007). [https://doi.org/https://doi.org/10.1016/S1570-2464\(07\)80021-8](https://doi.org/https://doi.org/10.1016/S1570-2464(07)80021-8), <http://www.sciencedirect.com/science/article/pii/S1570246407800218>
41. Miller, D., Volpe, M.: Focused labeled proof systems for modal logic. In: Logic for Programming, Artificial Intelligence, and Reasoning: 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings. pp. 266–280. Springer (2015)
42. Mints, G.E.: Closed categories and the theory of proofs. *Journal of Soviet Mathematics* (1981). <https://doi.org/10.1007/BF01404107>
43. Murawski, A.S., Ong, C.L.: Discreet games, light affine logic and PTIME computation. In: Clote, P., Schwichtenberg, H. (eds.) Computer Science Logic, 14th Annual Conference of the EACSL, Fischbachau, Germany, August 21–26, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1862, pp. 427–441. Springer (2000). https://doi.org/10.1007/3-540-44622-2_29, https://doi.org/10.1007/3-540-44622-2_29
44. Pfenning, F., Davies, R.: A judgmental reconstruction of modal logic. *Math. Struct. Comput. Sci.* **11**(4), 511–540 (2001). <https://doi.org/10.1017/S0960129501003322>, <https://doi.org/10.1017/S0960129501003322>
45. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977. pp. 46–57. IEEE Computer Society (1977). <https://doi.org/10.1109/SFCS.1977.32>, <https://doi.org/10.1109/SFCS.1977.32>
46. Prawitz, D.: Natural Deduction, A Proof-Theoretical Study. Almqvist and Wiksell (1965)
47. Simpson, A.: The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D. thesis, University of Edinburgh (1994)
48. Sørensen, M.H., Urzyczyn, P.: Lectures on the Curry-Howard isomorphism. Elsevier (2006)
49. Terese: Term rewriting systems. Cambridge University Press (2003)
50. Tubella, A.A., Straßburger, L.: Introduction to Deep Inference (Aug 2019), <https://hal.inria.fr/hal-02390267>, lecture
51. Vakarelov, D.: Modal logics for knowledge representation systems. *Theor. Comput. Sci.* **90**, 433–456 (01 1991)

52. Valliappan, N., Ruch, F., Tom'e Cortinas, C.: Normalization for fitch-style modal calculi. Proc. ACM Program. Lang. **6**(ICFP), 772–798 (2022). <https://doi.org/10.1145/3547649>, <https://doi.org/10.1145/3547649>

A General Formulation of Simultaneous Inductive-Recursive Definitions in Type Theory

Peter Dybjer
Department of Computing Science
Chalmers University of Technology
S-412 96 Göteborg, Sweden
peterd@cs.chalmers.se

May 12, 1998

Abstract

The first example of a simultaneous inductive-recursive definition in intuitionistic type theory is Martin-Löf's universe à la Tarski. A set U_0 of codes for small sets is generated inductively at the same time as a function T_0 , which maps a code to the corresponding small set, is defined by recursion on the way the elements of U_0 are generated.

In this paper we argue that there is an underlying *general* notion of simultaneous inductive-recursive definition which is implicit in Martin-Löf's intuitionistic type theory. We extend previously given schematic formulations of inductive definitions in type theory to encompass a general notion of simultaneous induction-recursion. This enables us to give a unified treatment of several interesting constructions including various universe constructions by Palmgren, Griffor, Rathjen, and Setzer and a constructive version of Aczel's Frege structures. Consistency of a restricted version of the extension is shown by constructing a realisability model in the style of Allen.

1 Introduction

Martin-Löf type theory is a foundational framework in which induction is the principal notion. It is, to quote Martin-Löf [27, p73], “intended to be a full scale system for formalising intuitionistic mathematics as developed, for example, in the book by Bishop [10]”.

It is also a typed functional programming language not unlike ML [33] or Miranda [9]. A “set” in the theory is defined inductively by listing its constructors with their types in much the same way as one defines a recursive datatype in ML or Miranda. But whereas ML and Miranda are based on the simply typed λ -calculus, Martin-Löf type theory also has dependent types. The other key difference is that only well-founded elements of datatypes (sets) and terminating programs (total functions) may be constructed. To ensure well-foundedness datatype definitions have to satisfy a kind of “strict positivity” criterion. Moreover, to ensure termination recursive function definitions are restricted to “structural” recursion, that is, recursion on the way the elements of the domain of definition are inductively generated.

For simple types we can use the following notion of strict positivity. Let

$$\text{intro} : \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow P$$

($n \geq 0$) be a constructor for the datatype P . Then α_i either does not contain any occurrences of P or has the form $\xi_1 \rightarrow \cdots \rightarrow \xi_m \rightarrow P$ ($m \geq 0$), where ξ_j does not have any occurrences of P . It is also clear what the appropriate notion of structural recursion is for such a recursive datatype. Note that we allow *generalised inductive definitions* since a constructor can have functional arguments ($m > 0$). Therefore, the informal semantic notions of well-founded element and terminating function depend on each other.

The introduction of dependent types dramatically increases the expressiveness of the language. In particular, we can interpret intuitionistic predicate logic by following Curry, Howard, and de Bruijn and identify propositions and sets. In addition to the ordinary non-dependent set formers $0, 1, +, \times$, and \rightarrow , which can be used for interpreting the logical connectives $\perp, \top, \vee, \wedge$, and \supset , we now also have Σ and Π , the disjoint union and Cartesian product of a family of sets, which can be used for interpreting the quantifiers \exists and \forall .

However, the appropriate notion corresponding to “strict positivity” becomes more complex in the context of dependent types. Instead of formulating such a general condition for inductive definitions of sets Martin-Löf [31, 27, 28, 29] gave rules for a collection of specific set formers. However, this collection may be extended when there is a need for it provided the informal semantic principles of the theory are respected.

The possibility of formulating a general schema was however mentioned in Martin-Löf 1972 [31]:

The type N is just the prime example of a type introduced by an *ordinary inductive definition*. However, it seems preferable to treat this special case rather than to give a necessarily much more complicated general formulation which would include $(\Sigma \in A)B(x)$, $A + B$, N_n and N as special cases. See Martin-Löf 1971 [26] for a general formulation of inductive definitions in the language of ordinary first order predicate logic.

The first such general schema was formulated by Backhouse [7] and covered the case of inductively defined sets (possibly depending on parameters). This schema was generalised to the case of inductively defined families of sets by Dybjer [19, 20]. Inductively defined families subsume inductively defined predicates, and this schema can be viewed as the type-theoretic generalisation of the natural deduction schema for inductively defined predicates in predicate logic given by Martin-Löf [26].

In this paper we introduce a further generalization of the schema in Dybjer [20]. It covers *simultaneous inductive-recursive definitions* including definitions of a variety of universes which were not accounted for by the old schema. It also gives rise to other interesting notions including a constructive version of Aczel’s Frege structures.

Universes in type theory are analogous to Grothendieck universes in set theory: they are sets of “small” sets and can be used for example for the formalisation of constructive category theory. Another interesting application of universes (here in conjunction with generalised inductive definitions) is Aczel’s universe of iterative sets, in which a constructive version of Zermelo-Fraenkel set theory **CZF** can be interpreted [3]. But in the standard formulations of type theory universes are needed also for the more basic purpose of defining families of sets by structural recursion. For example, the predicate Z (as used

in the type-theoretic proof that $0 \neq s(n)$ [29, 43]) with the recursion equations

$$\begin{aligned} Z(0) &= \top, \\ Z(s(n)) &= \perp \end{aligned}$$

is defined in terms of universes and the rule of N -elimination.

Martin-Löf [27] introduced an infinite tower of universes $U_0 : U_1 : U_2 : \dots$. These were formulated “à la Russell” [29], which means that there is no syntactic distinction between a small set considered as an element of a universe $A : U_i$ and considered as a set A . In contrast, the formulation “à la Tarski” [29] maintains such a distinction: $a : U_i$ is a *code* for the set $T_i(a)$. A universe à la Tarski should therefore be understood as a pair (U_i, T_i) consisting of a set U_i of codes and a decoding function T_i .

Further universes were introduced by Palmgren [35]. Firstly, he defined a *universe operator*, that is, an operator on families of sets which when applied to a universe (U_i, T_i) returns the next universe (U_{i+1}, T_{i+1}) . In this way the external sequence of universes à la Tarski $(U_0, T_0), (U_1, T_1), (U_2, T_2), \dots$ was internalised. Secondly, Palmgren introduced a *super-universe* closed under the universe operator as well as under all the usual set formers in type theory. Recently, even larger universes and universe operators have been proposed by Rathjen, Griffor, and Palmgren [39], Palmgren [36], and Setzer [42].

Martin-Löf type theory with generalised inductive definitions and universes have great proof-theoretic strength, see Griffor and Rathjen [23], Setzer [41], and Rathjen, Griffor, and Palmgren [39].

We conclude this introduction with a few words about the notation. We employ the “logical framework” formulation of Martin-Löf type theory [30, 34]. The core of this theory is a typed $\lambda\beta\eta$ -calculus with dependent types. There is a base type *set*, the type of sets, and for each object $A : \text{set}$, there is the type $\text{El}(A)$ (often written just A) of the elements of A . We write $(x : \alpha)\beta$ for the type of functions which map an object $a : \alpha$ to an object $f(a) : \beta[a/x]$. If β does not depend on $x : \alpha$ we may write $(\alpha)\beta$ (rather than $\alpha \rightarrow \beta$) instead of $(x : \alpha)\beta$. Abstraction is written $(x)b$ and application $f(a)$, rather than the usual notation $\lambda x.b$ and $f a$ from λ -calculus. We also write $(x_1, \dots, x_n)b = (x_1) \cdots (x_n)b$ for multiple abstraction, and $f(a_1, \dots, a_n) = f(a_1) \cdots (a_n)$ for multiple application.

In this type system $(\text{set})(\text{set})\text{set}$ is for example the type of binary logical connectives; $(A : \text{set})(B : (A)\text{set})\text{set}$ is the type of quantifiers; and $(N)\text{set}$ is the type of unary predicates on N and also the type of N -indexed families of sets.

The formulation of the schema for simultaneous induction-recursion in section 3 will use two auxiliary notions. Firstly, we will write $(a :: \alpha)\beta$ as an abbreviation of $(a_1 : \alpha_1) \cdots (a_n : \alpha_n)\beta$ and call α a sequence of types. (This abbreviation could be avoided by adding Σ -types to the λ -calculus with dependent types.) Secondly, we will say that a type is *small* if it contains no occurrences of *set*. (Small types are called s-types in [19].) Small types are almost like sets. But the logical framework formulation we use maintains a distinction between the set $\Pi(A, B)$ and the small type $(x : A)B(x)$. Since we want our schema to cover the rules for Π as well we need to refer to small types when formulating the requirements on the rules.

Plan of the paper. In section 2 we introduce simultaneous induction-recursion by two examples: the first universe à la Tarski and the fresh-lists (lists where all elements are distinct). In section 3 we give the general schema for simultaneous inductive-recursive definitions in type theory. We also show how to recover the first universe à la Tarski and the fresh-lists by instantiating the schema. In section 4 we discuss further universe constructions which are instances of the schema. In section 5 we show how the construction of Frege structures in type theory is yet another instance of the schema. In section 6 we build a classical Frege structure model of a restricted version of the schema using monotone inductive definitions. In section 7 we conclude.

2 Two examples of simultaneous induction-recursion

The prototypical example of simultaneous induction-recursion is Martin-Löf’s definition of the first universe à la Tarski [29]. It consists of the simultaneous *inductive* definition of the set U_0 of codes for small sets and the *recursive* definition of the decoding function $T_0 : (U_0)\text{set}$. U_0 has one introduction rule (and one equality rule) for each set former which is reflected in the universe. We here give two examples:

Π -formation is reflected by the following rule of U_0 -introduction:

$$\pi_0 : (u : U_0)(u' : (x : T_0(u))U_0)U_0,$$

and we have the following equality rule:

$$T_0(\pi_0(u, u')) = \Pi(T_0(u), (x)T_0(u'(x))).$$

Eq -formation is reflected by the following rule of U_0 -introduction:

$$eq_0 : (u : U_0)(b, b' : T_0(u))U_0,$$

and we have the following equality rule:

$$T_0(eq_0(u, b, b')) = Eq(T_0(u), b, b').$$

This definition does not fit the schema for strictly positive inductive definitions in Dybjer [19], since T_0 appears (even negatively) in the introduction rules for U_0 . In spite of this one can justify that it is a *predicative* definition in Martin-Löf's sense [27]. Allen [5] has suggested calling this kind of definition *half-positive*.

For example, the rules for π_0 stipulate the following way of constructing new elements of U_0 . At a certain stage we may have constructed an element u . Since T_0 is defined by U_0 -recursion, we can compute the set $T_0(u)$. Hence we can construct a function u' with domain $T_0(u)$ and range (the presently constructed elements of) U_0 . Hence we can construct an element $\pi_0(u, u')$. Moreover, the computation of $T_0(\pi_0(u, u'))$ will terminate since we already know that the computation of $T_0(u)$ and of $T_0(u'(x))$ for $x : T_0(u)$ will terminate and that the $T_0(u'(x))$'s will only be called a finite number of times.

A more down-to-earth example (due to Catarina Coquand) is the following inductive definition of the set of lists $Dlist$ with distinct elements and the simultaneous definition of the freshness relation $Fresh$.

Let A be the set from which the elements of the list are drawn, and let $a \# b$ mean that a and b are different elements of A , for some difference relation $\# : (A)(A)set$. We have the following formation rules

$$\begin{aligned} Dlist &: set, \\ Fresh &: (Dlist)(A)set \end{aligned}$$

and introduction rules

$$\begin{aligned} nil &: Dlist, \\ cons &: (b : A)(u : Dlist)(b' : Fresh(u, b))Dlist. \end{aligned}$$

$Fresh$ has the equality rules

$$\begin{aligned} Fresh(nil, a) &= \top, \\ Fresh(cons(b, u, b'), a) &= b \# a \wedge Fresh(u, a). \end{aligned}$$

Note that $cons$ has an extra argument b' , which is a proof that the new element b is fresh with respect to u . Hence the introduction rules for $Dlist$ refer to $Fresh$. But as for the case of the first universe we shall argue that it is a good predicative definition.

For example, the rules for $cons$ stipulate the following way of constructing new elements of $Dlist$. At a certain stage we may have constructed $u : Dlist$. Since $Fresh$ is defined by $Dlist$ -recursion, we already know what it means for an element $b : A$ to be fresh with respect to u , that is, we know what a proof $b' : Fresh(u, b)$ is. Hence it makes sense to construct an element $cons(b, u, b')$. Moreover, we can define $Fresh(cons(b, u, b'))$ in terms of the already constructed proposition $Fresh(u)$.

There are of course a number of alternative ways to define $Dlist$ and $Fresh$ using ordinary inductive definitions, but the inductive-recursive one seems natural and may be preferred for some purposes.

In both examples we simultaneously build a function and its domain of definition. This intuition can be captured using a classical notion of monotone inductive definition and thus yield a consistency proof. This is the idea behind the realisability model in section 5.

A set former in Martin-Löf type theory is specified by its formation and introduction rules. Moreover, in the standard formulation [28, 29] it has an elimination rule which expresses a general principle of definition of a function by structural recursion. A particular function defined by structural recursion (for example the addition function on natural numbers) is then obtained by instantiating the elimination rule in question. Its recursion equations are then derived by instantiating the equality rules. But it is also possible to formulate definition by structural recursion by an external schema (so that for example the addition function is obtained as an instance) as in Martin-Löf [27], Coquand [14], and Dybjer [19].

When considering simultaneous inductive-recursive definitions it is essential to adopt the latter approach (using an external schema). The reason is that the elimination rule expresses only definition by structural recursion on a previously (and not simultaneously) defined set. For example, the definition of T_0 is an instance of a recursive schema. It cannot be formulated in terms of U_0 -elimination, because already the formulation of U_0 -elimination refers to T_0 .

Moreover, we want the schema for simultaneous induction-recursion to specialise to universe constructions. It is therefore essential that we do not require that the value of a function defined by recursion necessarily is an element of a set (as in the traditional elimination rules). Instead the value can be an object of an arbitrary type (as in the “large” elimination rules [44, 45]. For example, the value may be a set (an object of the type *set*), so we have *recursively defined families of sets*.

Note that we have reversed the priority of the following concepts as compared to the standard formulations of Martin-Löf type theory [28, 29, 34]:

- *Elimination and equality rules* are special instances of the *recursive schemata*, whereas in the standard formulation the recursive schemata are derived from the elimination and equality rules.
- *Universes* are special kinds of simultaneous inductive-recursive definitions employing *set-valued recursion*, whereas in the standard formulation set-valued recursion is obtained from the elimination and equality rules in conjunction with universes.

3 Formalising the notion of a simultaneous inductive-recursive definition

We use the schematic style of Martin-Löf’s intuitionistic theory of iterated inductive definitions in predicate logic [26] for presenting the theory of simultaneous inductive-recursive definitions in type theory. As already mentioned the present schema is a generalisation of the schema in Dybjer [19, 20]. The main difference is that the extended schema for an introduction rule may refer to a function defined simultaneously by recursion. To highlight the similarity between the extended schema and the old schema we use the same notation here as in Dybjer [20].

The present description could form the basis for an implementation in the same way as the old schema [20] is the basis of Giménez’ [22] implementation of inductive definitions in a proof editor for Martin-Löf type theory, and as Coquand and Paulin’s formulation of inductive types in the calculus of constructions [16, 38] is the basis for the Coq-system [18].

To illustrate the schema, we show how the rules for the first universe and the fresh-lists can be derived by instantiation. Later sections contain further examples. It might be helpful to study these examples before studying the general formulation given in this section.

I present the case with one inductive and one recursive definition. Clearly, the schema can be generalised to the case with several simultaneous inductive and several (possibly zero) recursive definitions, but we will not spell out the details.

To begin with (in 3.1-3.4) we assume that there are no parameters. In 3.5 it is shown how to extend the schema to simultaneous inductive-recursive definitions with parameters.

A definition is always relative to a theory containing the rules for previously defined concepts. Thus the requirements on the different parts of the definitions ($\alpha, \psi, \beta, \xi, p, q$ below) are always judgements with respect to that theory.

3.1 Formation rules

Schema. Let α be a sequence of small types. A simultaneous inductive-recursive definition of an α -

indexed family of sets P and an α -indexed family of functions f defined by P -recursion has formation rules of the form

$$\begin{aligned} P &: (a :: \alpha) \text{set}, \\ f &: (a :: \alpha)(c : P(a))\psi[a]. \end{aligned}$$

Here we also require that $\psi[a]$ is a type under the assumptions $a :: \alpha$.

Examples. The formation rules for the first universe

$$\begin{aligned} U_0 &: \text{set}, \\ T_0 &: (c : U_0)\text{set} \end{aligned}$$

are obtained by letting P be U_0 , f be T_0 , α be the empty sequence, and $\psi[c]$ be *set*.

The formation rules for fresh-lists

$$\begin{aligned} Dlist &: \text{set}, \\ Fresh &: (c : Dlist)(a : A)\text{set} \end{aligned}$$

are obtained by letting P be $Dlist$, f be $Fresh$, α be empty and $\psi[c]$ be $(a : A)\text{set}$ (the type of predicates on A).

For an example where α is a non-empty sequence, see section 5 on Frege structures, where the properties of internal propositionality and internal truth are defined.

3.2 Introduction rules

Schema. A premise of an introduction rule is either *non-recursive* or *recursive*.

- A non-recursive premise has the form

$$b : \beta,$$

where β is a small type depending on the previous premises of the rule (see below).

- A recursive premise has the form

$$u : (x :: \xi)P(p[x]),$$

where ξ is a sequence of small types, and $p[x] :: \alpha$ assuming $x :: \xi$ and the previous premises of the rule (see below). If ξ is empty the premise is called *ordinary* and otherwise *generalised* (as in ordinary and generalised induction).

The type of the conclusion of the introduction rule has the form

$$P(q),$$

where $q :: \alpha$ depending on the premises of the rule (see below).

We shall now spell out the typing criteria for β in the schema above. (The criteria for ξ , p , and q are analogous.) These criteria may seem complex at a first glance, but are nothing but what results from spelling out the obvious possible dependencies that can occur.

We write $\beta = \beta[\dots, b', \dots, u', \dots]$, etc., to explicitly indicate the dependence on typical previous non-recursive $b' : \beta'$ and recursive $u' : (x' :: \xi')P(p'[x'])$ premises. (Non-recursive and recursive premises may appear in any order). The dependence on a previous recursive premise can occur only through an application of the simultaneously defined function f . Formally, this means that we require that

$$\beta[\dots, b', \dots, u', \dots] = \hat{\beta}[\dots, b', \dots, (x')f(p'[x'], u'(x')), \dots],$$

where $\hat{\beta}[\dots, b', \dots, v', \dots]$ is a small type in the context $(\dots, b' : \beta', \dots, v' : (x' :: \xi')\psi[p'[x']], \dots)$. Note that this context is obtained from the context of β by replacing each recursive premise of the form $u' : (x' :: \xi')P(p'[x'])$ by $v' : (x' :: \xi')\psi[p'[x']]$.

In the sequel we will write

$$\text{intro} : \dots (b : \beta) \dots (u : (x :: \xi)P(p[x])) \dots P(q)$$

for the general form of an introduction rule. It indicates that a typical constructor *intro* may have non-recursive premises (arguments) $b : \beta$ and recursive premises $u : (x :: \xi)P(p[x])$. There may be zero or more premises of either kind and they may appear in arbitrary order.

If we remove the possibility that β, ξ, p , and q depend on previous *recursive* premises, then we essentially recover the schema in Dybjer [20], because then f cannot appear in the introduction rules for P . Moreover, since a non-recursive premise then cannot depend on a recursive one, we can without loss of generality assume that all non-recursive premises precede the recursive premises.

Examples. The introduction rule for U_0 which reflects Π -formation

$$\pi_0 : (u : U_0)(u' : (x : T_0(u))U_0)U_0$$

is obtained in the following way. The first premise $u : U_0$ is recursive and ordinary, that is, ξ in the schema is empty. The second premise $u' : (x : T_0(u))$ is recursive and generalised and depends on the first recursive premise. It is obtained from the schema by letting $\xi[u] = \hat{\xi}[T_0(u)] = T_0(u)$, that is, $\hat{\xi}[v] = v$.

The introduction rule reflecting Eq -formation is

$$eq_0 : (u : U_0)(b, b' : T_0(u))U_0.$$

The first premise $u : U_0$ is recursive and ordinary. The second and and third premises $b, b' : T_0(u)$ are non-recursive and depend on the first recursive premise. They are obtained by letting $\beta[u] = \hat{\beta}[T_0(u)] = T_0(u)$, that is, $\hat{\beta}[v] = v$.

The second introduction rule for $Dlist$ is

$$\text{cons} : (b : A)(u : Dlist)(b' : \text{Fresh}(u, b))Dlist.$$

It has a first non-recursive premise with $\beta = A$. The second premise is recursive and ordinary (ξ is empty). The third premise is non-recursive with $\beta[b, u] = \hat{\beta}[b, \text{Fresh}(u)] = \text{Fresh}(u, b)$ so $\hat{\beta}[b, v] = v(b)$.

3.3 Equality rules for the simultaneously defined function

Schema. Let *intro* be a constructor and let as above $b : \beta$ and $u : (x :: \xi)P(p[x])$ be typical non-recursive and recursive premises respectively of the corresponding introduction rule. The form of the equality rule for f and *intro* is:

$$f(q, \text{intro}(\dots, b, \dots, u, \dots)) = e(\dots, b, \dots, (x)f(p[x], u(x)), \dots) : \psi[q]$$

in the context

$$(\dots, b : \beta, \dots, u : (x :: \xi)P(p[x]), \dots)$$

where

$$e(\dots, b, \dots, v, \dots) : \psi[q]$$

in the context

$$(\dots, b : \beta, \dots, v : (x :: \xi)\psi[p[x]], \dots).$$

Examples. The equality rules for T_0

$$\begin{aligned} T_0(\pi_0(u, u')) &= \Pi(T_0(u), (x)T_0(u'(x))), \\ T_0(eq_0(u, b, b')) &= Eq(T_0(u), b, b') \end{aligned}$$

are obtained by letting $e(v, v') = \Pi(v, v')$ and $e(v, b, b') = Eq(v, b, b')$ respectively.

The equality rules for Fresh

$$\begin{aligned} \text{Fresh}(\text{nil}) &= (a)\top, \\ \text{Fresh}(\text{cons}(b, u, b')) &= (a)(b \# a \wedge \text{Fresh}(u, a)) \end{aligned}$$

are obtained by letting $e = (a)\top$ and $e(b, v, b') = (a)(b \# a \wedge v(a))$ respectively.

3.4 A generalisation of universe elimination

Schema. Universe elimination (as described in Nordström, Petersson, and Smith [34]) expresses definition by U_0 -recursion *after* U_0 and T_0 are defined. Here we express the corresponding notion schematically. We also show that ordinary universe elimination is a special case of this schema.

In general, after the simultaneous inductive-recursive definition of P and f has been completed, we may define a new function

$$f' : (a :: \alpha)(c : P(a))\psi'[a, c],$$

by P -recursion. Here we require that $\psi'[a, c]$ is a type in the context $(a :: \alpha, c : P(a))$.

The equality rule has the form

$$f'(q, \text{intro}(\dots, b, \dots, u, \dots)) = e'(\dots, b, \dots, u, (x)f'(p[x], u(x)), \dots)$$

in the context

$$(\dots, b : \beta, \dots, u : (x :: \xi)P(p[x]), \dots)$$

where

$$e'(\dots, b, \dots, u, v, \dots) : \psi'[q, \text{intro}(\dots, b, \dots, u, \dots)]$$

in the context

$$(\dots, b : \beta, \dots, u : (x :: \xi)P(p[x]), v : (x :: \xi)\psi'[p[x], u(x)], \dots)$$

Note that the criteria are identical for a simultaneously defined function f and a function f' defined afterwards, except that the target type ψ' of f' (in contrast to ψ of f) may depend on c as well as on a , and that the RHS of a recursion equation e' for f' (in contrast the RHS of a recursion equation e for f) may depend on u as well as on v . This is simply because these new dependencies can occur only after P has been defined.

Examples. Ordinary U_0 -elimination

$$\begin{aligned} U_0\text{rec} &: (C : (U_0)\text{set}) \\ &(e : (u : U_0)(v : C(u))(u' : (x : T_0(u))U_0)(v' : (x : T_0(u))C(u'(x)))C(\pi_0(u, u'))) \\ &(e' : (u : U_0)(v : C(u))(b, b' : T(u))C(eq_0(u, b, b'))) \\ &\vdots \\ &(c : U_0)C(c) \end{aligned}$$

with

$$\begin{aligned} U_0\text{rec}(C, e, e', \pi_0(u, u')) &= e(u, U_0\text{rec}(C, e, e', u), u', (x)U_0\text{rec}(C, e, e', u'(x))), \\ U_0\text{rec}(C, e, e', eq_0(u, b, b')) &= e'(u, U_0\text{rec}(C, e, e', u), b, b'), \\ &\vdots \end{aligned}$$

is obtained from the general schema by letting C, e, e', \dots be parameters (see section 3.5) and $\psi'[c] = C(c)$.

Another example is the function

$$\text{length} : (c : \text{Dlist})N$$

which computes the length of a Dlist by Dlist -recursion. The typing rule of length is obtained by letting $\psi'[c] = N$ in the general schema. It has the equality rules

$$\begin{aligned} \text{length}(\text{nil}) &= 0, \\ \text{length}(\text{cons}(b, u, b')) &= s(\text{length}(u)). \end{aligned}$$

These are obtained by letting $e' = 0$ and $e'(b, u, v, b') = s(v)$ respectively in the general schema.

3.5 Parameters

The simultaneous inductive-recursive definition of $Dlist$ and $Fresh$ depends on the parameters $A : set$ and $\# : (A)(A)set$. If we make these dependencies explicit in the formation rules for $Dlist$ and $Fresh$ they become

$$\begin{aligned} Dlist &: (A : set)(\# : (A)(A)set)set, \\ Fresh &: (A : set)(\# : (A)(A)set)(c : Dlist(A, \#))(a : A)set. \end{aligned}$$

We also need to modify the introduction rules accordingly:

$$\begin{aligned} nil &: (A : set)(\# : (A)(A)set)Dlist(A, \#), \\ cons &: (A : set)(\# : (A)(A)set)(b : A)(u : Dlist(A, \#))(b' : Fresh(A, \#, u, b))Dlist(A, \#). \end{aligned}$$

We can extend the schema to account for parameters in general. The rule is simple: there can be zero or more parameters and they can have arbitrary types. The modified schema for formation rules is therefore

$$\begin{aligned} P &: (A :: \sigma)(a :: \alpha[A])set, \\ f &: (A :: \sigma)(a :: \alpha[A])(c : P(A, a))\psi[A, a]. \end{aligned}$$

where σ is an arbitrary sequence of types, where $\alpha[A]$ is a sequence of small types under the assumptions $A :: \sigma$ and $\psi[A, a]$ is a type under the assumptions $A :: \sigma, a :: \alpha$. The modified schema for an introduction rule is

$$intro : (A :: \sigma) \cdots (b : \beta[A]) \cdots (u : (x :: \xi[A])P(A, p[A, x])) \cdots P(A, q)$$

where σ is the same as for the formation rules, and where the requirements on β, ξ, p , and q depend on $A :: \sigma$ as well.

A recursive definition of a function f' (defined by recursion on P after the simultaneous inductive-recursive definition of P and f) may also depend on parameters. This was used above when showing that ordinary universe elimination is an instance of the schema in section 2.4. Again, there may be zero or more parameters, and these parameters may have arbitrary types. Note however that the parameters of f' need not coincide with the parameters for the inductive-recursive definition of P and f .

4 Universe hierarchies and super-universes

In this section we review the universe hierarchies and super-universes of Palmgren [35]. Palmgren presented these constructions in a one-off fashion with informal motivations for the rules. Here we use our notion of a simultaneous inductive-recursive definition to give a unified formal treatment of all these constructions: as for the rules for U_0 we can recover them by instantiating the appropriate part of the general schema above.

4.1 External universe hierarchies

The *second universe* has formation rules

$$\begin{aligned} U_1 &: set, \\ T_1 &: (U_1)set, \end{aligned}$$

and analogous rules for the constructors π_1 and eq_1 to those for π_0 and eq_0 . There is also an introduction and equality rule reflecting U_0 -formation:

$$\begin{aligned} u_{01} &: U_1, \\ T_1(u_{01}) &= U_0. \end{aligned}$$

We also wish to reflect T_0 as a function into the second universe:

$$t_{01} : (U_0)U_1,$$

and therefore we let t_{01} be a constructor for U_1 . (Palmgren also mentions the possibility of defining t_{01} by recursion on U_0 .) The equality rule for T_1 is:

$$T_1(t_{01}(b)) = T_0(b).$$

We can continue in an analogous way and define U_2 and T_2 , U_3 and T_3 , etc. and thus get an external universe hierarchy.

4.2 An internal universe hierarchy and a super-universe

The construction of U_{n+1} and T_{n+1} from U_n and T_n can be internalised. We give a simultaneous inductive-recursive definition of the set formers

$$\begin{aligned} \text{Nextu} &: (U : \text{set})(T : (U) \text{set}) \text{set}, \\ \text{Nextt} &: (U : \text{set})(T : (U) \text{set})(\text{Nextu}(U, T)) \text{set}, \end{aligned}$$

and let $U_{n+1} = \text{Nextu}(U_n, T_n)$ and $T_{n+1} = \text{Nextt}(U_n, T_n)$.

U and T are parameters of this definition. For simplicity, we suppress these parameters in the rules for Nextu and Nextt and write

$$\begin{aligned} \text{Nextu} &: \text{set}, \\ \text{Nextt} &: (\text{Nextu}) \text{set}. \end{aligned}$$

Introduction and equality rules correspond to those of the first universe, but we also need to reflect the code set U and the decoding function T :

$$*: \text{Nextu},$$

$$\text{Nextt}(*) = U,$$

$$t : (b : U) \text{Nextu},$$

$$\text{Nextt}(t(b)) = T(b).$$

A *super-universe* is obtained by also reflecting the next-universe construction inside a set U_∞ . The formation rules are

$$\begin{aligned} U_\infty &: \text{set}, \\ T_\infty &: (U_\infty) \text{set}. \end{aligned}$$

Introduction and equality rules correspond to those for the first universe, but we also need to reflect the first universe U_0 and the next-universe construction Nextu

$$u_0 : U_\infty,$$

$$T_\infty(u_0) = U_0,$$

$$\text{nextu} : (u : U_\infty)(u' : (T_\infty(u))U_\infty)U_\infty,$$

$$T_\infty(\text{nextu}(u, u')) = \text{Nextu}(T_\infty(u), (x)T_\infty(u'(x))).$$

It is straightforward to check that also these rules follow the general schema.

4.3 A parameterised super-universe

The construction of the super-universe U_∞ can be generalised. Instead of starting with the next-universe operator, we can start with an arbitrary operator given by a pair

$$\begin{aligned} F &: (U : \text{set})(T : (U \text{ set})\text{set}, \\ G &: (U : \text{set})(T : (U \text{ set}))(F(U, T))\text{set}, \end{aligned}$$

which maps a family of sets (U, T) into another family of sets $(F(U, T), (x)G(U, T, x))$. Then we can construct a super-universe (U_P, T_P) closed under this operator by a simultaneous inductive-recursive definition. The definition of (U_P, T_P) is analogous to the definition of (U_∞, T_∞) ; the only difference is that we replace Next_u by the parameter F . If we make the dependence on the parameters F and G explicit in the types of U_P and T_P we get

$$\begin{aligned} U_P &: (F : (U : \text{set})(T : (U \text{ set})\text{set}) \\ &\quad (G : (U : \text{set})(T : (U \text{ set}))(F(U, T))\text{set}) \\ &\quad \text{set}, \\ T_P &: (F : (U : \text{set})(T : (U \text{ set})\text{set}) \\ &\quad (G : (U : \text{set})(T : (U \text{ set}))(F(U, T))\text{set}) \\ &\quad (c : U_P(F, G)) \\ &\quad \text{set}. \end{aligned}$$

I learned about this generalization and its connection to Mahlo cardinals in set theory from Anton Setzer. For a complete account and more discussion, see Setzer [42]. (The reader should note that there are some differences between Setzer's formalisation and the one suggested here.)

Several more examples of simultaneous inductive-recursive definitions of large universes and universe operators can be found in the recent papers by Palmgren [36] and Rathjen, Griffor, and Palmgren [39].

5 Frege structures

The notion of a *Frege structure* was introduced by Peter Aczel [4]. One purpose was to provide an appropriate setting for λ -calculus (or abstract realisability) interpretations of Martin-Löf type theory. Another was to provide a model for a foundational framework where the notions of “proposition” and “truth” are primitive.

We shall here show how to construct a Frege structure in Martin-Löf type theory by using a simultaneous inductive-recursive definition. Thus we show a way to reduce this foundational framework to the foundational framework of type theory. This type-theoretic construction can be contrasted to Aczel's construction of Frege structures in classical set theory.

The notion of a Frege structure is an enrichment of the notion of a *λ -structure*. This is essentially the same as a *λ -model* in Barendregt [8] but is expressed in terms of an *explicitly closed family* \mathcal{F} . Here \mathcal{F}_0 is the set of *objects* of the Frege structure and \mathcal{F}_n is a set of n -ary functions on \mathcal{F}_0 , which is used for interpreting expressions with at most n free variables. A Frege structure comes with projection functions $\pi_n^i : \mathcal{F}_n \rightarrow \mathcal{F}_0$ and \mathcal{F} -functionals $\lambda_n : \mathcal{F}_{n+1} \rightarrow \mathcal{F}_n$ and $\text{App}_n : \mathcal{F}_n \times \mathcal{F}_n \rightarrow \mathcal{F}_n$ satisfying appropriate equations.

The objects of a Frege structure are used both for encoding propositions and other mathematical objects. Given a λ -structure we can encode the logical constants: a binary connective can be encoded as a binary function on \mathcal{F}_0 , and a quantifier can be encoded as a function from \mathcal{F}_1 to \mathcal{F}_0 .

A Frege structure is a λ -structure with encodings of logical constants together with a set of objects called *propositions* and a subset of these called *truths*. These collections have to satisfy *logical schemata*, such as [4, page 37]:

Implication: If a is a proposition and the object b is a proposition provided that a true, then $a \supset b$ is a proposition, such that $a \supset b$ is true iff a is true implies b is true.

Universal quantification: If f is a propositional function in \mathcal{F}_1 , then $\forall x f(x)$ is a proposition, such that $\forall x f(x)$ is true iff $f(a)$ is true for all objects a .

Note that $a \supset b$ is a non-standard notion of implication, since b is required to be a proposition only when a is true.

A λ -structure with an encoding of the logical constants can be obtained by a standard construction in the λ -calculus [4]. To get a Frege structure we need to construct the collections of propositions and truths. The basic idea is to view the logical schemata as inductively defining these collections. But a direct interpretation as a positive inductive definition is not possible, since the notion of truth appears negatively in the logical schema for implication. Instead, Aczel [4] showed that one can interpret the logical schemata as an operator on pairs of sets of objects, which is monotone with respect to the following “conservative extension” ordering:

$$\langle \mathcal{P}, \mathcal{T} \rangle \leq \langle \mathcal{P}', \mathcal{T}' \rangle \text{ iff } \mathcal{P} \subseteq \mathcal{P}' \text{ and } \mathcal{T} = \mathcal{T}' \cap \mathcal{P}.$$

Intuitively, when constructing new propositions, the notion of truth on old propositions should remain the same. Since this ordering is complete it follows by a standard argument that the operator has a least fixed point when working in classical set theory.

However, Aczel also made the following remark [4, page 55] which has provided motivation for the present work:

Nevertheless I believe this result to be constructively valid. A rigorous elaboration of this point would require an explicit discussion of the role of inductive definitions in constructive mathematics. It will have to suffice here if I simply assert that the logical schemata form the clauses of an inductive definition that generate the propositions and simultaneously give conditions for their truth.

The standard construction of a λ -structure with set of objects \mathcal{F}_0 and an explicit equivalence relation on objects \sim_0 , together with an encoding of the logical constants can be carried out in type theory (see for example Hedberg [24] for a formal development of constructive domain theory inside type theory).

It remains to turn the logical schemata into a simultaneous inductive definition of the property $\mathcal{P} : (a : \mathcal{F}_0) \text{set}$ of propositionality and a recursive definition of the truth of a proposition $\mathcal{T} : (a : \mathcal{F}_0)(c : \mathcal{P}(a))\text{set}$. Note that these formation rules are obtained by instantiating the schema with $P = \mathcal{P}$, $f = \mathcal{T}$, α the singleton sequence \mathcal{F}_0 , and $\psi[a] = \text{set}$. It is essential that \mathcal{T} has a second argument, the proof that an object is a proposition, since it is defined by recursion on that proof.

We also show the introduction rules corresponding to the logical schemata for implication and universal quantification. We first extract from the λ -structure a function $ev : (\mathcal{F}_1)(\mathcal{F}_0)\mathcal{F}_0$ which applies a unary function to an object, and codes $\supset : (\mathcal{F}_0)(\mathcal{F}_0)\mathcal{F}_0$ and $\forall : (\mathcal{F}_1)\mathcal{F}_0$ for implication and universal quantification. The rules are:

$$\supset\text{-intro} : (a : \mathcal{F}_0)(p : \mathcal{P}(a))(b : \mathcal{F}_0)(q : (\mathcal{T}(a, p))\mathcal{P}(b))(c : \mathcal{F}_0)(z : c \sim_0 (a \supset b))\mathcal{P}(c),$$

$$\mathcal{T}(c, \supset\text{-intro}(a, p, b, q, c, z)) = \Pi t : \mathcal{T}(a, p).\mathcal{T}(b, q(t)),$$

$$\forall\text{-intro} : (f : \mathcal{F}_1)(p : (x : \mathcal{F}_0)\mathcal{P}(ev(f, x)))(c : \mathcal{F}_0)(z : c \sim_0 \forall(f))\mathcal{P}(c),$$

$$\mathcal{T}(c, \forall\text{-intro}(f, p, c, z)) = \Pi x : \mathcal{F}_0.\mathcal{T}(ev(f, x), p(x)).$$

Note that these definitions ensure that \sim_0 is preserved by \mathcal{P} and \mathcal{T} . This can be shown formally by using the analogue of universe elimination.

The main point is that the logical schemata directly can be interpreted as a basic kind of definition, specifically, as a simultaneous inductive-recursive definition which is an instance of our schema. We also note the similarity between this understanding and the intuitive reading of Aczel’s construction as a conservative extension ordering: since \mathcal{T} is a function, the truth of a proposition cannot change once it is defined.

We therefore argue that we have shown a way to make Aczel’s claim, that his result is constructively valid, precise. Essential is our use of explicit proof-objects and the fundamental difference between an inductive and a recursive definition in type theory.

6 Realisability model

To prove the consistency we shall construct a realisability model. We first show how to interpret an arbitrary instance of the schema in 3.1-3.4. This interpretation also applies to definitions with parameters as specified in 3.5 provided these parameters satisfies a particular positivity criterion, see section 6.4.4.

We follow Aczel [2, 4] and interpret types as collections of objects of a Frege structure. But instead of first building collections of propositions and truths as in Aczel's work, we shall directly construct the collections of sets $\mathcal{S}et$ and elements $\mathcal{E}l[A]$ (for each $A \in \mathcal{S}et$) in a manner similar to Allen [5]. These collections will be inductively defined using Aczel's rule sets [1].

Like Aczel, we use 'object' to refer to an element of a λ -structure and 'collection' to refer to a subset of the elements of a λ -structure. Furthermore, since substitution in the λ -calculus is interpreted as set-theoretic application in the λ -structure, we will use the notation $f[x]$ for set-theoretic application in general.

We begin by reviewing Aczel's rule sets in 6.1. Then we review the interpretation of the logical framework in 6.2. The interpretation of sets is overviewed in 6.3. In 6.4 we then give the interpretation of the general schema for inductive-recursive definitions. Finally, we give an example theory in 6.5 and show how to instantiate the schematic interpretation in 6.4 to give an interpretation of this example theory. It is probably better to study the example interpretation in 6.5 before reading the interpretation of the general schema in 6.4.

6.1 Rule sets

We use Aczel's [1] set-theoretic notion of rule set for defining these collections. It is defined as follows.

A *rule* on a base set V in Aczel's sense is a pair of sets $\langle u, v \rangle$, often written

$$\frac{u}{v},$$

such that $u \subseteq V$ and $v \in V$.

Let Φ be a set of rules on V . A set w is Φ -closed if

$$\frac{u}{v} \in \Phi \wedge u \subseteq w \supset v \in w.$$

There is a least Φ -closed set

$$\mathcal{I}(\Phi) = \bigcap \{w \subseteq V \mid w \text{ } \Phi\text{-closed}\},$$

the set inductively defined by Φ .

Each rule set Φ on V generates a monotone operator

$$\phi(X) = \{v \in V \mid \exists \frac{u}{v} \in \Phi. u \subseteq X\}$$

on $\mathcal{P}ow(V)$, such that $\mathcal{I}(\Phi)$ is the least fixed point of ϕ .

We use rule sets rather than monotone operators here, since they allow a more direct encoding of type-theoretic inductive definitions expressed in terms of introduction rules.

6.2 Interpretation of the logical framework

We first briefly review the interpretation of the logical framework in a λ -structure satisfying the β - and the η -rule.

The open terms of the logical framework are interpreted as follows. Abstraction and application are interpreted in terms of the \mathcal{F} -functionals λ_n and App_n in the λ -structure. Variables are interpreted as projections π_n^i . For reasons of presentation we shall in the sequel use the same notation for a term and its interpretation in the λ -structure: $(x)b$ for $\llbracket (x)b \rrbracket$, $c(a)$ for $\llbracket c(a) \rrbracket$ and x for $\llbracket x \rrbracket$.

Using this notation we interpret the judgements of type theory relative to an environment ρ as follows:

$$\begin{aligned} \llbracket \alpha \text{ type} \rrbracket_\rho &\quad \text{is} \quad \llbracket \alpha \rrbracket_\rho \subseteq \mathcal{F}_0, \\ \llbracket \alpha = \beta \rrbracket_\rho &\quad \text{is} \quad \llbracket \alpha \rrbracket_\rho = \llbracket \beta \rrbracket_\rho, \\ \llbracket a : \alpha \rrbracket_\rho &\quad \text{is} \quad a_\rho \in \llbracket \alpha \rrbracket_\rho, \\ \llbracket a = b : \alpha \rrbracket_\rho &\quad \text{is} \quad a = b. \end{aligned}$$

\mathcal{F}_0 is the set of objects of the λ -structure (as in the section on Frege structures) and a_ρ is the result of applying the denotation $a \in \mathcal{F}_n$ to the sequence of n objects given by ρ . If the judgement in question is made under the assumptions (in the context)

$$x_1 : \alpha_1, \dots, x_n : \alpha_n$$

then ρ assigns objects to variables as follows:

$$x_1 = a_1 \in \llbracket \alpha_1 \rrbracket_{\rho_0}, \dots, x_n = a_n \in \llbracket \alpha_n \rrbracket_{x_1=a_1, \dots, x_{n-1}=a_{n-1}},$$

where ρ_0 is the empty environment.

The interpretation of a function type is

$$\llbracket (x : \alpha) \beta \rrbracket_\rho = \{c \in \mathcal{F}_0 \mid \forall a \in \llbracket \alpha \rrbracket_\rho. c(a) \in \llbracket \beta \rrbracket_{\rho, x=a}\}.$$

One can verify that all rules of the logical framework hold under this interpretation.

6.3 Overview of the interpretation of inductive and recursive definitions

We shall now show how to interpret sets and functions introduced by (possibly simultaneous) inductive and recursive definitions.

We call a specific sequence of definitions a *theory*. Recall from section 3 that the correctness of a given simultaneous inductive-recursive definition is always given relative to the sequence of prior definitions. Also recall the following:

- An inductive definition is given by a sequence of typings of constants for a set constructor and its element constructors.
- A recursive definition is given by a typing of a function constant together with its recursion equations.
- A simultaneous inductive-recursive definition is given by a sequence of typings of constants for a set constructor, a function constant, and the element constructors together with the recursion equations for the function constant.

The interpretation of an arbitrary theory following the schema has two parts.

1. Each *constant* of the theory is interpreted as an object of the λ -structure in such a way that all *recursion equations* (equality rules) are satisfied. (Given an interpretation of the constants we can construct the interpretation of an arbitrary term and it is thus clear what it means that an equation is satisfied in the model.)
2. The *base types set* and $El(A)$ are then interpreted as inductively defined collections Set and $El[A]$. Set and $El[A]$ are built up in stages. Assuming that the theory contains m *inductive* definitions (counting also the inductive parts of the inductive-recursive definitions), we construct a sequence of interpretations

$$\emptyset = El_0 \subseteq \dots \subseteq El_m = El \subseteq \mathcal{F}_0 \times Pow(\mathcal{F}_0)$$

Each El_i is defined inductively. It will be the case that each El_i is a functional relation, and we define Set_i as the domain of El_i . Thus we get a sequence

$$\emptyset = Set_0 \subseteq \dots \subseteq Set_m = Set \subseteq \mathcal{F}_0$$

Therefore each type α which is definable at stage i in the theory can be interpreted as a collection $\llbracket \alpha \rrbracket_i$ by interpreting each occurrence of Set as Set_i and each occurrence of $El(A)$ as $El_i[\llbracket A \rrbracket_i]$.

These two parts correspond to the following parts in Aczel [4].

1. The formation of an *independent family of \mathcal{F} -functionals* for the logical constants, and the use of fixed points to interpret recursion equations.
2. The construction of the collections of *propositions* and *truths*.

6.4 Interpretation of the general schema for simultaneous inductive-recursive definitions

6.4.1 Interpretation of constants

We assume that no definition has *parameters*. The interpretation of these are discussed in 6.4.4.

First we form the *list of set constructors* with associated arities, and for each inductively defined set we form its *list of element constructors* with associated arities. For each of these lists we interpret the constructors as objects of the λ -structure in such a way that there is another object *case* satisfying the recursion equations for case analysis. These have the form

$$\text{case}(d_1, \dots, d_n, \text{intro}_i(a_1, \dots, a_p)) = d_i(a_1, \dots, a_p),$$

where intro_i by abuse of notation is an object interpreting a constructor intro_i of arity p on the list in question. It follows that (the interpretation of) constructors are injective,

$$\text{intro}_i(a_1, \dots, a_p) = \text{intro}_i(a'_1, \dots, a'_p) \supset a_1 = a'_1 \wedge \dots \wedge a_p = a'_p,$$

and non-overlapping,

$$i \neq i' \supset \text{intro}_i(a_1, \dots, a_p) \neq \text{intro}_{i'}(a'_1, \dots, a'_{p'}).$$

We call this interpretation an *independent family of objects* for a list of constructors.

Independent families of objects can always be found, but the definability of *case* forces us to use “recursive” rather than “iterative” encodings, see the discussion by Parigot [37] and Altenkirch [6].

Secondly, for each recursively defined function constant, we construct an element satisfying the recursion equations in question. This can be done in a standard way using fixed points and case analysis. This construction is also used by Aczel [4].

6.4.2 Interpretation of base types

Assume that we have constructed $\emptyset = \mathcal{E}l_0 \subseteq \dots \subseteq \mathcal{E}l_i$ and that the $i + 1$ -st definition is a simultaneous inductive definition of a family of sets

$$P : (a :: \alpha)\text{set},$$

and recursive function

$$f : (a :: \alpha)(c : P(a))\psi[a].$$

A typical constructor is

$$\text{intro} : \dots (b : \beta) \dots (u : (x :: \xi)P(p[x])) \dots P(q)$$

and a typical recursion equation (equality rule) is

$$f(q, \text{intro}(\dots, b, \dots, u, \dots)) = e(\dots, b, \dots, (x)f(p[x], u(x)), \dots).$$

(The case of a simple inductive definition or of a simple recursive definition are easily obtained as degenerate versions.)

$\mathcal{E}l_{i+1}$ is now defined by a rule set obtained by adding the rules

$$\{\frac{\emptyset}{\langle P(a), \mathcal{P}[a] \rangle} | a \in \llbracket \alpha \rrbracket_i\}$$

to the rule set for $\mathcal{E}l_i$. Here \mathcal{P} is the interpretation of P as a family of collections indexed by $a \in \llbracket \alpha \rrbracket_i$ defined by

$$\mathcal{P}[a] = \{c \in \mathcal{F}_0 | a \mathcal{R}_P c\},$$

where $\mathcal{R}_P \subseteq \llbracket \alpha \rrbracket_i \times \mathcal{F}_0$ is an auxiliary relation between indices and objects. This relation is inductively defined by a rule set which is the union of the rule sets corresponding to each of the constructors. For example, to *intro* we associate the rule set

$$\{\frac{\dots \cup \{\langle p[x], u(x) \rangle | x \in \llbracket \xi \rrbracket_i\} \cup \dots}{\langle q, \text{intro}(\dots, b, \dots, u, \dots) \rangle} | \dots, b \in \llbracket \beta \rrbracket_i, \dots, \forall x \in \llbracket \xi \rrbracket_i. u(x) \in \mathcal{F}_0, \dots\}$$

on $\llbracket \alpha \rrbracket_i \times \mathcal{F}_0$. This rule set is well-defined, since $\llbracket \alpha \rrbracket_i, \dots, \llbracket \beta \rrbracket_i, \dots, \llbracket \xi \rrbracket_i$ are all well-defined. (We have not spelled out their dependencies however.)

6.4.3 Verification of the rules

The main point is that $\mathcal{El}[A] = \mathcal{El}_i[A]$ for $A \in \text{Set}_i$, and hence for any small type α in the previous theory (with i inductively defined families of sets), we have $\llbracket \alpha \rrbracket = \llbracket \alpha \rrbracket_i$.

As before, the formation rule is interpreted as

$$a \in \llbracket \alpha \rrbracket \supset P(a) \in \text{Set}$$

which can be shown to hold by inspecting the rule set for \mathcal{El} .

The typical introduction rule is interpreted as

$$\cdots \supset b \in \llbracket \beta \rrbracket_i \supset \cdots \supset (\forall x \in \llbracket \xi \rrbracket_i. (p[x]) \mathcal{R}_P(u(x))) \supset \cdots \supset q \mathcal{R}_P(\text{intro}(\dots, b, \dots, u, \dots)),$$

which follows directly by inspecting the rule set for \mathcal{R}_P .

The typing rule for f is interpreted as

$$a \in \llbracket \alpha \rrbracket \supset a \mathcal{R}_P c \supset f(a, c) \in \llbracket \psi[a] \rrbracket$$

This is proved by induction on \mathcal{R}_P using the recursion equations for f . For example, we need to prove that

$$q \in \llbracket \alpha \rrbracket \supset q \mathcal{R}_P(\text{intro}(\dots, b, \dots, u, \dots)) \supset f(q, \text{intro}(\dots, b, \dots, u, \dots)) \in \llbracket \psi[q] \rrbracket$$

from the induction hypotheses

$$p[x] \in \llbracket \alpha \rrbracket \supset \cdots \supset (\forall x \in \llbracket \xi \rrbracket_i. p[x] \mathcal{R}_P u(x)) \supset \cdots \supset f(p[x], u(x)) \in \llbracket \psi[p[x]] \rrbracket$$

But this follows directly by using the recursion equations for f .

Finally, the interpretation of the constants in 6.4.1 was specifically constructed to ensure that the equality rules for f are satisfied.

6.4.4 The interpretation of parameters

So far we have discussed the interpretation of set formers without parameters. Recall from subsection 3.5 that a set former can be parameterised with respect to an arbitrary sequence of types. It is easy to extend the interpretation to account for parameters as long as these parameters contain no occurrences of *set* in the α -part of a $(x : \alpha)\beta$. Because then we see that \mathcal{El} can be given by a monotone inductive definition using rule sets.

For an example of the interpretation of a set former with parameters, the reader is referred to the interpretation of Π in subsection 6.5.

When the preliminary version of this paper was written I was not aware of any interesting examples with such negative occurrences. But recently I learned that there indeed are several examples of such definitions of sets which are interesting as type-theoretic analogues of large cardinals. An example is the parameterised super-universe in 4.3.

It is therefore an interesting open problem to construct a classical set-theoretic model of the entire schema for simultaneous induction-recursion. The belief that the entire schema is consistent relies at present on an informal semantic analysis of the rules which shows that only well-founded elements and terminating functions can be constructed under the schema.

6.4.5 Consistency

The interpretation of \perp is the empty set. Hence we have shown that any theory obtainable by successive instantiations of the schema (satisfying the restriction on parameters) is consistent relative to classical set theory.

One can also argue from an informal semantical analysis that there can be no element $a : \perp$. Because when a is evaluated it would terminate with a value of the form $\text{intro}(a_1, \dots, a_n)$ where intro is a constructor for \perp . But \perp has no constructor. This is called “simple-minded consistency” in Martin-Löf [29].

6.5 Interpretation of an example theory

We now show how to interpret an example theory. This theory is chosen to illustrate different aspects of the general situation and consists of the following parts: (i) the inductive definition of the set N of natural numbers; (ii) the recursive definition of the addition function add ; (iii) the inductive definition of the family $N'(n)$ of finite sets indexed by the number n of elements of the set; (iv) the inductive definition of the cartesian product $\Pi(A, B)$ of a family of sets; and (v) the simultaneous inductive-recursive definition of a universe reflecting N , N' , and Π .

The reader should check that the interpretation given here is indeed obtainable from the schematic interpretation given in 6.4.

6.5.1 The example theory

We list the rules of the theory. Note that we write $\text{El}(A)$ rather than just A (as before) for the type of elements of the set A . This is because the interpretation of El is the crucial part of the realisability model.

$$\begin{aligned}
N &: \text{set}, \\
0 &: \text{El}(N), \\
s &: (u : \text{El}(N))\text{El}(N), \\
\\
\text{add} &: (\text{El}(N))(\text{El}(N))\text{El}(N), \\
\text{add}(m, 0) &= m, \\
\text{add}(m, s(u)) &= s(\text{add}(m, u)), \\
\\
N' &: (\text{El}(N))\text{set}, \\
0' &: (b : \text{El}(N))\text{El}(N'(s(b))), \\
s' &: (b : \text{El}(N))(u : \text{El}(N'(b)))\text{El}(N'(s(b))), \\
\\
\Pi &: (A : \text{set})(B : (\text{El}(A))\text{set})\text{set}, \\
\lambda &: (A : \text{set})(B : (\text{El}(A))\text{set})(b : (x : \text{El}(A))\text{El}(B(x)))\text{El}(\Pi(A, B)), \\
\\
U_0 &: \text{set}, \\
T_0 &: (\text{El}(U_0))\text{set}, \\
n_0 &: \text{El}(U_0), \\
n'_0 &: (b : \text{El}(N))\text{El}(U_0), \\
\pi_0 &: (u : \text{El}(U_0))(u' : (x : \text{El}(T_0(u)))\text{El}(U_0))\text{El}(U_0), \\
T_0(n_0) &= N, \\
T_0(n'_0(b)) &= N'(b), \\
T_0(\pi_0(u, u')) &= \Pi(T_0(u), (x)T_0(u'(x))).
\end{aligned}$$

6.5.2 Interpretation of the constants

The *list of set constructors* with arities is $N - 0, N' - 1, \Pi - 2, U_0 - 0$, and we construct an independent family of objects interpreting them.

The *lists of element constructors* with arities are

- $0 - 0, s - 1$ for N .
- $0' - 1, s' - 2$ for N' .

- $\lambda - 1$ for Π . (The first two arguments of λ are parameters, which are dropped under the interpretation: $\llbracket \lambda(A, B, b) \rrbracket = \lambda_n[[b]]$.)
- $n_0 - 0, n'_0 - 1, \pi_0 - 2$ for U_0 ,

and for each of these we construct an independent family of objects interpreting them.

The recursively defined functions are add and T_0 . By definition of independent family of objects there are elements $case_N$ and $case_{U_0}$, such that

$$\begin{aligned} case_N(d_1, d_2, 0) &= d_1, \\ case_N(d_1, d_2, s(u)) &= d_2(u), \\ case_{U_0}(d_1, d_2, d_3, n_0) &= d_1, \\ case_{U_0}(d_1, d_2, d_3, n'_0(b)) &= d_2(b), \\ case_{U_0}(d_1, d_2, d_3, \pi_0(u, u')) &= d_3(u, u'). \end{aligned}$$

Hence we can interpret add and T by solving the fixed point equations

$$\begin{aligned} add(m, n) &= case_N(m, (x)s(add(m, x)), n), \\ T_0(c) &= case_{U_0}(N, (b)N'(b), (u, u')\Pi(T_0(u), (x)T_0(u'(x))), c). \end{aligned}$$

This completes the interpretation of the constants.

6.5.3 Interpretation of the base types

We construct a sequence of functional relations

$$\emptyset = \mathcal{E}l_0 \subseteq \mathcal{E}l_1 \subseteq \dots \subseteq \mathcal{E}l_4 = \mathcal{E}l \subseteq \mathcal{F}_0 \times \mathcal{P}ow(\mathcal{F}_0)$$

since there are 4 inductively defined sets.

Natural numbers. $\mathcal{E}l_1$ is defined by the singleton rule set

$$\{\frac{\emptyset}{\langle N, \mathcal{N} \rangle}\},$$

where \mathcal{N} is the collection inductively defined by the rule set

$$\{\frac{\emptyset}{0} \cup \{\frac{\{a\}}{s(a)} | a \in \mathcal{F}_0\}$$

on \mathcal{F}_0 .

The family of finite sets. $\mathcal{E}l_2$ is obtained by adding the rules

$$\{\frac{\emptyset}{\langle N'(n), \mathcal{N}'[n] \rangle} | n \in \mathcal{N}\}$$

to the rule set for $\mathcal{E}l_1$. Here

$$\mathcal{N}'[n] = \{i \in \mathcal{F}_0 | n \mathcal{R}_{N'} i\}$$

where $\mathcal{R}_{N'} \subseteq \mathcal{N} \times \mathcal{F}_0$ is an auxiliary relation between indices and objects inductively defined by the rule set

$$\{\frac{\emptyset}{\langle s(n), 0'(n) \rangle} | n \in \mathcal{N}\} \cup \{\frac{\{\langle n, i \rangle\}}{\langle s(n), s'(n, i) \rangle} | n \in \mathcal{N} \wedge i \in \mathcal{F}_0\}$$

on $\mathcal{N} \times \mathcal{F}_0$.

Cartesian product of a family of sets. $\mathcal{E}l_3$ is obtained by adding

$$\{\frac{\{\langle A, \mathcal{A} \rangle\} \cup \{\langle B(x), \mathcal{B}[x] \rangle | x \in \mathcal{A}\}}{\langle \Pi(A, B), \prod[\mathcal{A}, \mathcal{B}] \rangle} | A, B \in \mathcal{F}_0, \mathcal{A} \in \mathcal{P}ow(\mathcal{F}_0), \mathcal{B} \in \mathcal{A} \rightarrow \mathcal{P}ow(\mathcal{F}_0)\}$$

to the rule set for \mathcal{El}_2 . Here $\prod[\mathcal{A}, \mathcal{B}] \subseteq \mathcal{F}_0$ for $\mathcal{A} \subseteq \mathcal{F}_0$ and $\mathcal{B}[x] \subseteq \mathcal{F}_0$ for $x \in \mathcal{A}$ is the collection inductively defined by the rule set

$$\{\frac{\emptyset}{\lambda(b)} | \forall x \in \mathcal{A}. b(x) \in \mathcal{B}[x]\}$$

on \mathcal{F}_0 .

The first universe. \mathcal{El}_4 is inductively defined by the rule set obtained by adding

$$\{\frac{\emptyset}{\langle U_0, \mathcal{U}_0 \rangle}\}$$

to the rule set for \mathcal{El}_3 . Here \mathcal{U}_0 is the collection inductively defined by the rule set

$$\{\frac{\emptyset}{n_0}\} \cup \{\frac{\emptyset}{n'_0(n)} | n \in \mathcal{N}\} \cup \{\frac{\{a\} \cup \{b(x) | x \in \mathcal{El}_3[T_0(a)]\}}{\pi_0(a, b)} | a, b \in \mathcal{F}_0\}$$

on \mathcal{F}_0 . (Note that this definition refers to \mathcal{El}_3 and that it follows that $T_0(a) \in \mathcal{Set}_3$ for all $a \in U_0$.)

6.5.4 Verification of the rules

General rules. First we verify the rule *set type*, that is,

$$\mathcal{Set} \subseteq \mathcal{F}_0$$

which is immediate. Then both rules *El(A) type* if $A : \text{set}$ and $\mathcal{El}(A) = \mathcal{El}(A')$ if $A = A' : \text{set}$ hold, since it is easy to prove that \mathcal{El} is functional:

$$\mathcal{El} \in \mathcal{Set} \rightarrow \mathcal{P}ow(\mathcal{F}_0).$$

This proof relies on the interpretation of set constructors as an independent family of objects.

Natural numbers. N -formation is interpreted as

$$N \in \mathcal{Set},$$

which can be shown by inspecting the rule set for \mathcal{El} .

Since $\mathcal{El}[N] = \mathcal{N}$, the rules of N -introduction are interpreted as

$$0 \in \mathcal{N}$$

and

$$a \in \mathcal{N} \supset s(a) \in \mathcal{N}.$$

Both follow directly by inspecting the rule set defining \mathcal{N} .

Addition. The typing rule for *add* is interpreted as

$$m \in \mathcal{N} \supset n \in \mathcal{N} \supset add(m, n) \in \mathcal{N}.$$

This can be shown by induction on $n \in \mathcal{N}$ using the recursion equations for *add*.

The equality rules for *add* follow directly from the recursion equations. Moreover, we can check that the typings in the equality rules are satisfied.

The family of finite sets. N' -formation is interpreted as

$$n \in \mathcal{N} \supset N'(n) \in \mathcal{Set},$$

which can be shown by inspecting the rule set for \mathcal{El} .

Since $i \in \mathcal{El}[N'(n)]$ iff $n \mathcal{R}_{N'} i$, the rules of N' -introduction are interpreted as

$$n \in \mathcal{N} \supset (s(n)) \mathcal{R}_{N'} (0'(n))$$

and

$$n \in \mathcal{N} \supset n\mathcal{R}_{N'} i \supset (s(n))\mathcal{R}_{N'}(s'(n, i)).$$

Both follow directly by inspecting the rule set defining $\mathcal{R}_{N'}$.

Cartesian product of a family of sets. Π -formation is interpreted as

$$A \in \mathcal{S}et \supset (\forall x \in \mathcal{E}l[A].B(x) \in \mathcal{S}et) \supset \Pi(A, B) \in \mathcal{S}et.$$

which can be shown by inspecting the rule set for $\mathcal{E}l$.

Π -introduction is interpreted as

$$(\forall x \in \mathcal{A}.b(x) \in \mathcal{B}[x]) \supset \lambda(b) \in \prod[\mathcal{A}, \mathcal{B}]$$

which can be shown to hold by inspecting the rule set for $\prod[\mathcal{A}, \mathcal{B}]$.

The first universe. U_0 -formation is interpreted as

$$U_0 \in \mathcal{S}et,$$

which can be shown by inspecting the rule set for $\mathcal{E}l$.

Since $\mathcal{E}l[U_0] = \mathcal{U}_0$, T_0 -typing is interpreted as

$$a \in \mathcal{U}_0 \supset T_0(a) \in \mathcal{S}et.$$

This is proved by proving the stronger property that $a \in \mathcal{U}_0 \supset T_0(a) \in \mathcal{S}et_3$ by induction on \mathcal{U}_0 . For example, we need to show that $\pi_0(a, b) \in \mathcal{U}_0 \supset T_0(\pi_0(a, b)) \in \mathcal{S}et_3$ from $a \in \mathcal{U}_0 \supset T_0(a) \in \mathcal{S}et_3$ and $b(x) \in \mathcal{U}_0 \supset T_0(b(x)) \in \mathcal{S}et_3$ for all $x \in \mathcal{E}l_3[T_0(a)]$. But this follows from the facts that $T_0(\pi_0(a, b)) = \Pi(T_0(a), (x)T_0(b(x)))$ and that $\mathcal{S}et_3$ is closed under Π .

The rules of U_0 -introduction are interpreted as

$$n_0 \in \mathcal{U}_0,$$

$$n \in \mathcal{N} \supset N'(n) \in \mathcal{U}_0,$$

$$a \in \mathcal{U}_0 \supset (\forall x \in \mathcal{E}l[T_0(a)].b(x) \in \mathcal{U}_0) \supset \pi_0(a, b) \in \mathcal{U}_0.$$

They follow directly by inspecting the rule set defining \mathcal{U}_0 using that $\mathcal{E}l[T_0(a)] = \mathcal{E}l_3[T_0(a)]$.

The equality rules for T_0 follow directly, since $T_0 \in \mathcal{F}_0$ was constructed to satisfy the corresponding untyped equalities. Moreover, we easily check that the typings in the equality rules are satisfied.

7 Concluding remarks

The formulation of simultaneous inductive–recursive definitions is obtained by a minor syntactic modification of the schema in Dybjer [20]. This adds evidence to the fundamental nature of the schematic natural deduction formulation of inductive definitions in type theory.

The idea to consider this generalisation was inspired by Nax Mendler’s paper [32] on the category-theoretic semantics of universes in type theory. Our analysis improves fundamentally on Mendler’s, since the category-theoretic machinery can be applied only if the rules for U_0 and T_0 already have been represented as an endofunctor on a category of families of sets. This representation is not itself analysed and also loses the U_0 -recursive nature of T_0 . It is not clear how to use category-theoretic ideas for obtaining a formal system for simultaneous induction-recursion.

The idea to enrich Frege structures with proof objects can also be found in Sato [40]. However, Sato works in a type-free constructive theory and not in type theory. Working in the same framework as Sato, Kameyama [25] has developed an approach to half-positive inductive definitions. His aims are similar to ours: to formulate a general notion which subsumes the construction of Frege structures and enables the interpretation of Martin-Löf type theory. In his type-free context the distinction between inductively and recursively defined sets (that our approach is based on) does not exist. Instead he considers simultaneous inductive definitions which give rise to operators which are monotone in the sense of an ordering which generalises Aczel’s ordering for Frege structures described above.

As further examples of simultaneous induction-recursion, we would like to mention in particular the computability predicates used by Martin-Löf [31, 27] and C. Coquand [12] for proving normalisation of type theory, and the logical relations introduced by T. Coquand [13] for proving soundness and completeness of an algorithm for testing conversion in type theory. These constructions are similar in nature to the collections of propositions and truths in a Frege structure, and can be given a classical explanation in an analogous way. By defining constructions of these kinds by simultaneous induction-recursion we have paved the way for “internal type theory” [21], that is, to locally reflect the metatheory of type theory in itself. In particular, we hope to extend the technique of reduction-free normalization [15, 11, 17] developed for the simply typed case to dependent types.

References

- [1] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland, 1977.
- [2] P. Aczel. The strength of Martin-Löf’s type theory with one universe. In S. Miettinen and J. Väänanen, editors, *Proceedings of the Symposium on Mathematical Logic (Oulu 1974)*, pages 1–32, 1977. Report No 2 of Dept. Philosophy, University of Helsinki.
- [3] P. Aczel. The type theoretic interpretation of constructive set theory. In A. MacIntyre, L. Pacholski, and J. Paris, editors, *Logic Colloquium ’77*, pages 55–66. North-Holland, 1978.
- [4] P. Aczel. *Frege Structures and the Notions of Proposition, Truth, and Set*, pages 31–59. North-Holland, 1980.
- [5] S. Allen. *A Non-Type-Theoretic Semantics for Type-Theoretic Language*. PhD thesis, Department of Computer Science, Cornell University, 1987.
- [6] T. Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, The University of Edinburgh, Department of Computer Science, November 1993.
- [7] R. Backhouse. On the meaning and construction of the rules in Martin-Löf’s theory of types. In *Proceedings of the Workshop on General Logic, Edinburgh, February 1987*. Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1988. ECS-LFCS-88-52.
- [8] H. P. Barendregt. *The Lambda Calculus*. North-Holland, 1984. Revised edition.
- [9] R. Bird and P. Wadler. *Introduction to Functional Programming*. Prentice Hall, 1988.
- [10] E. Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, 1967.
- [11] C. Coquand. From semantics to rules: a machine assisted analysis. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Proceedings of CSL ’93, LNCS 832*, 1993.
- [12] C. Coquand. A realizability interpretation of Martin-Löf’s type theory. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998. To appear.
- [13] T. Coquand. An algorithm for testing conversion in type theory. In *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.
- [14] T. Coquand. Pattern matching with dependent types. In *Proceedings of The 1992 Workshop on Types for Proofs and Programs*, June 1992.
- [15] T. Coquand and P. Dybjer. Intuitionistic model constructions and normalization proofs. *Mathematical Structures in Computer Science*, 7:75–94, 1997.
- [16] T. Coquand and C. Paulin. Inductively defined types, preliminary version. In *LNCS 417, COLOG ’88, International Conference on Computer Logic*. Springer-Verlag, 1990.

- [17] D. Čubrić, P. Dybjer, and P. Scott. Normalization and the Yoneda embedding. *Mathematical Structures in Computer Science*, 1998. To appear.
- [18] G. Dowek, A. Felty, H. Herbelin, G. Huet, C. Paulin, and B. Werner. The Coq proof assistant version 5.6, user's guide. Technical report, INRIA Rocquencourt - CNRS ENS Lyon, 1991.
- [19] P. Dybjer. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In *Logical Frameworks*, pages 280–306. Cambridge University Press, 1991.
- [20] P. Dybjer. Inductive families. *Formal Aspects of Computing*, pages 440–465, 1994.
- [21] P. Dybjer. Internal type theory. In *TYPES '95, Types for Proofs and Programs*, number 1158 in Lecture Notes in Computer Science, pages 120–134. Springer, 1996.
- [22] E. Giménez. A command for inductive sets in ILF. Master Thesis, Universidad de la República, Montevideo, 1992.
- [23] E. Griffor and M. Rathjen. The strength of some Martin-Löf type theories. *Archive of Mathematical Logic*, 33:337–385, 1994.
- [24] M. Hedberg. *Type Theory and the External Logic of Programs*. PhD thesis, Chalmers University of Technology and University of Göteborg, 1994.
- [25] Y. Kameyama. A type-free theory of half-monotone inductive definitions. *International Journal of Foundations of Computer Science*, 6(3):203–234, 1995.
- [26] P. Martin-Löf. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 179–216. North-Holland, 1971.
- [27] P. Martin-Löf. An intuitionistic theory of types: Predicative part. In *Logic Colloquium '73*, pages 73–118. North-Holland, 1975.
- [28] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science, VI, 1979*, pages 153–175. North-Holland, 1982.
- [29] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.
- [30] P. Martin-Löf. Amendment to intuitionistic type theory. Notes from a lecture given in Göteborg, March 1986.
- [31] P. Martin-Löf. An intuitionistic theory of types. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998. To appear. Reprinted version of an unpublished report from 1972.
- [32] P. F. Mendler. Predicative type universes and primitive recursion. In *Proceedings Sixth Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1991.
- [33] R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML*. MIT Press, 1990.
- [34] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: an Introduction*. Oxford University Press, 1990.
- [35] E. Palmgren. *On Fixed Point Operators, Inductive Definitions and Universes in Martin-Löf's Type Theory*. PhD thesis, Uppsala University, 1991.
- [36] E. Palmgren. On universes in type theory. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory*. Oxford University Press, 1998. Preprint. To appear.
- [37] M. Parigot. Programming with proofs: a second order type theory. In H. Ganzinger, editor, *ESOP'88, 2nd European Symposium on Programming, Nancy, LNCS 300*, pages 145–159, March 1988.

- [38] C. Paulin-Mohring. Inductive definitions in the system Coq - rules and properties. In *Proceedings Typed λ -Calculus and Applications*, pages 328–245. Springer-Verlag, LNCS, March 1993.
- [39] M. Rathjen, E. R. Griffor, and E. Palmgren. Inaccessibility in constructive set theory and type theory. *Annals of Pure and Applied Logic*, 1998. To appear.
- [40] M. Sato. Adding proof objects and inductive definition mechanism to Frege structures. In T. Ito and A. Meyer, editors, *Proc. International Conference on Theoretical Aspects of Computer Science*, number 526 in LNCS, pages 53–87. Springer Verlag, 1991.
- [41] A. Setzer. *Proof theoretical strength of Martin-Löf Type Theory with W-type and one universe*. PhD thesis, Fakultät für Mathematik der Ludwig-Maximilians-Universität München, 1993.
- [42] A. Setzer. Extending Martin-Löf Type Theory by one Mahlo-universe. Preprint, 1996.
- [43] J. Smith. The independence of Peano’s fourth axiom from Martin-Löf’s type theory without universes. *Journal of Symbolic Logic*, 49(3), 1988.
- [44] J. Smith. Propositional functions and families of types. *Notre Dame Journal of Formal Logic*, 30(3):442–458, 1989.
- [45] B. Werner. A normalization proof for an impredicative type system with large elimination over integers. In B. Nordström, K. Petersson, and G. Plotkin, editors, *Proceedings of the 1992 Workshop on Proofs and Programs*, pages 361–377. Department of Computer Sciences, Chalmers University of Technology, June 1992.

FAITHFUL IDEAL MODELS FOR RECURSIVE POLYMORPHIC TYPES*

MARTÍN ABADI
*Digital Equipment Corporation
Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301, USA*

BENJAMIN PIERCE[†]
*School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA*

GORDON PLOTKIN[‡]
*Department of Computer Science
King's Building
University of Edinburgh
Edinburgh, EH9 3JZ, UK*

ABSTRACT

We explore ideal models for a programming language with recursive polymorphic types, variants of the model studied by MacQueen, Plotkin, and Sethi. The use of suitable ideals yields a close fit between models and programming language. Two of our semantics of type expressions are faithful, in the sense that programs that behave identically in all contexts have exactly the same types.

Keywords: Polymorphism, recursive types, full abstraction, ideals, metric models.

1. Introduction

Often, a formal semantics assigns different values to programs that behave identically in all contexts [1, 2, 3]. In other words, the semantics of programming-language expressions is not fully abstract. This mismatch between model and programming language elicits diverse reactions. Some propose extensions to the programming language, while others prefer modifying the semantics. Both of these attitudes have suggested fruitful lines of research. Yet others point out that the semantics still serves its purposes fairly well, for example in proving the safety of evaluation schemes. At any rate, full abstraction is one of the main criteria in assessing programming-language semantics [4].

A similar situation commonly arises in the semantics of type systems. In its simplest form, a semantics for a type system maps each type expression to a reasonable subset of the domain of values. Let us call these subsets types. In most semantics,

*A preliminary version of this paper was presented at the 4th IEEE Symposium on Logic in Computer Science in June 1989.

[†]This work was started at Digital Equipment Corporation, Systems Research Center.

[‡]This work was started at the Center for the Study of Language and Information, Stanford University.

some programs that behave identically in all contexts not only may receive different meanings—they may even belong to different types. Roughly, we call a semantics *faithful* if programs that behave identically in all contexts have identical types.

In logical form, the faithfulness problem concerns the soundness of the usual rule of inference

$$\frac{e : T \quad e' = e}{e' : T}$$

which says that if e has type T and e' and e are “equal” then e' has type T . In the standard reading of this rule, two expressions are “equal” when they are equal in a given model. We take two expressions to be “equal” when they behave identically in all contexts. A faithful semantics should validate the rule under this interpretation.

Clearly, a semantics of type expressions is automatically faithful if the underlying semantics of expressions is fully abstract. The converse is false, however, and it seems intriguing (and perhaps easier) to study faithfulness in isolation from full abstraction.

In this paper, we give semantics for a language with recursive polymorphic types, like Standard ML [5]. Imitating MacQueen, Plotkin, and Sethi, we adopt and extend the ideal model [6]. In the ideal model, type expressions are interpreted as ideals, certain subsets of the universe of values. We place new restrictions on ideals, defining the classes of generated ideals, coarse ideals, and abstract ideals. In our variants of the model, type expressions are interpreted as ideals in these classes. We obtain faithful semantics of type expressions. These semantics validate the stronger rule of inference

$$\frac{e : T \quad e' \sqsubseteq e}{e' : T}$$

which says that if e has type T and e' is “less than” e then e' has type T , with “less than” interpreted contextually (just as “equal” above).

One danger in tampering with a model is making it mathematically intractable, for instance by making it too syntactic too soon. We may also simply give up useful properties. Fortunately, our type systems share many features with the one explored by MacQueen, Plotkin, and Sethi. In particular, a large family of recursive type equations have unique solutions.

The departures from the original ideal model are perhaps best illustrated with an example. Consider the following expression of the untyped lambda calculus:

```
explode-||-or = λ x.
  if x(true, btm)
  and x(btm, true)
  and not x(false, false)
  then wrong
  else true
```

where `btm` is a divergent expression, such as $(\lambda y. y y)$ $(\lambda y. y y y)$, and `wrong` represents a dynamic type error.

How should we type the expression `explode-||-or`? The first branch of the `if-then-else` is only taken when the input represents the parallel-or function,

which returns `true` whenever either of its arguments is `true`, even if the other argument diverges. In many common models, and in the one considered by MacQueen, Plotkin, and Sethi in particular, the parallel-or function exists. In these models, `explode-||-or` returns `wrong` with a binary boolean function as input, and hence it seems ill-typed. On the other hand, it is well-known that the parallel-or function cannot be expressed in the usual lambda calculus, as studied by Plotkin [2]. Therefore, whenever `explode-||-or` is applied to an expression in the language that represents a binary boolean function, `explode-||-or` returns `true` if it terminates. Thus, intuitively, there seem to be grounds for asserting that `explode-||-or` should have type $(\text{bool} \times \text{bool} \rightarrow \text{bool}) \rightarrow \text{bool}$, the type of functions that map binary boolean functions to boolean values. This is the case in our semantics.

In the next section we describe the setting for this work: a language, a model, the semantics of the language in the model, and some useful expressions of the language. We also define faithfulness precisely. In Section 3 we study the generated ideals; this section contains analogues to the central results in [6]. In Section 4 we use the generated ideals to give a semantics to type expressions, and then restrict our attention to special classes of generated ideals; we prove that two semantics are faithful. Finally, in Section 5, we consider more syntactic approaches, where types are sets of terms.

2. The Setting

We work within the ideal model of types, following MacQueen, Plotkin, and Sethi. The setting is almost exactly identical to theirs. We refer the reader to their paper for definitions and lemmas that we omit or only sketch, and recommend the study of their paper as a preliminary for fully understanding the constructions below.

In this section we briefly review a simple programming language, a model, and the denotational semantics of the programming language in the model. Then we define faithfulness. We also study some expressions of the programming language that play a role in the formulation of syntactic counterparts to semantic concepts.

2.1. The Programming Language

The basic programming language is a simple untyped lambda calculus with constants. We start with the following grammar:

$$e ::= c \mid x \mid \lambda x.e \mid e(e')$$

Here, e and e' range over the set EXP of expressions, c over a suitable set of constants, and x over the set VAR of variables. We insist on having a sufficiently rich set of constants, such as the one considered in [6]: `true`, `false`, `cond`, `0`, `z` (test for zero), `+1`, `-1`, `pair`, π_1 and π_2 (for extracting pair elements), `inl` and `inr` (for forming sums), `outl` and `outr` (to recover elements from sums), and `isl` and `isr` (for distinguishing cases in sums). We also use abbreviations, such as `and`, `if-then-else`, and $x < m$ (with m fixed).

The only new feature of our language is an additional `cases` construct. Intuitively, this construct enables us to decide whether e_{sel} is a boolean, a natural, a

pair, a sum, or a function, and to return a different result in each of the cases:

```

cases esel
  bool: ebool
  nat: enat
  pair: epair
  sum: esum
  fun: efun
end

```

It can be argued that the addition of **cases** is natural, and even that **cases** either should be definable or ought to have been in the programming language in the first place. At any rate, we use **cases** only for writing a limited family of expressions. It is common to assume that similar expressions are definable [1], and we include **cases** only to make their definition straightforward.

2.2. The Model

To give a semantics to this language, we consider a universe \mathbf{V} that satisfies the isomorphism equation

$$\mathbf{V} \cong \mathbf{T} + \mathbf{N} + (\mathbf{V} \times \mathbf{V}) + (\mathbf{V} + \mathbf{V}) + (\mathbf{V} \rightarrow \mathbf{V}) + \mathbf{W},$$

where \mathbf{T} is the flat domain of boolean values, \mathbf{N} is the flat domain of natural numbers, \mathbf{W} is the type-error domain $\{w\}_{\perp}$, and \times , $+$, and \rightarrow represent the usual product, coalesced sum, and continuous function space operations, respectively (coalesced sum identifies the least elements of its two arguments). The value w represents all dynamic type errors.

The universe \mathbf{V} can be constructed as the limit of a sequence of approximations $\mathbf{V}_0, \mathbf{V}_1, \dots$, where

$$\mathbf{V}_0 = \{\perp\}$$

and, for all i ,

$$\mathbf{V}_{i+1} = \mathbf{T} + \mathbf{N} + (\mathbf{V}_i \times \mathbf{V}_i) + (\mathbf{V}_i + \mathbf{V}_i) + (\mathbf{V}_i \rightarrow \mathbf{V}_i) + \mathbf{W}.$$

We omit the details of the construction, which are standard [7, 6].

As usual, we have an ordering \sqsubseteq on \mathbf{V} ; we read $x \sqsubseteq y$ as “ x is less defined than y ” or “ x approximates y .“ An element of \mathbf{V} is finite if whenever it approximates the least upper bound of a chain it approximates some element of the chain. The function μ_i is the embedding from \mathbf{V}_i to \mathbf{V} , and μ_i^R is its inverse projection. The rank $r(a)$ of a finite element a of \mathbf{V} is the least i such that a “appears” in \mathbf{V}_i during the construction of \mathbf{V} as a limit; the rank function is only defined on finite elements. More precisely, $r(a)$ equals the least i such that $a = \mu_i \circ \mu_i^R(a)$, provided a is finite.

2.3. The Semantics of Expressions

We assign a meaning $\llbracket e \rrbracket$ to each expression e of the programming language. The definition of $\llbracket \cdot \rrbracket$ appears in Fig. 1, with the following notation:

- s in \mathbf{V} , where s belongs to a summand \mathbf{S} of \mathbf{V} , is the injection of s into \mathbf{V} ;

- **wrong** is an abbreviation for w in \mathbf{V} ;
- if $v = (s \text{ in } \mathbf{V})$ for some $s \in \mathbf{S}$ then $v|_{\mathbf{S}}$ is s , and otherwise it is \perp ;
- $v \sqsubset \mathbf{S}$ yields \perp if v is \perp , true if $v = (s \text{ in } \mathbf{V})$ for some $s \in \mathbf{S}$ (with $s \neq \perp$), and false otherwise;
- a mapping in $\text{VAR} \rightarrow \mathbf{V}$ is called an environment; $\rho\{\mathbf{x} \leftarrow v\}$ is the environment obtained from ρ by giving the value v to the variable \mathbf{x} .

We omit the equations for constants.

$\llbracket \cdot \rrbracket : \text{EXP} \rightarrow (\text{VAR} \rightarrow \mathbf{V}) \rightarrow \mathbf{V}$	
$\llbracket \mathbf{x} \rrbracket_\rho$	$= \rho(\mathbf{x})$
$\llbracket \lambda \mathbf{x}. \mathbf{e}_{\text{body}} \rrbracket_\rho$	$= (\lambda v. \llbracket \mathbf{e}_{\text{body}} \rrbracket_{\rho\{\mathbf{x} \leftarrow v\}}) \text{ in } \mathbf{V}$
$\llbracket \mathbf{e}_{\text{fun}}(\mathbf{e}_{\text{arg}}) \rrbracket_\rho$	$= \text{let } v = \llbracket \mathbf{e}_{\text{fun}} \rrbracket_\rho \text{ in}$ $\quad \text{if } v \sqsubset (\mathbf{V} \rightarrow \mathbf{V}) \text{ then } (\llbracket \mathbf{e}_{\text{fun}} \rrbracket_\rho _{\mathbf{V} \rightarrow \mathbf{V}})(\llbracket \mathbf{e}_{\text{arg}} \rrbracket_\rho)$ $\quad \text{else wrong}$
$\llbracket \text{cases } \mathbf{e}_{\text{sel}} \text{ bool: } \mathbf{e}_{\text{bool}} \dots \rrbracket_\rho$	$= \text{let } v = \llbracket \mathbf{e}_{\text{sel}} \rrbracket_\rho \text{ in}$ $\quad \text{if } v \sqsubset \mathbf{T} \text{ then } \llbracket \mathbf{e}_{\text{bool}} \rrbracket_\rho$ $\quad \text{else if ...}$ $\quad \text{else wrong}$

Fig. 1: Definition of the meaning function for expressions

In our constructions, we are particularly interested in denotable elements, that is, the elements in the range of $\llbracket \cdot \rrbracket$:

Definition 2.3.1. *An element $v \in \mathbf{V}$ is denutable if $v = \llbracket \mathbf{e} \rrbracket$ for some closed expression \mathbf{e} .*

In the rest of this paper, as in this definition, we often do not mention the environment ρ when it is irrelevant. Specifically, if \mathbf{e} is a closed expression then, informally, we may write $\llbracket \mathbf{e} \rrbracket$ instead of $\llbracket \mathbf{e} \rrbracket_\rho$, since $\llbracket \mathbf{e} \rrbracket_\rho$ clearly does not depend on ρ .

2.4. Faithfulness

At this point, a precise discussion of full abstraction and faithfulness is in order.

Roughly, a semantics of expressions is fully abstract if two expressions that behave identically in all contexts receive the same interpretation. We could be interested in different aspects of the behavior of an expression. We find it sufficient here to “observe termination,” that is, to identify the behavior of \mathbf{e} with the set of all contexts \mathcal{C} such that $\llbracket \mathcal{C}[\mathbf{e}] \rrbracket \neq \perp$ —or, equivalently, with the set of all contexts \mathcal{C} such that $\llbracket \mathcal{C}[\mathbf{e}] \rrbracket = \perp$. In other words, we say that two expressions \mathbf{e} and \mathbf{e}' behave identically if, for all contexts \mathcal{C} , we have $\llbracket \mathcal{C}[\mathbf{e}] \rrbracket = \perp$ if and only if $\llbracket \mathcal{C}[\mathbf{e}'] \rrbracket = \perp$. For

the definition to make sense, we consider only the contexts such that $\mathcal{C}[e]$ and $\mathcal{C}[e']$ are closed; we leave similar requirements implicit in the rest of the paper.

Other notions of observation are also justifiable; for instance, we might care about the set of contexts \mathcal{C} such that $\llbracket \mathcal{C}[e] \rrbracket$ is a boolean value. Fortunately, many reasonable kinds of observation are equivalent to the one we choose (see [4] for an informal discussion of this point).

Thus, full abstraction requires that, for all e and e' , whenever $\llbracket \mathcal{C}[e] \rrbracket = \perp$ if and only if $\llbracket \mathcal{C}[e'] \rrbracket = \perp$ for all contexts \mathcal{C} , it must also be the case that $\llbracket e \rrbracket = \llbracket e' \rrbracket$. An interesting refinement of this definition allows the ordering relation \sqsubseteq to come into play. In this refinement, for all e and e' , whenever $\llbracket \mathcal{C}[e] \rrbracket = \perp$ implies $\llbracket \mathcal{C}[e'] \rrbracket = \perp$ for all contexts \mathcal{C} , it must also be the case that $\llbracket e' \rrbracket \sqsubseteq \llbracket e \rrbracket$. (The converse holds trivially.)

Let us write $e \models \mathcal{C}$ if $\llbracket \mathcal{C}[e] \rrbracket \neq \perp$, and $e' \sqsubseteq e$ if $e' \models \mathcal{C}$ implies $e \models \mathcal{C}$ for all \mathcal{C} . In this notation, the refined definition of full abstraction simply says that \sqsubset and \sqsubseteq are identical on denotable elements.

Note that, in either case, as in [1], full abstraction is a purely semantic criterion and does not refer to a particular operational semantics. This is a convenient way to consider the full-abstraction property as an intrinsic feature of a semantics.

Analogously, a semantics of type expressions is faithful if two expressions that behave identically in all contexts have identical types. For this, it suffices that each type be abstract, in the sense that membership in the type does not distinguish between expressions that behave identically in all contexts. This tentative definition can also be refined to take the ordering relation into account, and motivates a stronger definition:

Definition 2.4.1. *A set I is abstract if, for all closed expressions e and e' , if $e' \sqsubset e$ and $\llbracket e \rrbracket \in I$ then $\llbracket e' \rrbracket \in I$.*

In an expanded, symbolic form, a set I is abstract if

$$\forall e, e'. [\forall \mathcal{C}. (\llbracket \mathcal{C}[e'] \rrbracket = \perp \supset \llbracket \mathcal{C}[e] \rrbracket = \perp) \supset (\llbracket e' \rrbracket \in I \supset \llbracket e \rrbracket \in I)].$$

Informally, the abstract sets are “closed downwards” in \sqsubset . This requirement follows from common intuitions on the structure of types, the same intuitions that underlie the use of ideals (we come back to this point in Section 3).

Finally, we can define faithfulness. The definition implies the one we have been using informally.

Definition 2.4.2. *Given a set of well-formed type expressions, a semantics is faithful if the meaning of each well-formed type expression is abstract.*

In logical terms, then, we require the soundness of the rule

$$\frac{e : T \quad e' \sqsubseteq e}{e' : T}$$

with \sqsubseteq interpreted as \sqsubset .

2.5. Projection Functions

The following expressions are useful in establishing correspondences between syntax and semantics. In particular, they are related to the rank function. We use these expressions and some of their properties (given in Lemma 2.5.1) in the main proofs below.

First we define a sequence of expressions p_n^m :

```

 $p_0^m = \text{btm}$ 

 $p_{n+1}^m = \lambda x.$ 
  cases x
  bool: x
  nat: if  $x < m$  then x else btm
  pair: pair( $p_n^m(\pi_1 x)$ ,  $p_n^m(\pi_2 x)$ )
  sum: if  $\text{isl}(x)$  then  $\text{inl}(p_n^m(\text{outl } x))$ 
        else  $\text{inr}(p_n^m(\text{outr } x))$ 
  fun:  $p_n^m \circ x \circ p_n^m$ 
end

```

where, as before, btm is a divergent expression, that is, an expression that represents \perp , such as $(\lambda y. y y) (\lambda y. y y)$. Intuitively, each p_n^m is a projection that maps an object to an approximation with smaller rank.

We also define a sequence of expressions p_n :

```

 $p_0 = \text{btm}$ 

 $p_{n+1} = \lambda x.$ 
  cases x
  bool: x
  nat: x
  pair: pair( $p_n(\pi_1 x)$ ,  $p_n(\pi_2 x)$ )
  sum: if  $\text{isl}(x)$  then  $\text{inl}(p_n(\text{outl } x))$ 
        else  $\text{inr}(p_n(\text{outr } x))$ 
  fun:  $p_n \circ x \circ p_n$ 
end

```

Intuitively, each p_n is the limit of the expressions p_n^m as m grows, and is also a projection. The limit of the expressions p_n as n grows is the identity function.

Lemma 2.5.1. *Let id be the identity function in \mathbf{V} ; for all m and n ,*

1. $\llbracket p_n^m \rrbracket$ has a finite range;
2. $\llbracket p_n^m \rrbracket \sqsubseteq \text{id}$;
3. $\llbracket p_n^m \rrbracket \circ \llbracket p_n^m \rrbracket = \llbracket p_n^m \rrbracket$;

4. all elements in the range of $\llbracket p_n^m \rrbracket$ are finite;
5. $\llbracket p_n^m \rrbracket \sqsubseteq \llbracket p_n^{m+1} \rrbracket$ and $\llbracket p_n^m \rrbracket \sqsubseteq \llbracket p_{n+1}^m \rrbracket$;
6. $\llbracket p_n \rrbracket = \bigsqcup_m \llbracket p_n^m \rrbracket$;
7. $\llbracket p_n \rrbracket = \mu_n \circ \mu_n^R$;
8. $\text{id} = \bigsqcup_n \llbracket p_n \rrbracket = \bigsqcup_n \llbracket p_n^n \rrbracket$;
9. $\llbracket p_n \rrbracket \circ \llbracket p_n^m \rrbracket = \llbracket p_n^m \rrbracket$.

Proof We prove the claims in order.

1. $\llbracket p_n^m \rrbracket$ has a finite range: The proof is by induction on n . The case $n = 0$ is trivial. Assume that the proposition holds for some n , to check it for $n+1$. The range of $\llbracket p_{n+1}^m \rrbracket$ consists of numbers less than m , pairs and sums of elements in the range of $\llbracket p_n^m \rrbracket$, and functions into the range of $\llbracket p_n^m \rrbracket$ and determined by their values on the range of $\llbracket p_n^m \rrbracket$. By the induction hypothesis, this set must be finite.
2. $\llbracket p_n^m \rrbracket \sqsubseteq \text{id}$: Again, we use induction on n . The base case is trivial. The inductive step follows immediately from the monotonicity of all functions.
3. $\llbracket p_n^m \rrbracket \circ \llbracket p_n^m \rrbracket = \llbracket p_n^m \rrbracket$: Once more, we use induction on n . The base case is trivial. The inductive step follows from the remarks in 1 on the nature of $\llbracket p_{n+1}^m \rrbracket$'s range.
4. All elements in the range of $\llbracket p_n^m \rrbracket$ are finite: Suppose that for some e and for some chain $\langle y_l \rangle$ we have $\llbracket p_n^m(e) \rrbracket \sqsubseteq \bigsqcup_l y_l$. Then $\llbracket p_n^m(e) \rrbracket \sqsubseteq \bigsqcup_l \llbracket p_n^m \rrbracket(y_l)$ (by the continuity of $\llbracket p_n^m \rrbracket$ and 3). Hence $\llbracket p_n^m(e) \rrbracket \sqsubseteq \llbracket p_n^m \rrbracket(y_k)$ for some k (by 1), and, finally, $\llbracket p_n^m(e) \rrbracket \sqsubseteq y_k$ for some k (by 2).
5. $\llbracket p_n^m \rrbracket \sqsubseteq \llbracket p_n^{m+1} \rrbracket$ and $\llbracket p_n^m \rrbracket \sqsubseteq \llbracket p_{n+1}^m \rrbracket$: The two propositions are proved separately, by induction on m and n , respectively. Both arguments are trivial.
6. $\llbracket p_n \rrbracket = \bigsqcup_m \llbracket p_n^m \rrbracket$: First, $\bigsqcup_m \llbracket p_n^m \rrbracket$ is defined, by 5. Furthermore, it is simple to prove by induction on n that $\llbracket p_n \rrbracket \sqsubseteq \bigsqcup_m \llbracket p_n^m \rrbracket$ (using continuity) and $\bigsqcup_m \llbracket p_n^m \rrbracket \sqsubseteq \llbracket p_n \rrbracket$ (using monotonicity).
7. $\llbracket p_n \rrbracket = \mu_n \circ \mu_n^R$: The proof is by induction on n and requires elementary properties of the embeddings μ_n . Specifically, it requires that $\mu_0 \circ \mu_0^R$ is totally undefined, that $\mu_{n+1} \circ \mu_{n+1}^R$ is the identity function on boolean values and numbers, that if f is a function then $\mu_{n+1} \circ \mu_{n+1}^R(f) = (\mu_n \circ \mu_n^R) \circ f \circ (\mu_n \circ \mu_n^R)$, and similar properties for pairs and sums. The argument is straightforward.
8. $\text{id} = \bigsqcup_n \llbracket p_n \rrbracket = \bigsqcup_n \llbracket p_n^n \rrbracket$: The former equality follows from 7 while the latter one follows from 6.
9. $\llbracket p_n \rrbracket \circ \llbracket p_n^m \rrbracket = \llbracket p_n^m \rrbracket$: This follows directly from 3 and 6, which implies that $\llbracket p_n^m \rrbracket \sqsubseteq \llbracket p_n \rrbracket \sqsubseteq \text{id}$. \square

These properties can now be used to prove the following lemma, which explains how to reduce the rank of a denotable value $\llbracket e \rrbracket$ while remaining “above” a given lower bound a .

Lemma 2.5.2. *If $r(a) = n$ and $a \sqsubseteq \llbracket e \rrbracket$ then there exists m such that $a \sqsubseteq \llbracket p_n^m(e) \rrbracket \sqsubseteq \llbracket e \rrbracket$ and $r(\llbracket p_n^m(e) \rrbracket) \leq n$.*

Proof This follows easily from Lemma 2.5.1.

Because a has rank n , $\llbracket p_n \rrbracket(a) = a$, and hence the monotonicity of $\llbracket p_n \rrbracket$ guarantees that $a \sqsubseteq \llbracket p_n \rrbracket \llbracket e \rrbracket$. In addition, since $\llbracket p_n \rrbracket \llbracket e \rrbracket = \bigsqcup_k \llbracket p_n^k \rrbracket \llbracket e \rrbracket$ and a is finite, there exists m such that $a \sqsubseteq \llbracket p_n^m \rrbracket \llbracket e \rrbracket$, that is, $a \sqsubseteq \llbracket p_n^m(e) \rrbracket$. Furthermore, $\llbracket p_n^m(e) \rrbracket \sqsubseteq \llbracket e \rrbracket$, since $\llbracket p_n^m \rrbracket$ is a projection.

Finally, $\llbracket p_n^m(e) \rrbracket$ is finite; Lemma 2.5.1 yields that $\llbracket p_n \rrbracket \llbracket p_n^m(e) \rrbracket = \llbracket p_n^m(e) \rrbracket$, and hence $r(\llbracket p_n^m(e) \rrbracket) \leq n$. \square

3. On Tying Ideals to Language

In the original ideal model, two different ideals may contain exactly the same denotable elements. Hence these ideals cannot really be distinguished from the point of view of the programming language; their distinction may be regarded as irrelevant, or even bothersome. Analogous situations arise in other typical semantics.

In this section we study a notion of type with closer ties to the programming language. Each type can be obtained from the set of its denotable elements. Since these types are ideals, we call them denotably-generated ideals, or generated ideals for short. We define functions on generated ideals and prove that these functions share many of the desirable mathematical properties of the corresponding functions on arbitrary ideals.

3.1. Generated Ideals

As is commonly argued, types consist of sets of values with a common structure. Intuitively, the structural criteria embodied by type distinctions should be preserved “downwards” and “under limits” in the \sqsubseteq ordering. This principle underlies the identification of types with ideals:

Definition 3.1.1. *A set I is an ideal if*

- $I \neq \emptyset$;
- for all x and y , if $x \sqsubseteq y$ and $y \in I$ then $x \in I$; and
- for all increasing sequences $\langle x_n \rangle$, if $x_n \in I$ for all n then $\bigsqcup_n x_n \in I$.

Using Lemma 2.5.1, one can easily show that there is a least ideal containing any given non-empty set of values, X . It is

$$Id(X) = \{x \mid \text{for all } n, \llbracket p_n^n \rrbracket(x) \sqsubseteq \text{some element of } X\}.$$

Generated ideals are ideals obtained from their denotable elements, as follows:

Definition 3.1.2. *I is a generated ideal if*

- I is an ideal; and
- if $x \in I$ then there exists an increasing sequence of denotable values $\langle x_n \rangle$ such that $x_n \in I$ for all n and $x \sqsubseteq \bigsqcup x_n$.

Analogously to the case of ideals, one has:

Theorem 3.1.3. *If X is a set of denotable elements, then $Id(X)$ is generated.*

Proof Certainly $Id(X)$ is an ideal. Suppose that x is an element of it. Set

$$R_n = \{z \in [\![p_n]\!](V) \mid z \text{ is denotable, } z \in Id(X), \text{ and } [\![p_n]\!](x) \sqsubseteq z\}.$$

By Lemma 2.5.1 this set is finite. It is also non-empty as for any n there is a y in X such that $[p_n](x) \sqsubseteq y$. But then as y is denotable, and using Lemma 2.5.1 again, $[p_n](y)$ is in R_n . Finally, for every z' in R_{n+1} there is a z in R_n such that $z \sqsubseteq z'$ (take $z = [p_n](z')$ and use Lemma 2.5.1). We can therefore apply König's Lemma to the R_n to find an increasing sequence $\langle z_n \rangle$ (z_n in R_n), and using Lemma 2.5.1 one sees that this sequence verifies the second condition for $Id(X)$ being a generated ideal. \square

It follows that the set of generated ideals, **GIdl**, is a complete lattice, with for any collection I_λ ($\lambda \in \Lambda$) of generated ideals:

$$\begin{aligned}\sqcup I_\lambda &= Id(\{x \in \sqcup I_\lambda \mid x \text{ is denotable}\}), \\ \sqcap I_\lambda &= Id(\{x \in \cap I_\lambda \mid x \text{ is denotable}\}).\end{aligned}$$

It is worth remarking that $I \sqcup J$ is $I \cup J$, that $\sqcup I_\lambda$ has the same finite denotable elements as $\cup I_\lambda$, although not, in general, the same denotable ones, and that $\sqcap I_\lambda$ has the same denotable elements as $\cap I_\lambda$.

Ideals are mathematically tractable in part because they are determined by their finite elements: indeed if I is an ideal then

$$I = Id(\{x \in I \mid x \text{ is finite}\}).$$

Analogously, generated ideals are mathematically tractable in part because they are determined by their finite denotable elements, as we show in Proposition 3.1.4 and Theorem 3.1.5. (But, not surprisingly, constructions with generated ideals sometimes require more work and care than their analogues with arbitrary ideals.)

Proposition 3.1.4. *Every denotable value is the limit of a chain of finite denotable values.*

Proof Every denotable element $[\![e]\!]$ is the limit of an increasing sequence $\langle [\![p_n(e)]\!] \rangle$ of finite denotable elements, by Lemma 2.5.1. \square

Theorem 3.1.5. *Let I be a generated ideal. Then*

$$I = Id(\{x \in I \mid x \text{ is finite and denotable}\}).$$

Proof In one direction, $I = Id(\{x \in I \mid x \text{ is finite}\}) \supseteq Id(\{x \in I \mid x \text{ is finite and denotable}\})$. In the other direction, by Proposition 3.1.4 every denotable element of I is in $Id(\{x \in I \mid x \text{ is finite and denotable}\})$, and then the second condition for I to be generated ensures that every element of I is in this set. \square

3.2. Operations and Metric

In this subsection we define some operations and a metric on generated ideals. Most of the typical operations on ideals are unchanged for generated ideals; we define new versions of intersection and exponentiation. The metric is also new. Some symbols in the definitions below represent both functions on generated ideals within \mathbf{V} and functions on domains such as \mathbf{V} ; the intended meaning of the symbols should be clear from context.

Definition 3.2.1. Let I, J, J_1, \dots, J_n be generated ideals, \mathcal{K} a collection of generated ideals, and f a function from \mathbf{GIdl}^{n+1} to \mathbf{GIdl} .

- \mathbf{T} and \mathbf{N} are the generated ideals of boolean values and natural numbers, respectively.
- $I \times J$ is the product of I and J .
- $I + J$ is the coalesced sum of I and J (+ identifies the least elements of its two arguments).
- $I \rightarrow J$ is a special sort of exponentiation of I and J : the smallest generated ideal that contains $\{\llbracket e \rrbracket \in (\mathbf{V} \rightarrow \mathbf{V}) \mid \text{if } \llbracket a \rrbracket \in I \text{ then } \llbracket ea \rrbracket \in J\}$.
- $(\forall_{\mathcal{K}} f)(J_1, \dots, J_n) = \sqcap_{I \in \mathcal{K}} f(I, J_1, \dots, J_n)$.
- $(\exists_{\mathcal{K}} f)(J_1, \dots, J_n) = \sqcup_{I \in \mathcal{K}} f(I, J_1, \dots, J_n)$.
- $(\mu f)(J_1, \dots, J_n)$ is the unique I such that $I = f(I, J_1, \dots, J_n)$, provided that f is contractive, in the sense defined below.

It is simple to check that all the definitions are proper, with the exception of the definition of μ . That the definition of μ is proper follows from the Banach Fixpoint Theorem [8], stated below as Theorem 3.2.5. It is worth pointing out that our definition of $I \rightarrow J$ differs from the usual one even on denotable elements.

The usual metric on arbitrary ideals does not suit our purposes; in particular, we cannot exploit it to demonstrate the existence of solutions to recursive type equations. Generated ideals call for a different metric.

Definition 3.2.2. Given two generated ideals I and J , their closeness $c(I, J)$ is the least rank of a denotable x such that $x \in I$ but $x \notin J$, or vice versa. We call such an x a witness for I and J . As usual, this determines a distance function: $d(I, J) = 2^{-c(I, J)}$.

Theorem 3.2.3. The generated ideals together with the distance function d form a complete metric space.

Proof Theorem 3.1.5 guarantees that $c(I, J) = \infty$ if and only if $I = J$. In addition, it is obvious both that $c(I, J) = c(J, I)$ and that $c(I, K) \geq \min(c(I, J), c(J, K))$. It remains to show that every Cauchy sequence converges. The proof of this is almost identical to the one for arbitrary ideals (Theorem 3

in [6]). Let $\langle I_n \rangle$ be a Cauchy sequence. Let I^o be the set of finite denotable members of almost all I_n . Then the limit of $\langle I_n \rangle$ is the generated ideal I that has the members of I^o as its denotable finite values. \square

As in the complete metric space defined by MacQueen, Plotkin, and Sethi, the notion of contractiveness and the Banach Fixpoint Theorem are the basic tools for proving that recursive type equations have unique solutions in this complete metric space.

Definition 3.2.4. *The function G is contractive if there exists a real number $0 \leq s < 1$ such that, for all $X_1, \dots, X_n, X'_1, \dots, X'_n$, we have*

$$d(G(X_1, \dots, X_n), G(X'_1, \dots, X'_n)) \leq s \cdot \max\{d(X_i, X'_i) \mid 1 \leq i \leq n\}.$$

The function G is nonexpansive when $0 \leq s \leq 1$ instead.

Theorem 3.2.5 (.) Banach *If F is a contractive function on a complete metric space then the equation $X = F(X)$ has a unique solution.*

As could be expected, this unique solution is constructed by choosing an arbitrary X_0 and finding the limit of the sequence $X_0, F(X_0), F(F(X_0)), \dots$

In order to apply the Banach Fixpoint Theorem, it remains to prove that the operations defined above are contractive in the new metric, or at least nonexpansive in the new metric.

Theorem 3.2.6. *The operations \sqcap and \sqcup are nonexpansive. The operations \times , $+$, and \rightarrow are contractive.*

Proof The general structure of the proof is taken from Proposition 6 and Theorem 7 of [6]. (A slightly more direct proof, which never mentions nondenotable elements, is easy to derive, but we prefer to imitate the previous proof in order to make similarities obvious.) Only \rightarrow requires a new argument; we omit discussion of the other operations.

Let I , J , I' , and J' be generated ideals. We want to show that $c(I \rightarrow J, I' \rightarrow J') > \min(c(I, I'), c(J, J'))$. Consider a witness $\llbracket e \rrbracket$ of least rank for the difference between $I \rightarrow J$ and $I' \rightarrow J'$.

Without loss of generality, we assume that $\llbracket ex \rrbracket \in J$ for all $\llbracket x \rrbracket \in I$, but $\llbracket ex \rrbracket \notin J'$ for some $\llbracket x \rrbracket \in I'$. In fact, we can even take this $\llbracket x \rrbracket$ to be finite, since every denotable value is the limit of a chain of finite denotable values (by Proposition 3.1.4), application is continuous, and ideals are closed under approximations and limits.

Since $\llbracket e \rrbracket$ must be a non-bottom function, Proposition 4 of [6] gives us $\llbracket e \rrbracket = \sqcup(a_i \Rightarrow b_i)$ for some finite, non-zero number of a_i and b_i . Let $a = \sqcup\{a_i \mid a_i \sqsubseteq \llbracket x \rrbracket\}$. Then $a \sqsubseteq \llbracket x \rrbracket$ and $\llbracket e \rrbracket(a) = \llbracket ex \rrbracket = \sqcup\{b_i \mid a_i \sqsubseteq \llbracket x \rrbracket\}$. Immediately, a and $\llbracket ex \rrbracket$ have rank smaller than $\llbracket e \rrbracket$.

Two cases should be considered.

If $a \in I$ then we can take $\llbracket ex \rrbracket$ as a witness for J and J' . To show this, we first observe that $\llbracket e \rrbracket(a) \in J$: a is less than the limit of a denotable chain of elements of I , which $\llbracket e \rrbracket$ maps to denotable elements of J because $\llbracket e \rrbracket \in I \rightarrow J$; then $\llbracket e \rrbracket(a) \in J$

by continuity and the definition of ideals. Since $\llbracket e \rrbracket(a) = \llbracket ex \rrbracket$, we can take $\llbracket ex \rrbracket$ as a witness for J and J' .

If $a \notin I$, we would hope to take it as a witness for I and I' . Unfortunately, a may not be denotable! On the other hand, $\llbracket x \rrbracket$ is denotable, but its rank may be too large. We need to find a witness that combines the virtues of $\llbracket x \rrbracket$ (denotability) and a (small rank).

To do this, we reduce the rank of $\llbracket x \rrbracket$ with an application of a projection function: if $r(a) = n$, we let $w = p_n^m(x)$, as described in Lemma 2.5.2.

Obviously, $\llbracket w \rrbracket$ is denotable and its rank is no greater than the rank of a , which is less than the rank of $\llbracket e \rrbracket$. Furthermore, $a \sqsubseteq \llbracket w \rrbracket$. Hence $\llbracket w \rrbracket \notin I$. On the other hand, $\llbracket w \rrbracket \sqsubseteq \llbracket x \rrbracket$ and hence $\llbracket w \rrbracket \in I'$. Thus, $\llbracket w \rrbracket$ is a denotable witness of smaller rank than $\llbracket e \rrbracket$ for I and I' . \square

Theorem 3.2.7. *If $\mathcal{K} \subseteq \mathbf{GIdl}$ and f is contractive (nonexpansive) in its last n arguments then so are $(\forall_{\mathcal{K}} f)$ and $(\exists_{\mathcal{K}} f)$. If f is contractive (nonexpansive) then so is μf .*

Proof The arguments are identical to those for Theorems 8 and 9 in [6]. \square

4. Semantics for Type Expressions

We can use the results on the complete metric space of generated ideals and on the operations in this space to give a semantics to a large class of type expressions. This class is the same one treated in [6] and suffices as the core of the type system for rich programming languages, such as Standard ML.

The generated-ideal semantics of type expressions is more accurate than the usual ones. Despite the features of generated ideals, however, the semantics is not quite faithful. Therefore, we restrict ourselves to subsets of the generated ideals. We define two subsets, the coarse ideals and the abstract ideals; they are rich enough for providing models for recursive polymorphic types. In both cases, the semantics of type expressions is faithful.

4.1. A Model Based on Generated Ideals

Often, one can determine that a function is contractive by examining the symbols used to express it. This motivates the definition of formally contractive type expressions—type expressions for functions that are “obviously” contractive. Roughly, formally contractive type expressions are those built up from **bool**, **int**, contractive operations, and nonexpansive operations followed by contractive operations.

Definition 4.1.1. *The expression σ is a formally contractive type expression in the variable t if one of the following conditions holds:*

- σ has one of the forms **bool**, **int**, t' (with $t' \neq t$), $\sigma_1 \times \sigma_2$, $\sigma_1 + \sigma_2$, or $\sigma_1 \rightarrow \sigma_2$.
- σ has one of the forms $\sigma_1 \cap \sigma_2$ or $\sigma_1 \cup \sigma_2$, with both σ_1 and σ_2 formally contractive in t .

- σ has one of the forms $\forall t'.\sigma_1$, $\exists t'.\sigma_1$, or $\mu t'.\sigma_1$, with either $t = t'$ or σ_1 formally contractive in t .

The definition of formally contractive type expressions naturally leads to the definition of well-formed type expressions:

Definition 4.1.2. *The expression σ is a well-formed type expression if one of the following conditions holds:*

- σ is **bool**, **int**, or t .
- σ has one of the forms $\sigma_1 \times \sigma_2$, $\sigma_1 + \sigma_2$, $\sigma_1 \rightarrow \sigma_2$, $\sigma_1 \cap \sigma_2$, or $\sigma_1 \cup \sigma_2$, with both σ_1 and σ_2 well-formed.
- σ has one of the forms $\forall t.\sigma_1$ or $\exists t.\sigma_1$, with σ_1 well-formed.
- σ has the form $\mu t.\sigma_1$, with σ_1 well-formed and formally contractive in t .

TEXP is the set of well-formed type expressions.

$\llbracket \quad \rrbracket : \text{TEXP} \rightarrow (\text{TVAR} \rightarrow \mathbf{Idl}) \rightarrow \mathbf{Idl}$	
$\llbracket \text{bool} \rrbracket_\rho$	= \mathbf{T}
$\llbracket \text{int} \rrbracket_\rho$	= \mathbf{N}
$\llbracket t \rrbracket_\rho$	= $\rho(t)$
$\llbracket \sigma_1 \cap \sigma_2 \rrbracket_\rho$	= $\llbracket \sigma_1 \rrbracket_\rho \sqcap \llbracket \sigma_2 \rrbracket_\rho$
$\llbracket \sigma_1 \cup \sigma_2 \rrbracket_\rho$	= $\llbracket \sigma_1 \rrbracket_\rho \sqcup \llbracket \sigma_2 \rrbracket_\rho$
$\llbracket \sigma_1 \times \sigma_2 \rrbracket_\rho$	= $\llbracket \sigma_1 \rrbracket_\rho \times \llbracket \sigma_2 \rrbracket_\rho$
$\llbracket \sigma_1 + \sigma_2 \rrbracket_\rho$	= $\llbracket \sigma_1 \rrbracket_\rho + \llbracket \sigma_2 \rrbracket_\rho$
$\llbracket \sigma_1 \rightarrow \sigma_2 \rrbracket_\rho$	= $\llbracket \sigma_1 \rrbracket_\rho \rightarrow \llbracket \sigma_2 \rrbracket_\rho$
$\llbracket \forall t.\sigma \rrbracket_\rho$	= $\forall \mathcal{K}(\lambda I \in \mathbf{Idl}. \llbracket \sigma \rrbracket_{\rho \{t \leftarrow I\}})$
$\llbracket \exists t.\sigma \rrbracket_\rho$	= $\exists \mathcal{K}(\lambda I \in \mathbf{Idl}. \llbracket \sigma \rrbracket_{\rho \{t \leftarrow I\}})$
$\llbracket \mu t.\sigma \rrbracket_\rho$	= $\mu(\lambda I \in \mathbf{Idl}. \llbracket \sigma \rrbracket_{\rho \{t \leftarrow I\}})$

Fig. 2: Definition of the meaning function for type expressions

Imitating MacQueen, Plotkin, and Sethi, we can define a semantics for well-formed type expressions. The meaning function $\llbracket \quad \rrbracket^G$ associates a generated ideal with each well-formed type expression under each type assignment for the free type variables in the expression: if TVAR is the set of type variables, then

$$\llbracket \quad \rrbracket^G : \text{TEXP} \rightarrow (\text{TVAR} \rightarrow \mathbf{GIdl}) \rightarrow \mathbf{GIdl}.$$

We define the semantics in Fig. 2. There, we use **Idl** to refer to the collection of ideals under consideration, in this case **GIdl**, and \mathcal{K} to refer to $\{I \in \mathbf{Idl} \mid \text{wrong} \notin I\}$ (the same semantic templet reappears below, and **Idl** refers to other sets of ideals).

We can prove:

Theorem 4.1.3. *The generated-ideal semantics is well defined.*

Proof The proof is straightforward, given the machinery of the previous section. It resembles the proof of Theorem 10 in [6]. \square

What is the type of the `explode-||-or` function, according to this semantics? Whenever $\llbracket \text{explode-}||\text{-or} \rrbracket$ is given a denotable argument in the generated ideal $\mathbf{T} \times \mathbf{T} \rightarrow \mathbf{T}$, it yields a result in \mathbf{T} . Thus, `explode-||-or` is of type $(\mathbf{bool} \times \mathbf{bool} \rightarrow \mathbf{bool}) \rightarrow \mathbf{bool}$, as we originally suggested.

We may also notice that the expression

```
ignore-||-or = λ x.
  if x(true, btm)
  and x(btm, true)
  and not x(false, false)
  then true
else true
```

behaves identically to `explode-||-or` in all contexts, and, as we would desire, also has type $(\mathbf{bool} \times \mathbf{bool} \rightarrow \mathbf{bool}) \rightarrow \mathbf{bool}$.

These observations indicate that the semantics fits reasonably well with the programming language. The fit is also confirmed by the remarks in Section 5 on the isomorphism between generated ideals and sets of terms.

However, the fit is not as close as it could be. The generated ideals in \mathbf{V} may distinguish two expressions with identical behavior. As a trivial example,

$$\llbracket \text{explode-}||\text{-or} \rrbracket \in \{v \mid v \sqsubseteq \llbracket \text{explode-}||\text{-or} \rrbracket\}$$

while

$$\llbracket \text{ignore-}||\text{-or} \rrbracket \notin \{v \mid v \sqsubseteq \llbracket \text{explode-}||\text{-or} \rrbracket\}.$$

Since a free type variable can take $\{v \mid v \sqsubseteq \llbracket \text{explode-}||\text{-or} \rrbracket\}$ as meaning, the generated-ideal semantics fails to be faithful. (It is an open question whether the generated-ideal semantics is faithful when restricted to type expressions with no free type variables.) In order to guarantee a perfect fit between syntax and semantics, a more finely tuned concept of type is needed.

4.2. Two Faithful Models

In this subsection we consider concepts of type rich enough as a basis to a semantics for recursive polymorphic types, yet coarse enough not to make irrelevant distinctions between expressions that behave identically. The crucial observation is that we do not need all of \mathbf{GIdl} to define a semantics. We restrict our attention to two subsets of \mathbf{GIdl} , the coarse ideals, \mathbf{CIdl} , and the abstract ideals, \mathbf{AIdl} .

The subset \mathbf{CIdl} is close to the minimum needed for a suitable semantics for recursive polymorphic types. For instance, \mathbf{T} is in \mathbf{CIdl} while (we conjecture) $\{\perp, 3\}$ is not; this seems acceptable, since \mathbf{T} is useful as the meaning of `bool`, while $\{\perp, 3\}$ is of no use at all as a programming type.

Definition 4.2.1. *\mathbf{CIdl} is the smallest subset of \mathbf{GIdl} that contains \mathbf{T} and \mathbf{N} , and is closed under binary and infinitary \sqcap and \sqcup , under \times , $+$, and \rightarrow , and under limits of Cauchy sequences.*

The closure conditions on the set of ideals \mathbf{CIdl} are monotone (with respect to set inclusion), hence Definition 4.2.1 is proper.

We can reproduce the semantics of type expressions using coarse ideals instead of generated ideals. We define

$$\llbracket \]^c : TEXP \rightarrow (TVAR \rightarrow \mathbf{CIdl}) \rightarrow \mathbf{CIdl}.$$

The semantics is given by Fig. 2 (with \mathbf{CIdl} playing the role of \mathbf{Idl}). We apply the Banach Fixpoint Theorem as usual, to prove that the semantics is well defined.

Theorem 4.2.2. *The coarse-ideal semantics is well defined.*

Proof Again, the proof is straightforward, given the machinery of the previous section. It resembles the proof of Theorem 10 in [6]. \square

It remains to show that this semantics is faithful. For this purpose, we use \mathbf{AIdl} , the class of abstract generated ideals—for short, abstract ideals. (Abstract sets were defined in Subsection 2.4.)

Definition 4.2.3. *\mathbf{AIdl} is the set of abstract generated ideals.*

Abstract ideals serve in the study of coarse ideals, and also give rise to a natural semantics of type expressions, of independent interest. It turns out that \mathbf{CIdl} is a subset of \mathbf{AIdl} . Hence, a semantics based on \mathbf{CIdl} benefits from the features of \mathbf{AIdl} .

To guarantee that \mathbf{CIdl} and \mathbf{AIdl} both suffice as the basis for a faithful semantics, we prove that they share each other's features. \mathbf{AIdl} enjoys the same closure properties as \mathbf{CIdl} . Hence \mathbf{CIdl} , the smallest set with these closure properties, must be a subset of \mathbf{AIdl} . In other words, all coarse ideals are abstract (and probably not vice versa: $\{\perp, 3\}$ is abstract but apparently not coarse).

Lemma 4.2.4. *\mathbf{AIdl} contains \mathbf{T} and \mathbf{N} , and is closed under binary and infinitary \sqcap and \sqcup , under \times , $+$, and \rightarrow , and under limits of Cauchy sequences. If f is a function from \mathbf{AIdl}^{n+1} to \mathbf{AIdl} then so are $(\forall_K f)$, $(\exists_K f)$, and, provided f is contractive, (μf) .*

Proof We consider each of the closure conditions in turn. We assume that the arguments (if any) to an operation are abstract, to prove that the result is abstract as well. The cases vary in difficulty.

- The proposition is trivial for \mathbf{T} . Suppose that $e' \sqsubset e$, that is, e' yields \perp in all contexts where e does. This must be true, in particular, for the context `if (_ or not _) then btm else wrong`, hence if $\llbracket e \rrbracket \in \mathbf{T}$ then $\llbracket e' \rrbracket \in \mathbf{T}$.
- The proposition is also trivial for \mathbf{N} . Suppose that $e' \sqsubset e$, that is, e' yields \perp in all contexts where e does. This must be true, in particular, for the context `if (z(_) or not z(_)) then btm else wrong`, and hence if $\llbracket e \rrbracket \in \mathbf{N}$ then $\llbracket e' \rrbracket \in \mathbf{N}$.

- The cases of binary \sqcap and \sqcup , infinitary \sqcap , and \times and $+$ are solved with simple applications of the hypothesis. (When considering binary \sqcap and \sqcup and infinitary \sqcap , one should recall that they coincide with the corresponding set-theoretic operations on denotable elements.)
- For \rightarrow , suppose that I and J are abstract, and that $e' \sqsubset e$ and $\llbracket e \rrbracket \in (I \rightarrow J)$. The definition of \sqsubset yields that $e'(a) \sqsubset e(a)$, for all a . Furthermore, if $\llbracket a \rrbracket \in I$ then $\llbracket e(a) \rrbracket \in J$ and, since J is abstract, $\llbracket e'(a) \rrbracket \in J$. By the definition of \rightarrow , it follows that $\llbracket e' \rrbracket \in (I \rightarrow J)$, as we wanted to show.
- The case for infinitary \sqcup requires a slightly more elaborate argument. Assume that $e' \sqsubset e$. We consider chains $\langle \llbracket p_n^n(e) \rrbracket \rangle$ and $\langle \llbracket p_n^n(e') \rrbracket \rangle$ of finite denotable elements with limits $\llbracket e \rrbracket$ and $\llbracket e' \rrbracket$, respectively (see Proposition 3.1.4). The definition of \sqsubset yields $p_n^n(e') \sqsubset p_n^n(e)$ for all n . Since each of the ideals coming into the union is abstract by the hypothesis, for all n , each ideal must contain $\llbracket p_n^n(e') \rrbracket$ if it contains $\llbracket p_n^n(e) \rrbracket$. If $\llbracket e \rrbracket$ is a limit point of the union of the ideals then $\llbracket p_n^n(e) \rrbracket$ is in the union, for all n . Then $\llbracket p_n^n(e') \rrbracket$ is in the union too, for all n , and finally $\llbracket e' \rrbracket$ is a limit point of the union as well.
- Closure under limits of Cauchy sequences requires an argument similar to the previous one. Assume that $e' \sqsubset e$. We consider chains $\langle \llbracket p_n^n(e) \rrbracket \rangle$ and $\langle \llbracket p_n^n(e') \rrbracket \rangle$ of finite denotable elements with limits $\llbracket e \rrbracket$ and $\llbracket e' \rrbracket$, respectively (see Proposition 3.1.4). The definition of \sqsubset yields $p_n^n(e') \sqsubset p_n^n(e)$ for all n . Since each of the ideals in the sequence is abstract by the hypothesis, for all n , each ideal must contain $\llbracket p_n^n(e') \rrbracket$ if it contains $\llbracket p_n^n(e) \rrbracket$. If the limit contains $\llbracket e \rrbracket$ then it must contain $\llbracket p_n^n(e) \rrbracket$. Therefore, since the sequence is a Cauchy sequence, all sufficiently late terms in the sequence contain $\llbracket p_n^n(e) \rrbracket$ (only finitely many terms can differ from the limit on elements of rank n). In turn, it follows that all sufficiently late terms in the sequence must also contain $\llbracket p_n^n(e') \rrbracket$, and then the limit of the sequence must contain $\llbracket p_n^n(e') \rrbracket$. Finally, the limit of the sequence must also contain $\llbracket e' \rrbracket$.
- Similarly, if f is contractive and maps abstract ideals to abstract ideals then so do $(\forall_K f)$ and $(\exists_K f)$: this follows immediately from their definition, together with the closure of **AIdl** under infinitary \sqcap and \sqcup .
- If f maps abstract ideals to abstract ideals then so does (μf) : this follows immediately from the definition of μ , together with the closure of **AIdl** under limits of Cauchy sequences (since fixpoints are obtained as limits of Cauchy sequences). \square

To finish, we define an abstract-ideal semantics and prove that both the coarse-ideal semantics and the abstract-ideal semantics are faithful.

We define the abstract-ideal semantics using the templet of Fig. 2 (with **AIdl** playing the role of **Idl**):

$$\llbracket \]^A : TEXP \rightarrow (TVar \rightarrow \mathbf{AIdl}) \rightarrow \mathbf{AIdl}.$$

Theorem 4.2.5. *The abstract-ideal semantics is well defined.*

Proof First we should point out that **AIdl** is closed under the necessary type operations, as shown in Lemma 4.2.4. Then the argument is the usual one. \square

Theorem 4.2.6. *Both the coarse-ideal semantics and the abstract-ideal semantics are faithful.*

Proof Both semantics map type expressions to elements of **AIdl**, which are all abstract by definition. (For the coarse-ideal semantics, this follows from Lemma 4.2.4.) \square

This result is fairly robust with respect to changes in the interpretation of terms. Consider an interpretation more abstract than ours, that is, an interpretation that identifies the meanings of more programs. If the interpretation is sound, some meanings of programs will be identified within abstract sets, but not across the boundaries of abstract sets. Therefore, we can map the abstract ideals of our semantics for type expressions to ideals over the more abstract model, and obtain the corresponding faithful semantics.

5. Syntactic Solutions

While generated ideals have close connections with the world of expressions, they still belong to the world of values. In some situations, a more syntactic model, where types are certain sets of expressions, may seem desirable. (In fact, this work originated in just such a situation; the preference for a syntactic model was simply a matter of taste.)

In this section we take advantage of the properties of generated ideals to discuss models of recursive polymorphic types where sets of expressions are used instead of sets of values. Finally, we speculate on a totally syntactic model.

5.1. Term Ideals

A simple way to obtain a model where types are sets of expressions is to induce these sets from the abstract ideals.

Definition 5.1.1. *If $I \subseteq \mathbf{V}$ then the representation of I is the set of all closed expressions e such that $\llbracket e \rrbracket \in I$.*

Definition 5.1.2. *I is a term ideal if it is the representation of some abstract ideal. **TIdl** is the set of term ideals, and R is the one-to-one function that maps abstract ideals to their representations.*

Immediately, we can define a semantics of type expression using term ideals,

$$\llbracket \]^T : \mathbf{TExp} \rightarrow (\mathbf{TVAR} \rightarrow \mathbf{TIdl}) \rightarrow \mathbf{TIdl},$$

thus

$$\llbracket \sigma \rrbracket_\rho^T = R(\llbracket \sigma \rrbracket_{R^{-1} \circ \rho}^A).$$

Actually, the term ideals can be characterized directly, rather naturally.

Theorem 5.1.3. *I is a term ideal if and only if*

- $I \neq \emptyset$; and
- for all sequences $e_1 \sqsubset e_2 \sqsubset \dots$ of elements of I , for all e , if for all C such that $e \models C$ there exists i such that $e_i \models C$, then $e \in I$.

Proof Assume that I is a term ideal, that $e_1 \sqsubset e_2 \sqsubset \dots$ is a sequence of elements of I , and that for all C if $e \models C$ then $e_i \models C$ for some i . We wish to show that $e \in I$. Trivially, it suffices to argue that each $p_n^n(e) \in I$. As the range of $\llbracket p_n^n \rrbracket$ is finite, the sequence $\langle p_n^n(e_i) \rangle$ contains infinitely many times the same element (up to semantic equality). Let $p_n^n(e_m)$ be such an element. Suppose that $p_n^n(e) \models C$. Then there exists k such that $p_n^n(e_k) \models C$, and the assumption that $e_1 \sqsubset e_2 \sqsubset \dots$ yields $p_n^n(e_j) \models C$ for all $j \geq k$. In particular, $p_n^n(e_m) \models C$, since $p_n^n(e_m)$ arises infinitely often in the sequence $\langle p_n^n(e_i) \rangle$. Then we have $p_n^n(e) \sqsubset p_n^n(e_m) \sqsubset e_m \in I$ and, therefore, $p_n^n(e) \in I$.

Conversely, suppose that I is a non-empty set that satisfies the given closure condition. We have to find an abstract ideal J such that $I = R(J)$. Let

$$J = Id(\{\llbracket e \rrbracket \mid e \in I\}).$$

By Theorem 3.1.3 this is a generated ideal.

Evidently, we have $I \subseteq R(J)$. In order to show the converse, suppose that $\llbracket e' \rrbracket$ is in $R(J)$. Then for every n , $\llbracket p_n^n(e') \rrbracket \sqsubseteq \llbracket e \rrbracket$ for some $e \in I$. The closure condition, applied to the constant sequence $e \sqsubset e \sqsubset \dots$, yields $p_n^n(e') \in I$. Finally, we consider the sequence $p_1^1(e') \sqsubset p_2^2(e') \sqsubset \dots$, and conclude that $e' \in I$.

As for abstraction, suppose $e' \sqsubset e$ and $\llbracket e \rrbracket \in J$; again, we consider the constant sequence $e \sqsubset e \sqsubset \dots$, to derive that $e' \in I$ and, therefore, that $\llbracket e' \rrbracket \in J$. \square

The proof of Theorem 5.1.3 suggests a useful formula for the inverse of R :

$$R^{-1}(I) = Id(\{\llbracket e \rrbracket \mid e \in I\}).$$

The operations on term ideals have direct definitions, and for example we can write

$$I \sqcap J = I \cap J$$

and

$$I \rightarrow J = \{e \mid \llbracket e \rrbracket \in (\mathbf{V} \rightarrow \mathbf{V}) \text{ and } \forall e' \in I. e(e') \in J\}.$$

These definitions correspond with those for **AIdl**, in the sense that, for example

$$I \sqcap J = R(R^{-1}(I) \cap R^{-1}(J))$$

and

$$I \rightarrow J = R(R^{-1}(I) \rightarrow R^{-1}(J)).$$

With their aid we can easily cast the definition of $\llbracket \cdot \rrbracket^T$ in a homomorphic form, as in Fig. 2.

5.2. Operational Ideals

We have not defined a reduction relation for expressions in the programming language, or for that matter any kind of operational semantics for the programming language. To finish, we speculate on an alternative, even more syntactic development of our theory. This development requires the use of a well-behaved reduction relation. (However, we do not propose a particular reduction relation.)

We imagine that we have distinguished a set of “canonical” closed expressions. For example, the canonical boolean expressions are `true` and `false`. Similarly, some set F of canonical expressions are distinguished as functions. We write $e \Rightarrow e'$ for “the closed expression e reduces to the canonical expression e' .”

An operational definition for T seems reasonable in this setting:

$$T = \{e \mid \text{if } e \Rightarrow e' \text{ then } e' \in \{\text{true}, \text{false}\}\}.$$

We may define function types similarly:

$$I \rightarrow J = \{e \mid \text{if } e \Rightarrow e' \text{ then } e' \in F \text{ and if } a \in I \text{ then } ea \in J\}.$$

For suitable notions of reduction, these definitions should coincide with the previous ones. Proving that these definitions do coincide, however, may pose a hard adequacy problem; see [4, 9] for a discussion of adequacy. For instance, one has to show that $[e] \in T$ if and only if $e \Rightarrow e'$ implies $e' \in \{\text{true}, \text{false}\}$, and similarly that $[e] \in (V \rightarrow V)$ if and only if $e \Rightarrow e'$ implies $e' \in F$.

In the end, we obtain a class of “operational ideals.” This approach, though laborious, provides totally syntactic solutions to recursive type equations.

6. Conclusions

Recursive polymorphic types can be modeled with various degrees of accuracy. MacQueen, Plotkin, and Sethi have defined a pure ideal semantics, with no reference to the syntax of the programming language considered; some type-inference rules for the language are semantically sound, but the correspondence between syntax and semantics goes not much further. We have introduced some simple syntactic notions into our ideal semantics, obtaining first the generated ideals and then the coarse ideals and the abstract ideals. The coarse-ideal semantics and the abstract-ideal semantics are slightly more complex than the original ideal semantics, but as accurate as possible, that is, faithful.

It remains unclear how to extend our methods to other type systems, for example to systems with explicit polymorphism. It seems clear, however, that faithfulness is a generally relevant issue in the semantics of type systems.

Acknowledgements

We are grateful to Luca Cardelli, Stavros Cosmadakis, and an anonymous referee for useful suggestions, and to Cynthia Hibbard for editorial help.

References

- [1] Robin Milner. Fully abstract models of typed λ -calculi. *Theoretical Computer Science*, 4:1–22, 1977.
- [2] Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–256, 1977.
- [3] Allen Stoughton. *Fully Abstract Models of Programming Languages*. Pitman/Wiley, 1988. Research Notes in Theoretical Computer Science.
- [4] Albert Meyer. Semantical paradigms: notes for an invited lecture. In *Proceedings of the Third Annual IEEE Symposium on Logic in Computer Science*, pages 236–253, July 1988. With two appendices by Stavros S. Cosmadakis.
- [5] Robin Milner. A proposal for Standard ML. In *Proceedings of the 1984 ACM Symposium on Lisp and Functional Programming*, pages 184–197, August 1984.
- [6] David MacQueen, Gordon Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71:95–130, 1986.
- [7] Henk P. Barendregt. *The Lambda Calculus*. North Holland, Revised edition, 1984.
- [8] Stefan Banach. Sur les opérations dans les ensembles abstraits et leurs applications aux équations intégrales. *Fund. Math.*, 3:7–33, 1922.
- [9] Stavros S. Cosmadakis. Computing with recursive types. In *Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science*, pages 24–38, June 1989.

On Traced Monoidal Closed Categories

MASAHIKO HASEGAWA

Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502, Japan

Email: hassei@kurims.kyoto-u.ac.jp

28 November 2007

The structure theorem of Joyal, Street and Verity says that every traced monoidal category \mathcal{C} arises as a monoidal full subcategory of the tortile monoidal category $\mathbf{Int} \mathcal{C}$. In this paper we focus on a simple observation that a traced monoidal category \mathcal{C} is *closed* if and only if the canonical inclusion from \mathcal{C} into $\mathbf{Int} \mathcal{C}$ has a right adjoint. Thus, every traced monoidal closed category arises as a monoidal co-reflexive full subcategory of a tortile monoidal category. From this, we derive a series of facts on traced models of linear logic, and some on models of fixed-point computation.

1. Introduction

In (Joyal *et al.* 1996), Joyal, Street and Verity introduced the notion of traced monoidal categories. They showed that every traced monoidal category \mathcal{C} fully faithfully embeds in a tortile monoidal category $\mathbf{Int} \mathcal{C}$, and that this Int-construction gives a left biadjoint of the forgetful 2-functor from the 2-category of tortile monoidal categories to that of traced monoidal categories. This remarkable result attracted much attention from theoretical computer scientists, particularly in connection with linear logic (Girard 1987) and Geometry of Interaction (Girard 1989; Abramsky and Jagadeesan 1994; Abramsky 1996; Haghverdi 2000; Abramsky *et al.*; Haghverdi and Scott 2006), where the special case of traced symmetric monoidal categories and compact closed categories (Kelly and Laplaza 1980) is of interest.

In this paper we shall see that the monoidal closed structure can be tied with the Int-construction in an unexpected manner. Namely, we show the following result (Theorem 4.1):

Theorem. A traced monoidal category \mathcal{C} is closed if and only if the embedding from \mathcal{C} into $\mathbf{Int} \mathcal{C}$ has a right adjoint.

Despite its simplicity, to the best of our knowledge, this fact has not been pointed out in the literature. Perhaps this is partly because the Int-construction works too nicely: tortile monoidal categories are closed, therefore every traced monoidal category embeds into a closed one just via the Int-construction. So it seems that for this reason we did not feel that traced monoidal closed categories were themselves interesting (with (Coccia *et al.* 2002) as an exception where traced monoidal closed categories with extra structure

are used for modelling higher-order cyclic shared structures). However, tortile structure (or compact closed structure in the symmetric case) describes just a very special kind of closedness, and not every traced monoidal closed category is tortile. Our result says that the relation between tortile structure and general closed structure is still in harmony: every traced monoidal closed category arises as a monoidal co-reflexive full subcategory of a tortile monoidal category. The embedding from \mathcal{C} to $\mathbf{Int}\mathcal{C}$ may not preserve the closed structure, but the closed structure on \mathcal{C} is determined by the closed structure of $\mathbf{Int}\mathcal{C}$ and the co-reflection.

The author noticed this result in July 2005, after learning from Paul-André Melliès about the traced monoidal closed category of negative Conway games (Melliès 2004). It took the author a few months to realize that it did have several applications on models of linear logic and also models of fixed-point computation. After that, Shin-ya Katsumata and Susumu Nishimura discovered a concrete example of this in the study of program transformations in 2006 (Katsumata and Nishimura 2006), and then Katsumata gave a striking application in the theory of attribute grammars in 2007 (Katsumata 2007). These discoveries prompted the author to write down the result so that it is hopefully accessible to theoretical computer scientists with mild background in category theory.

To this end, instead of just stating and proving the main result (which would need only a few pages, see Section 4), we shall include all needed definitions and results (but without proofs) on traced monoidal categories, Int-construction, monoidal closed categories and monoidal adjunctions in Section 2 and 3, making the paper largely self-contained. Section 5 is devoted for applications in models of linear logic and fixed-point computation; for the latter the author's old work on recursion created from cyclic sharing is recast as linear fixed-point operators in traced models of intuitionistic linear logic. In addition, we shall make use of the graphical presentation for monoidal categories and monoidal functors (Joyal and Street 1991; Cockett and Seely 1999; Melliès 2006) (which goes back to Penrose's diagrams for calculating with tensors (Penrose and Rindler 1984)), which is intuitively helpful and technically convenient.

While most work in computer science focus on the symmetric case (traced symmetric monoidal categories and compact closed categories), in this paper, whenever possible, we consider general traced balanced (braided) monoidal categories and tortile monoidal categories, following the original development by Joyal, Street and Verity; most of our results are valid for this generality — and also we expect (rather optimistically) that non-symmetric situations will become useful in future developments in computer science.

2. Preliminaries

2.1. Monoidal Categories

A *monoidal category* (Mac Lane 1971) (or *tensor category* (Joyal and Street 1993)) $\mathcal{C} = (\mathcal{C}, \otimes, I, a, l, r)$ consists of a category \mathcal{C} , a functor $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$, an object $I \in \mathcal{C}$ and natural isomorphisms $a_{A,B,C} : (A \otimes B) \otimes C \xrightarrow{\sim} A \otimes (B \otimes C)$, $l_A : I \otimes A \xrightarrow{\sim} A$ and

$r_A : A \otimes I \xrightarrow{\sim} A$ such that the following two diagrams commute for all A, B, C and D :

$$\begin{array}{ccc}
((A \otimes B) \otimes C) \otimes D & \xrightarrow{a} & (A \otimes B) \otimes (C \otimes D) \\
\downarrow a \otimes D & & \downarrow a \\
(A \otimes (B \otimes C)) \otimes D & \xrightarrow{a} & A \otimes ((B \otimes C) \otimes D) \xrightarrow{A \otimes a} A \otimes (B \otimes (C \otimes D))
\end{array}$$

$$\begin{array}{ccc}
(A \otimes I) \otimes B & \xrightarrow{a} & A \otimes (I \otimes B) \\
\searrow r \otimes B & & \swarrow A \otimes l \\
& A \otimes B &
\end{array}$$

Mac Lane's coherence theorem (Mac Lane 1971; Joyal and Street 1993) states that all diagrams built from a, l and r commute. It follows that, in principle, most results stated for the *strict* monoidal categories (where a, l, r are the identity arrows and $(A \otimes B) \otimes C$ is identified with $A \otimes (B \otimes C)$, and similarly for tensor unit) can be reformulated and proved without strictness. In many places of this paper, for ease of presentation, we avoid putting brackets on tensor products when rigour is guaranteed by the coherence theorem.

A *braiding* is a natural isomorphism $c_{A,B} : A \otimes B \xrightarrow{\sim} B \otimes A$ such that both c and c^{-1} satisfy the following “bilinearity” diagrams (the case for c^{-1} is omitted):

$$\begin{array}{ccccc}
(A \otimes B) \otimes C & \xrightarrow{a} & A \otimes (B \otimes C) & \xrightarrow{c} & (B \otimes C) \otimes A \\
\downarrow c \otimes C & & & & \downarrow a \\
(B \otimes A) \otimes C & \xrightarrow{a} & B \otimes (A \otimes C) & \xrightarrow{B \otimes c} & B \otimes (C \otimes A)
\end{array}$$

A *symmetry* is a braiding such that $c_{A,B} = c_{B,A}^{-1}$. A *braided/symmetric monoidal category* is a monoidal category equipped with a braiding/symmetry.

A *twist* or a *balance* for a braided monoidal category is a natural isomorphism $\theta_A : A \xrightarrow{\sim} A$ such that $\theta_I = id_I$ and $\theta_{A \otimes B} = c_{B,A} \circ (\theta_B \otimes \theta_A) \circ c_{A,B}$ hold. A *balanced monoidal category* is a braided monoidal category with a twist. Note that a symmetric monoidal category is a balanced monoidal category with $\theta_A = id_A$ for every A .

2.2. Monoidal Functors and Natural Transformations

For monoidal categories $\mathcal{C} = (\mathcal{C}, \otimes, I, a, l, r)$ and $\mathcal{C}' = (\mathcal{C}', \otimes', I', a', l', r')$, a *monoidal functor* from \mathcal{C} to \mathcal{C}' is a tuple (F, m, m_I) where F is a functor from \mathcal{C} to \mathcal{C}' , m is a natural transformation from $F(-) \otimes' F(=)$ to $F(- \otimes =)$ and $m_I : I' \rightarrow FI$ is an arrow

in \mathcal{C}' , satisfying the coherence conditions below.

$$\begin{array}{ccccc}
 (FA \otimes' FB) \otimes' FC & \xrightarrow{m \otimes' FC} & F(A \otimes B) \otimes' FC & \xrightarrow{m} & F((A \otimes B) \otimes C) \\
 \downarrow a' & & & & \downarrow Fa \\
 FA \otimes' (FB \otimes' FC) & \xrightarrow{FA \otimes' m} & FA \otimes' F(B \otimes C) & \xrightarrow{m} & F(A \otimes (B \otimes C)) \\
 I' \otimes' FA & \xrightarrow{l'} & FA & & FA \otimes' I' \xrightarrow{r'} FA \\
 \downarrow m_I \otimes FA & & \uparrow Fl & & \downarrow FA \otimes m_I \\
 FI \otimes' FA & \xrightarrow{m} & F(I \otimes A) & & FA \otimes' FI \xrightarrow{m} F(A \otimes I) \\
 & & & & \uparrow Fr
 \end{array}$$

It is called *strong* if $m_{A,B}$ and m_I are all isomorphisms.

A *balanced monoidal functor* from a balanced \mathcal{C} to another \mathcal{C}' is a monoidal functor (F, m, m_I) which additionally satisfies the following condition

$$\begin{array}{ccc}
 FA \otimes' FB & \xrightarrow{c'} & FB \otimes' FA \\
 \downarrow m & & \downarrow m \\
 F(A \otimes B) & \xrightarrow{Fc} & F(B \otimes A)
 \end{array}$$

and $F(\theta_A) = \theta_{FA}$.

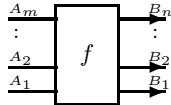
For monoidal functors $(F, m, m_I), (G, n, n_I)$ with the same source and target monoidal categories, a *monoidal natural transformation* from (F, m, m_I) to (G, n, n_I) is a natural transformation $\varphi : F \rightarrow G$ such that the following diagrams commute:

$$\begin{array}{ccc}
 FA \otimes' FB & \xrightarrow{m} & F(A \otimes B) \\
 \downarrow \varphi \otimes' \varphi & & \downarrow \varphi \\
 GA \otimes' GB & \xrightarrow{n} & G(A \otimes B)
 \end{array}
 \quad
 \begin{array}{ccc}
 & I' & \\
 m_I \swarrow & \searrow n_I & \\
 FI & \xrightarrow{\varphi} & GI
 \end{array}$$

A (*balanced/symmetric*) *monoidal adjunction* between (*balanced/symmetric*) monoidal categories is an adjunction in which both of the functors are (*balanced/symmetric*) monoidal and the unit and counit are monoidal natural transformations.

2.3. Geometry of Monoidal Categories

In this paper we make use of the graphical language for monoidal categories, known as string diagrams or Penrose diagrams, developped by Joyal and Street (Joyal and Street 1991). A morphism $f : A_1 \otimes A_2 \otimes \dots \otimes A_m \rightarrow B_1 \otimes B_2 \otimes \dots \otimes B_n$ in a monoidal category can be drawn as (from left to right)



Morphisms can be composed, either sequentially —

$$f : X \rightarrow Y, g : Y \rightarrow Z \quad \Rightarrow \quad g \circ f : X \rightarrow Z$$

or in parallel:

$$f : A \rightarrow B, g : C \rightarrow D \quad \Rightarrow \quad f \otimes g : A \otimes C \rightarrow B \otimes D$$

Braids are expressed by crossing: $c = \overbrace{\text{Diagram of two strands crossing}}^c$, $c^{-1} = \overbrace{\text{Diagram of two strands crossing}}^{c^{-1}}$. Twists are drawn as $\theta = \overbrace{\text{Diagram of a twist}}^\theta$ and $\theta^{-1} = \overbrace{\text{Diagram of a twist}}^{\theta^{-1}}$.

Remark 2.1. Links in these pictures should be regarded as “ribbons” or “framed tangles”, as stressed in (Joyal and Street 1991; Joyal and Street 1993; Shum 1994). Our notation for twists is intended to be a reasonable alternative for the ribbon twisting notation used in the literature.

Monoidal functors and monoidal natural transformations also allow concise graphical presentations, as demonstrated by Cockett and Seely (Cockett and Seely 1999) and Melliès (Melliès 2006). Consider a monoidal functor $F = (F, m, m_I) : \mathcal{C} \rightarrow \mathcal{D}$. Given $f : A \otimes B \rightarrow C$ in \mathcal{C} , we may draw a picture with “box” (here with the dark shadow, should not be confused with the square around f)

which represents $FA \otimes FB \xrightarrow{m_{A,B}} F(A \otimes B) \xrightarrow{Ff} FC$. The dark area is in \mathcal{C} , whereas the white area is in \mathcal{D} . Similarly, given $a : I \rightarrow A$, the picture

represents $I \xrightarrow{m_I} FI \xrightarrow{F a} FA$. The three coherence conditions of monoidal functors ensure that this notation works well for general $f : A_1 \otimes \dots \otimes A_n \rightarrow B$, where the grouping of A_i 's does not matter. In addition, if F is strong monoidal, we can do the same for tensors in the codomain. It is a pleasant exercise to write down the coherence conditions for braids and monoidal natural transformations using this box notation. For instance, one of the diagrams for monoidal natural transformations can be shown as follows.

See (Cockett and Seely 1999) and (Melliès 2006) for further details and examples.

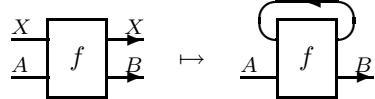
3. Traced Monoidal Categories

3.1. Traced Monoidal Categories

We present a slightly simplified definition of traced monoidal categories, where trace is defined in an object-wise manner. Such a theory of object-wise trace has been developed by Blute, Cockett and Seely for linearly distributive categories (Blute *et al.* 2000); Milner also gave a similar axiomatization for his reflexive action calculi (Milner 1994).

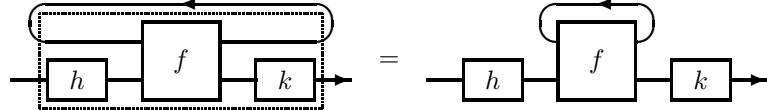
A *traced monoidal category* (Joyal *et al.* 1996) is a balanced monoidal category \mathcal{C} equipped with a family of functions, called *trace operator*

$$Tr_{A,B}^X : \mathcal{C}(A \otimes X, B \otimes X) \longrightarrow \mathcal{C}(A, B)$$



subject to the following four coherence axioms.

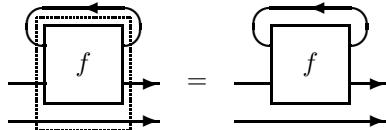
$$\text{Tightening (Naturality)} \quad Tr_{A',B'}^X((k \otimes id_X) \circ f \circ (h \otimes id_X)) = k \circ Tr_{A,B}^X(f) \circ h$$



$$\text{Yanking} \quad Tr_{X,X}^X(c_{X,X}) \circ \theta_X^{-1} = id_X = Tr_{X,X}^X(c_{X,X}^{-1}) \circ \theta_X$$

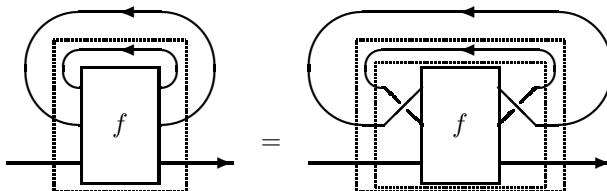


$$\text{Superposing} \quad Tr_{C \otimes A, C \otimes B}^X(id_C \otimes f) = id_C \otimes Tr_{A,B}^X(f)$$



Exchange

$$Tr_{A,B}^X(Tr_{A \otimes X, B \otimes X}^Y(f)) = Tr_{A,B}^Y(Tr_{A \otimes Y, B \otimes Y}^X((id_B \otimes c_{Y,X}) \circ f \circ (id_A \otimes c_{Y,X}^{-1})))$$



Readers familiar with the paper by Joyal, Street and Verity (Joyal *et al.* 1996) should find no difficulty in seeing that these are all derivable from the original axiomatization. Conversely, original axioms are derivable from our axioms; we shall give a slightly non-trivial derivation of the *Sliding* and *Vanishing for tensor* in an appendix. The remaining *Vanishing for unit* is in fact redundant in the original axiomatisation (as demonstrated in an appendix), so our axioms are equivalent to the original axioms in *ibid.*

3.2. Tortile Monoidal Categories

A *tortile monoidal category* (Shum 1994) (also *ribbon category* (Yetter 2001)) is a balanced monoidal category with an object A^* for each object A , unit $\eta_A : I \rightarrow A \otimes A^*$ and counit $\varepsilon_A : A^* \otimes A \rightarrow I$ such that each of

$$\begin{aligned} A &\xrightarrow{l_A^{-1}} I \otimes A \xrightarrow{\eta_A \otimes id_A} (A \otimes A^*) \otimes A \xrightarrow{a_{A, A^*, A}} A \otimes (A^* \otimes A) \xrightarrow{id_A \otimes \varepsilon_A} A \otimes I \xrightarrow{r_A} A \\ A^* &\xrightarrow{r_{A^*}^{-1}} A^* \otimes I \xrightarrow{id_{A^*} \otimes \eta_A} A^* \otimes (A \otimes A^*) \xrightarrow{a_{A^*, A, A^*}^{-1}} (A^* \otimes A) \otimes A^* \xrightarrow{\varepsilon_A \otimes id_A} I \otimes A^* \xrightarrow{l_{A^*}} A^* \end{aligned}$$

is the identity and moreover $\theta_A^* = \theta_{A^*}$ holds, where, for $f : A \rightarrow B$, $f^* : B^* \rightarrow A^*$ is given by (omitting l , r and a)

$$B^* \xrightarrow{id_{B^*} \otimes \eta_A} B^* \otimes A \otimes A^* \xrightarrow{id_{B^*} \otimes f \otimes id_{A^*}} B^* \otimes B \otimes A^* \xrightarrow{\varepsilon_B \otimes id_{A^*}} A^*.$$

It follows that $(-)^*$ extends to a contravariant equivalence, $A^{**} \simeq A$, and the functor $(-)\otimes A$ is left adjoint to $(-)\otimes A^*$ with unit $id_X \otimes \eta_A : X \rightarrow X \otimes A \otimes A^*$ and counit $id_X \otimes \varepsilon_A : X \otimes A^* \otimes A \rightarrow X$. Note that tortile symmetric monoidal categories (in which braiding is a symmetry and twist is the identity) are familiar compact closed categories (Kelly and Laplaza 1980). The unit and counit in tortile monoidal categories can be drawn as  and  respectively. With them, three axioms are expressed as follows.

$$\begin{array}{ccc} \text{Diagram of a curved arrow pointing right, representing the unit } \eta. & = & \text{Diagram of a curved arrow pointing left, representing the counit } \varepsilon. \end{array}$$

The importance of tortile monoidal categories in knot theory comes from the following result:

Theorem 3.1. (Shum 1994) The tortile monoidal category freely generated by a single object is equivalent to the category of framed tangles.

Therefore, tortile monoidal categories give rise to invariants (or models) for tangles, just in the same sense that cartesian closed categories give rise to models of the simply typed lambda calculi (Lambek and Scott 1986).

Any tortile monoidal category has a unique trace (called *canonical trace* (Joyal *et al.* 1996)), hence is also a traced monoidal category. The canonical trace is given by combining η , ε , c and θ :

$$Tr_{A,B}^X f = (id_B \otimes (\varepsilon_X \circ (id_{X^*} \otimes \theta_X) \circ c_{X,X^*})) \circ (f \otimes id_{X^*}) \circ (id_A \otimes \eta_X)$$

It follows that a monoidal full subcategory of a tortile monoidal category is traced.

3.3. The Int-Construction

In fact, every traced monoidal category arises in this way: given a traced monoidal category \mathcal{C} , we can construct a tortile monoidal category $\mathbf{Int} \mathcal{C}$ to which \mathcal{C} fully faithfully embeds, via the Int-construction of Joyal, Street and Verity — an abstract version of “Geometry of Interaction” of Girard/Abramsky. Below we briefly recall its ingredients.

Objects of $\mathbf{Int} \mathcal{C}$ are pairs of objects of \mathcal{C} . An arrow from (A^+, A^-) to (B^+, B^-) in $\mathbf{Int} \mathcal{C}$ is an arrow from $A^+ \otimes B^-$ to $B^+ \otimes A^-$ in \mathcal{C} . The identity on (A^+, A^-) is $id_{A^+} \otimes \theta_{A^-}^{-1} \in \mathcal{C}(A^+ \otimes A^-, A^+ \otimes A^-)$. The composition of $f \in \mathbf{Int} \mathcal{C}((A^+, A^-), (B^+, B^-)) = \mathcal{C}(A^+ \otimes B^-, B^+ \otimes A^-)$ and $g \in \mathbf{Int} \mathcal{C}((B^+, B^-), (C^+, C^-)) = \mathcal{C}(B^+ \otimes C^-, C^+ \otimes B^-)$ is given by

Next we look at the monoidal structure. On objects, we define tensor and unit by $(A^+, A^-) \otimes (B^+, B^-) = (A^+ \otimes B^+, B^- \otimes A^-)$ and $I = (I, I)$. On arrows, for $f_1 : (A_1^+, A_1^-) \rightarrow (B_1^+, B_1^-)$ and $f_2 : (A_2^+, A_2^-) \rightarrow (B_2^+, B_2^-)$, define $f_1 \otimes f_2$ by

Braids and twists in $\mathbf{Int} \mathcal{C}$ are not quite obvious:

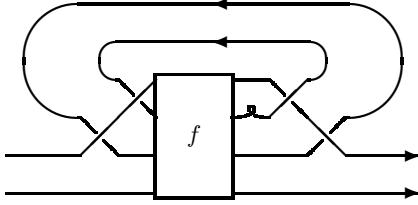
(It is an interesting exercise to write down c^{-1} and θ^{-1} explicitly.)

Finally we describe the duality, which is not so hard: $(A^+, A^-)^* = (A^-, A^+)$. The unit $\eta_{(A^+, A^-)} : I \rightarrow (A^+, A^-) \otimes (A^+, A^-)^*$ is given by $id_{A^+} \otimes \theta_{A^-}^{-1}$. The counit $\varepsilon_{(A^+, A^-)} : (A^+, A^-)^* \otimes (A^+, A^-) \rightarrow I$ is $id_{A^-} \otimes \theta_{A^+}$. We can extend $(-)^*$ to be a contravariant equivalence on $\mathbf{Int} \mathcal{C}$ — on arrows $f : (A^+, A^-) \rightarrow (B^+, B^-)$ define $f^* : (B^+, B^-)^* \rightarrow (A^+, A^-)^*$ as

Theorem 3.2. (Joyal *et al.* 1996) These data determine a tortile monoidal structure on

Int \mathcal{C} . Moreover, the functor $\mathcal{N} : \mathcal{C} \rightarrow \mathbf{Int} \mathcal{C}$ sending A to (A, I) strongly preserves the traced monoidal structure, and is full faithful.

Explicitly, the canonical trace on **Int** \mathcal{C} can be given as follows. (It is not entirely obvious for the non-symmetric case.) For $f : (A^+, A^-) \otimes (X^+, X^-) \rightarrow (B^+, B^-) \otimes (X^+, X^-)$, its trace $Tr^{(X^+, X^-)} f : (A^+, A^-) \rightarrow (B^+, B^-)$ is



It easily follows that \mathcal{N} preserves trace up to canonical isomorphisms.

In fact, Int-construction is *universal*, as shown in *ibid.*: it gives a left biadjoint to the forgetful 2-functor from the 2-category of tortile monoidal categories to that of traced monoidal categories.

3.4. Trace-Fixpoint Correspondence

The following correspondence between traces and fixed-point operators on categories with finite products, noticed by Martin Hyland and the author independently[†], shows that traces and fixed-point operators commonly used in computer science are very closely related.

Let \mathcal{C} be a category with finite products. A parameterized fixed-point operator on \mathcal{C} is a family of functions

$$(-)^\dagger : \mathcal{C}(A \times X, X) \rightarrow \mathcal{C}(A, X)$$

which is natural in A and satisfies the *fixed-point equation* $f^\dagger = f \circ \langle id_A, f^\dagger \rangle : A \rightarrow X$ for any $f : A \times X \rightarrow X$. It is called a *Conway fixed-point operator* if it satisfies the dinaturality

$$(f \circ \langle \pi_{A,X}, g \rangle)^\dagger = f \circ \langle id_A, (g \circ \langle \pi_{A,Y}, f \rangle)^\dagger \rangle : A \rightarrow X$$

for $f : A \times Y \rightarrow X$ and $g : A \times X \rightarrow Y$, and the diagonal property

$$(f \circ (id_A \times \Delta_X))^\dagger = (f^\dagger)^\dagger : A \rightarrow X$$

for $f : A \times X \times X \rightarrow X$, where $\Delta_X : X \rightarrow X \times X$ is the diagonal map.

Theorem 3.3. (Hasegawa 1997; Hasegawa 1999) For any category with finite products, to give a Conway operator is to give a trace (where finite products are taken as the monoidal structure).

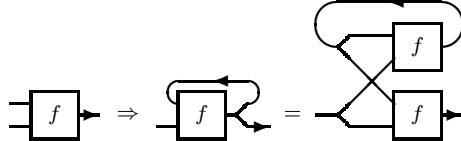
[†] We should note that mathematically equivalent observations were made by several authors before the notion of traced monoidal categories was introduced, in particular by Bloom and Ésik (Bloom and Ésik 1993) and Ştefanescu (Ştefanescu 2000).

Here we shall just recall the constructions of this bijective correspondence:

$$\frac{f : A \times X \rightarrow X}{f^\dagger = Tr_{A,X}^X(\Delta_X \circ f) : A \rightarrow X}$$

$$\frac{g : A \times X \rightarrow B \times X}{Tr_{A,B}^X(g) = \pi_{B,X} \circ (g \circ (id_A \times \pi'_{B,X}))^\dagger : A \rightarrow B}$$

The construction of Conway operator from a trace can be drawn as follows.



Thus many categories in denotational/algebraic semantics are traced. For example, the category **Cppo** of pointed cpo's and continuous functions is traced, where trace determined by the least fixed-points.

4. Traced Monoidal Closed Categories

So far we have not thought much about *closed structure*, or higher-types. Recall that a monoidal category \mathcal{C} is *closed* if $- \otimes A : \mathcal{C} \rightarrow \mathcal{C}$ has a right adjoint $A \multimap -$:

$$\mathcal{C}(X \otimes A, Y) \simeq \mathcal{C}(X, A \multimap Y)$$

In particular, tortile monoidal categories are closed, with $A \multimap B = B \otimes A^*$. We will denote (the Y -component of) the counit of this adjunction by $\text{ev}_{A,Y} : (A \multimap Y) \otimes A \rightarrow Y$, and for $f : X \otimes A \rightarrow Y$ let $\Lambda(f) : X \rightarrow A \multimap Y$ be the unique arrow satisfying $\text{ev}_{A,Y} \circ (\Lambda(f) \otimes id_A) = f$.

In the context of linear logic (Girard 1987), being symmetric monoidal closed means that we can interpret the intuitionistic multiplicative fragment (tensor \otimes , unit \top , and linear implication \multimap) in \mathcal{C} . In the rest of this paper, we will see that, for a traced monoidal category, closedness has yet another reading in terms of the Int-construction, which in turn is also related to the modality $!$ and linear decomposition $A \rightarrow B = !A \multimap B$ in linear logic.

4.1. Monoidal Closed Categories and Co-reflection

It is known that a monoidal co-reflective full subcategory of a monoidal closed category is also closed (although the closed structure may not be preserved by the inclusion):

Lemma 4.1. (folklore[‡]). Let $\mathcal{C} \xrightleftharpoons[\mathcal{U}]{\mathcal{F}}$ be a monoidal adjunction. If \mathcal{F} is full faithful and \mathcal{D} is closed, then \mathcal{C} is also closed, with $A \multimap_{\mathcal{C}} B = \mathcal{U}(\mathcal{F}A \multimap_{\mathcal{D}} \mathcal{F}B)$.

[‡] Martin Hyland (private communication) attributes this result to Brian Day.

Proof.

$$\begin{aligned}
 \mathcal{C}(C \otimes A, B) &\simeq \mathcal{D}(\mathcal{F}(C \otimes A), \mathcal{F}B) & \mathcal{F} \text{ is full faithful} \\
 &\simeq \mathcal{D}(\mathcal{F}C \otimes \mathcal{F}A, \mathcal{F}B) & \mathcal{F} \text{ is strong monoidal} \\
 &\simeq \mathcal{D}(\mathcal{F}C, \mathcal{F}A \multimap \mathcal{F}B) & \mathcal{D} \text{ is closed} \\
 &\simeq \mathcal{C}(C, \mathcal{U}(\mathcal{F}A \multimap \mathcal{F}B)) & \mathcal{F} \dashv \mathcal{U}
 \end{aligned}$$

Here we appeal to the fact that an adjunction $\mathcal{F} \dashv \mathcal{U}$ is a monoidal adjunction if and only if \mathcal{F} is strong monoidal (Kelly 1974). \square

4.2. Main Observation

Below we present a variation for traced monoidal categories. It characterizes closedness in terms of an adjunction associated to the Int-construction.

Theorem 4.1. (main observation) Let \mathcal{C} be a traced monoidal category, and $\mathcal{N} : \mathcal{C} \rightarrow \mathbf{Int} \mathcal{C}$ be the canonical inclusion from \mathcal{C} into $\mathbf{Int} \mathcal{C}$ (i.e. $\mathcal{N}(A) = (A, I)$). Then \mathcal{C} is closed if and only if \mathcal{N} has a right adjoint.

Corollary 4.1. Every traced monoidal closed category is equivalent to a monoidal co-reflexive full subcategory of a tortile monoidal category.

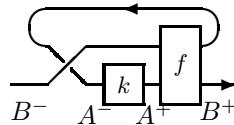
Proof of Theorem 4.1. “if” follows from the previous folklore lemma, as \mathcal{N} is full faithful and strong symmetric monoidal. Note that, by the lemma,

$$A \multimap_{\mathcal{C}} B \simeq \mathcal{U}(\mathcal{N}A \multimap_{\mathbf{Int} \mathcal{C}} \mathcal{N}B) = \mathcal{U}((A, I)^* \otimes (B, I)) \simeq \mathcal{U}(B, A)$$

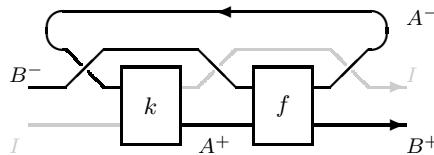
where \mathcal{U} is right adjoint to \mathcal{N} . This suggests how we proceed to show the “only if” part. That is, if \mathcal{C} is closed, define $\mathcal{U}(A^+, A^-) = A^- \multimap A^+$. For $f : (A^+, A^-) \rightarrow (B^+, B^-)$, let $\mathcal{U}(f) : (A^- \multimap A^+) \rightarrow (B^- \multimap B^+)$ be

$$\Lambda(Tr_{(A^- \multimap A^+) \otimes B^-, B^+}^{A^-}(f \circ (\text{ev}_{A^-, A^+} \otimes id_{B^-}) \circ (id_{A^- \multimap A^+} \otimes c_{B^-, A^-})))$$

or, more internally, a map sending $k : A^- \rightarrow A^+$ to $Tr_{B^-, B^+}^{A^-}(f \circ (k \otimes B^-) \circ c_{B^-, A^-}) : B^- \rightarrow B^+$, see the picture below.



In other words, $\mathcal{U}(f)$ describes “composition with f in $\mathbf{Int} \mathcal{C}$ ” in terms of \mathcal{C} .



From this, it is immediate to see that \mathcal{U} is indeed a functor.

Finally, it is easy to see the adjointness:

$$\begin{aligned}\mathbf{Int} \mathcal{C}(N(A), (B^+, B^-)) &= \mathbf{Int} \mathcal{C}((A, I), (B^+, B^-)) \\ &= \mathcal{C}(A \otimes B^-, B^+ \otimes I) \\ &\simeq \mathcal{C}(A, B^- \multimap B^+) \\ &= \mathcal{C}(A, \mathcal{U}(B^+, B^-)).\end{aligned}$$

The (A^+, A^-) -component of the counit of the adjunction is given by $\text{ev}_{A^-, A^+} : (A^- \multimap A^+) \otimes A^- \rightarrow A^+$, while the unit is trivial. \square

Note that the adjunction in the theorem above gives rise to an idempotent balanced monoidal comonad $N\mathcal{U}$ on $\mathbf{Int} \mathcal{C}$ which sends (A^+, A^-) to $(A^- \multimap A^+, I)$. Recall that, for an idempotent comonad K on a category \mathcal{C} , its co-Kleisli category \mathcal{C}_K and co-Eilenberg-Moore category \mathcal{C}^K are both equivalent to the co-reflexive full subcategory of \mathcal{C} whose objects are in the image of K . Thus we have:

Corollary 4.2. For any traced monoidal closed category \mathcal{C} , there is an idempotent balanced monoidal comonad on $\mathbf{Int} \mathcal{C}$ such that its co-Kleisli category is equivalent to \mathcal{C} .

Thus all traced monoical closed categories come from tortile monoidal categories with an idempotent balanced monoidal comonad. The converse is not true, however: there is a tortile monoidal category \mathcal{Y} with an idempotent balanced monoidal comonad $\mathcal{Y}_!$ such that $\mathbf{Int}(\mathcal{Y}_!)$ is not equivalent to \mathcal{Y} — see Section 6.2.

Remark 4.1. It might be the case that Int-construction and co-Kleisli construction give rise to a biadjunction between the 2-category of traced monoidal closed categories and a suitable 2-category of tortile monoidal categories with idempotent balanced monoidal comonad, but for now we do not know the answer.

5. Applications

5.1. Models of Linear Logic

We have already noted that symmetric monoidal closed categories are the models of multiplicative fragment of intuitionistic linear logic (IMLL). Here we shall quickly recall additional structures needed for modelling other elements of linear logic (Seely 1989; Barr 1991; Benton 1995; Bierman 1995; Barber and Plotkin 1997; Hyland and Schalk 2003; Melliès 2003).

A symmetric monoidal closed category with an object \perp so that the canonical map from A to $(A \multimap \perp) \multimap \perp$ is invertible for all objects A is called a **-autonomous category* (Barr 1991). **-autonomous* categories are the models of multiplicative fragment of classical linear logic (MLL), where \perp (called a dualizing object) models the falsity and $A \multimap \perp$ the linear negation of A . Compact closed categories (= tortile symmetric monoidal categories) are special instances of **-autonomous* categories, so they also are models of MLL.

To interpret additive conjunctions and disjunctions of linear logic, it suffices just to assume finite products and finite coproducts. However, they are not particularly impor-

tant in this paper; we do not know useful conditions to ensure the existence of finite (bi)products[§] on $\mathbf{Int} \mathcal{C}$, so for now additives do not have good place in our story.

A symmetric monoidal adjunction between a category with finite products and a symmetric monoidal category gives rise to a comonad on the symmetric monoidal category, which models the exponential $!$ of linear logic (Barber and Plotkin 1997; Bierman 1995). Such a comonad is called a *linear exponential comonad* (Hyland and Schalk 2003). Explicitly, a linear exponential comonad is a symmetric monoidal comonad $!$ on a symmetric monoidal category \mathcal{C} such that the category of its coalgebras is a category of commutative comonoids, which means

- there are specified monoidal natural transformations $e_A : !A \rightarrow I$ and $d_A : !A \rightarrow !A \otimes !A$ which form a commutative comonoid $(!A, e_A, d_A)$ in \mathcal{C} and also are coalgebra morphisms from $(!A, \delta_A)$ to (I, m_I) and $(!A \otimes !A, m_{!A,!A} \circ (\delta_A \otimes \delta_A))$ respectively, and
- any coalgebra morphism from $(!A, \delta_A)$ to $(!B, \delta_B)$ is also a comonoid morphism from $(!A, e_A, d_A)$ to $(!B, e_B, d_B)$.

In summary, a model of multiplicative exponential fragment of intuitionistic linear logic (IMELL) is a symmetric monoidal closed category with a linear exponential comonad. A model of multiplicative exponential fragment of classical linear logic (MELL) is a $*$ -autonomous category with a linear exponential comonad.

Returning to our study on traced categories, if \mathcal{C} is a traced cartesian closed category, our main theorem implies that $\mathcal{C} \xrightleftharpoons[\mathcal{U}]{\mathcal{N}} \mathbf{Int} \mathcal{C}$ is a symmetric monoidal adjunction, thus $\mathcal{N}\mathcal{U}$ is a linear exponential comonad on $\mathbf{Int} \mathcal{C}$.

Corollary 5.1. For any traced cartesian closed category \mathcal{C} , there is an idempotent linear exponential comonad on $\mathbf{Int} \mathcal{C}$ such that its co-Kleisli category is equivalent to \mathcal{C} .

Explicitly, this comonad sends (A^+, A^-) to $(A^- \Rightarrow A^+, 1)$.

Together with the trace-fixpoint correspondence:

Corollary 5.2. Any cartesian closed category with a Conway fixed-point operator is equivalent to one arising from a compact closed model of MELL via the co-Kleisli construction.

We can go further. Since monoidal adjunctions are closed under composition, Int-construction actually sends a traced model for IMELL to a compact closed model for MELL:

Corollary 5.3. Let \mathcal{C} be a traced symmetric monoidal closed category with a linear exponential comonad $!$ (i.e. a model of IMELL). Then $\mathbf{Int} \mathcal{C}$ is equipped with a linear exponential comonad $!$ ' given by $!(A^+, A^-) = \mathcal{N}(!(\mathcal{U}(A^+, A^-))) = (!!(A^- \multimap A^+), I)$.

$$! \subset \mathcal{C} \xrightleftharpoons[\mathcal{U}]{\mathcal{N}} \mathbf{Int} \mathcal{C}$$

[§] If a tortile monoidal category has finite products, they are biproducts (Houston 2006).

This gives an alternative possibility for interpreting exponentials in Geometry of Interaction — not quite the same as Girard's (Girard 1989) or the treatment by Abramsky, Haghverdi, Scott et al (Abramsky and Jagadeesan 1994; Haghverdi 2000; Abramsky *et al.*; Haghverdi and Scott 2006).

5.2. Linear Fixed-points: Recursion from Cyclic Sharing Revisited

In (Hasegawa 1997; Hasegawa 1999), it is shown that[¶]

Theorem 5.1. Given a symmetric monoidal adjunction $\mathcal{C} \xrightleftharpoons[\perp]{U} F \mathcal{D}$ between a category \mathcal{C} with finite products (taken as the monoidal structure) and a traced symmetric monoidal category \mathcal{D} , there exists a family of functions

$$(-)^\dagger : \mathcal{D}(FA \otimes X, X) \longrightarrow \mathcal{D}(FA, X)$$

which is natural in A and dinatural in X .

Explicitly, f^\dagger is given by

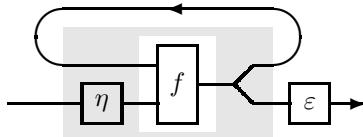
$$\varepsilon_X \circ Tr_{FA,FUX}^{FUX}(m_{UX,UX}^F)^{-1} \circ F(\Delta_{UX} \circ Uf \circ m_{FA,X}^U \circ (\eta_A \times id_{UX})) \circ m_{A,UX}^F$$

where η and ε are the unit and counit of the adjunction (should not be confused with those for tortile monoidal categories). In particular, f^\dagger satisfies the fixed-point equation:

$$f^\dagger = f \circ (id_{FA} \otimes f^\dagger) \circ m^{F^{-1}} \circ F\Delta_A$$

This result has been used for providing semantics of recursion in lambda calculi with cyclic sharing (Hasegawa 1997; Hasegawa 1999).

Using the box notation (Cockett and Seely 1999; Melliès 2006), this f^\dagger can be nicely expressed as follows.



In this picture, the inner box and outer box correspond to functors U and F respectively, and the components in the gray zone belong to \mathcal{C} while those in white to \mathcal{D} . Happily, we do not have to be bothered by the coherence morphisms m^F and m^U . In an appendix we give a graphical derivation of the fixed-point equation using the box notation, which replaces the lengthy calculation in (Hasegawa 1999).

If \mathcal{D} is closed, the operator $(-)^{\dagger}$ can be replaced by a family of arrows of $\mathcal{D}(FU(X \multimap X), X) \simeq \mathcal{D}(I, FU(X \multimap X) \multimap X)$. In terms of the linear lambda calculus corresponding

[¶] In (Hasegawa 1997; Hasegawa 1999), only Kleisli adjunctions of commutative monads were considered, and this theorem was stated with an additional condition that F is an identity-on-object, strict monoidal functor. But this restriction is not essential, as we demonstrate here.

to intuitionistic linear logic (Barber and Plotkin 1997), this amounts to a *linear* fixed-point combinator $\mathsf{Y}_X : !(\mathcal{X} \multimap \mathcal{X}) \multimap \mathcal{X}$ such that

$$\mathsf{Y}_X (!M) = M (\mathsf{Y}_X (!M))$$

holds for any term $M : \mathcal{X} \multimap \mathcal{X}$ with no free linear variable. Note that this is different from the usual fixed-point combinator with type $!(!\mathcal{X} \multimap \mathcal{X}) \multimap \mathcal{X}$ which returns a fixed point of a non-linear map of type $!\mathcal{X} \multimap \mathcal{X}$. As demonstrated in (Hasegawa 1997; Hasegawa 1999), linear fixed-point operators can exist even in the settings where such a standard non-linear fixed-point operator is not available.^{||}

Proposition 5.1. Suppose that \mathcal{D} is a traced symmetric monoidal closed category with a linear exponential comonad $!$. Then there exists a family of arrows $\mathsf{fix}_X : !(\mathcal{X} \multimap \mathcal{X}) \longrightarrow \mathcal{X}$ such that, for any $f : !A \otimes \mathcal{X} \longrightarrow \mathcal{X}$, $f^\dagger = \mathsf{fix}_X \circ !(\Lambda(f)) \circ \delta_A : !A \rightarrow \mathcal{X}$ (with $\Lambda(f) : !A \longrightarrow \mathcal{X} \multimap \mathcal{X}$ the adjoint mate of f) is a fixed-point of f , that is, f^\dagger agrees with

$$!A \xrightarrow{d} !A \otimes !A \xrightarrow{id_{!A} \otimes f^\dagger} !A \otimes \mathcal{X} \xrightarrow{f} \mathcal{X}.$$

As we have seen, if \mathcal{C} is a traced symmetric monoidal closed category with a linear exponential comonad $!$ (traced model of IMELL), then $\mathbf{Int} \mathcal{C}$ is a compact closed category with a linear exponential comonad $!^*(A^+, A^-) = (!(\mathcal{X}^- \multimap \mathcal{X}^+), I)$ (compact model of MELL). Therefore, both \mathcal{C} and $\mathbf{Int} \mathcal{C}$ admit interpretations of such linear fixed-points. The linear fixed-point on $(\mathcal{X}^+, \mathcal{X}^-)$ in $\mathbf{Int} \mathcal{C}$ is determined by the linear fixed-point on $\mathcal{X}^- \multimap \mathcal{X}^+$ in \mathcal{C} . Spelling out the detail, to give a linear fixed-point $\mathsf{Y}_{(\mathcal{X}^+, \mathcal{X}^-)} : !^*((\mathcal{X}^+, \mathcal{X}^-) \multimap (\mathcal{X}^+, \mathcal{X}^-)) \multimap (\mathcal{X}^+ \otimes \mathcal{X}^-)$ in $\mathbf{Int} \mathcal{C}$ is to give a map of

$$(\mathcal{X}^- \otimes !((\mathcal{X}^+ \otimes \mathcal{X}^-) \multimap (\mathcal{X}^+ \otimes \mathcal{X}^-))) \multimap \mathcal{X}^*$$

in \mathcal{C} , which is described as

$$\lambda(y^{X^1} \otimes !f^{((\mathcal{X}^+ \otimes \mathcal{X}^-) \multimap (\mathcal{X}^+ \otimes \mathcal{X}^-))}). \mathsf{Y}_{\mathcal{X}^- \multimap \mathcal{X}^+} (!(\mathcal{U}f)) y.$$

Example 5.1. Let \mathcal{C} be a traced cartesian closed category. Take a morphism $f = \langle f^+, f^- \rangle : (\mathcal{X}^+, \mathcal{X}^-) \rightarrow (\mathcal{X}^+, \mathcal{X}^-)$ in $\mathbf{Int} \mathcal{C}$, thus $f^+ : \mathcal{X}^+ \times \mathcal{X}^- \rightarrow \mathcal{X}^+$ and $f^- : \mathcal{X}^+ \times \mathcal{X}^- \rightarrow \mathcal{X}^-$ in \mathcal{C} . Using the simply typed lambda calculus with μ -fixpoint operator as an internal language of \mathcal{C} , $\mathcal{U}f : \mathcal{X}^{+\mathcal{X}^-} \rightarrow \mathcal{X}^{+\mathcal{X}^-}$ can be described as

$$\lambda k^{\mathcal{X}^- \rightarrow \mathcal{X}^+}. \lambda y^{\mathcal{X}^-}. f^+(\mu x^{\mathcal{X}^+}. k(f^-(x, y)), y^-) : (\mathcal{X}^- \rightarrow \mathcal{X}^+) \rightarrow (\mathcal{X}^- \rightarrow \mathcal{X}^+)$$

where $\mu z. g(z)$ denotes the corresponding Conway fixed-point. The linear fixed-point $f^\dagger : (1, 1) \rightarrow (\mathcal{X}^+, \mathcal{X}^-)$ is determined by a morphism from \mathcal{X}^- to \mathcal{X}^+ in \mathcal{C} , which is given by

^{||} One may define $\mathsf{Y}'_X : !(!\mathcal{X} \multimap \mathcal{X}) \multimap \mathcal{X}$ from Y by (using the syntax of DILL (Barber and Plotkin 1997))

$$\lambda g^{(!\mathcal{X} \multimap \mathcal{X})}. \text{let } !f^{!\mathcal{X} \multimap \mathcal{X}} \text{ be } g \text{ in let } !z^{\mathcal{X}} \text{ be } \mathsf{Y}_{!\mathcal{X}} (\lambda y^{!\mathcal{X}}. \text{let } !x^{\mathcal{X}} \text{ be } y \text{ in } !(f(!x))) \text{ in } z$$

but this does not satisfy the non-linear fixed-point equation $\mathsf{Y}' (!M) = M (!(\mathsf{Y}' (!M)))$. More concretely, consider the compact closed category \mathbf{Rel} of sets and relations, with the powerset comonad — this has a linear fixed-point operator (Hasegawa 1997; Hasegawa 1999) but no non-linear one (cf. (Melliès 2006)).

the fixed-point of $\mathcal{U}f$, that is,

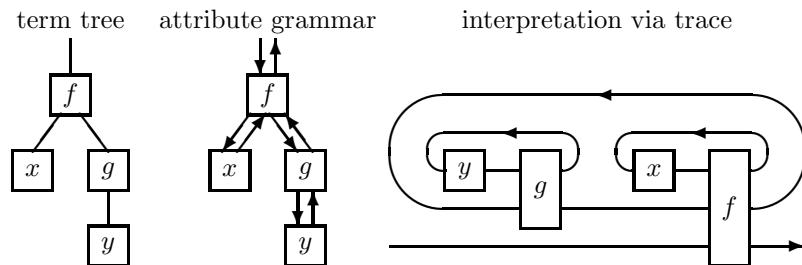
$$\mu k^{X^- \rightarrow X^+}.\lambda y^{X^-}.f^+(\mu x^{X^+}.k(f^-(x, y)), y^-) : X^- \rightarrow X^+.$$

6. Related Work and Discussions

6.1. Program Transformations and Attribute Grammars

Recently, Katsumata and Nishimura (Katsumata and Nishimura 2006) introduced a program transformation technique called *(semantic) higher-order removal*. Roughly, their technique transforms a higher-order map $g : (A^- \Rightarrow A^+) \Rightarrow (B^- \Rightarrow B^+)$ (created in the process of dealing with fusions of functions with accumulating parameters, which involves certain bi-directional information flow) to a less-expensive first-order map $f : A^+ \times B^- \Rightarrow B^+ \times A^-$ such that $\mathcal{U}(f) = g$ holds, where \mathcal{U} is right adjoint to \mathcal{N} . They give a syntactic condition which ensures that g is in the image of \mathcal{U} in their semantic models, and presented a procedure for identifying f such that $\mathcal{U}(f) = g$.

More recently, Katsumata (Katsumata 2007) has shown that a substantial part of the theory of attribute grammars (Knuth 1968) can be carried out very cleanly in terms of traced monoidal categories and Int-construction. Very roughly, an attribute grammar assigns computation with bidirectional information flow to term trees of a context free grammar, which can be interpreted in traced monoidal categories just in the same manner as Geometry of Interaction:



In Katsumata's work, the adjunction $\mathcal{N} \dashv \mathcal{U}$ of Theorem 4.1 provides the equivalence between attribute grammars and synthesized attribute grammars (attribute grammars with just bottom-up information flow). This generalizes the result on the equivalence between such attribute grammars formulated in a domain-theoretic setting citeCM79.

6.2. Game Semantics

An interesting (non)example is the category \mathcal{Y}^- of negative Conway games (Melliès 2004). \mathcal{Y}^- is a symmetric monoidal full subcategory of the compact closed category \mathcal{Y} of Conway games (Joyal 1977). The inclusion from \mathcal{Y}^- to \mathcal{Y} has a right adjoint. Thus (by the folklore lemma) \mathcal{Y}^- is a *traced symmetric monoidal closed category*. \mathcal{Y}^- is one of very few interesting traced symmetric monoidal closed categories which are neither cartesian closed nor compact closed.

Int \mathcal{Y}^- is *not* equivalent to \mathcal{Y} —thus it does not really fit in our result. But the difference is subtle; more precisely, **Int** \mathcal{Y}^- is equivalent to a compact closed full subcategory of \mathcal{Y} ,

whose objects are the tensor products of positive and negative games. We expect that similar situation should be found in many categories of games, which would deserve further study. A recent relevant example is the category of multi-bracketed games of Melliès and Tabareau (Melliès and Tabareau 2007).

6.3. Concluding Remarks

We have observed that closedness for a traced monoidal category is equivalent to an adjointness associated to the Int-construction. This simple result has a number of applications, on models of linear logic and fixed-point computation. We hope that these provide some good motivations to study traced monoidal *closed* categories. In particular, we expect that it is possible to develop “Geometry of Higher-Order Interaction” along the line of this work.

We shall conclude this paper by giving some further research directions. Firstly, to obtain a more solid understanding of traced monoidal closed categories, it is desirable to find a good concrete description of *free* traced monoidal closed categories. Intuitively, it should be a sort of “higher-order tangles” — in the same sense that a free traced monoidal category can be given as a category of usual (framed) tangles. Alternatively, it might be useful to consider free tortile monoidal categories with idempotent balanced monoidal comonads.

Secondly, related to the previous direction, and less ambitiously, it should be useful to develop a syntax for traced monoidal closed categories. Perhaps this can be done by extending existing term calculi or proof nets for linear logic.

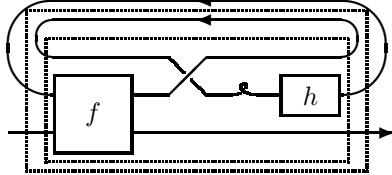
Thirdly, useful ways of constructing traced monoidal closed categories should be studied. We are yet to see if the constructions based on uniformity (Hasegawa 2004) can be used for the closed setting as well.

Appendix A. Derivation of Sliding and Vanishing

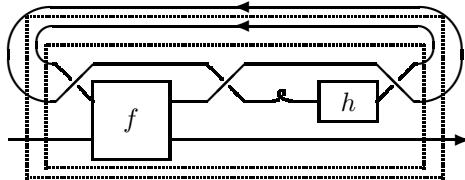
Proposition A.1. *Sliding* $Tr_{A,B}^X((id_B \otimes h) \circ f) = Tr_{A,B}^Y(f \circ (id_A \otimes h))$ is derivable.

Proof. By *Yanking*, the left hand side is equal to

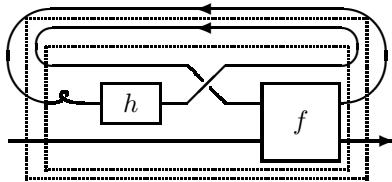
Using *Superposing* and *Tightening*, we have



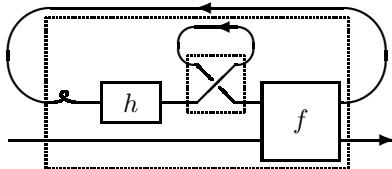
Then we apply *Exchange*:



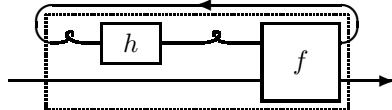
Thanks to the naturality of braidings, this is equal to the following.



Applying *Tightening* and *Superposing*, we obtain



By *Yanking*



This is equal to the right hand side of the equation, by the naturality of twisting. \square

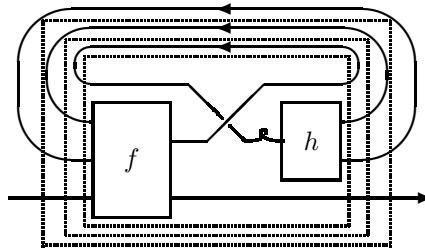
Proposition A.2. *Vanishing for tensor $Tr_{A,B}^X(Tr_{A \otimes X, B \otimes X}^Y(f)) = Tr_{A,B}^{X \otimes Y}(f)$ is derivable.*

Proof. We first show a variant of sliding

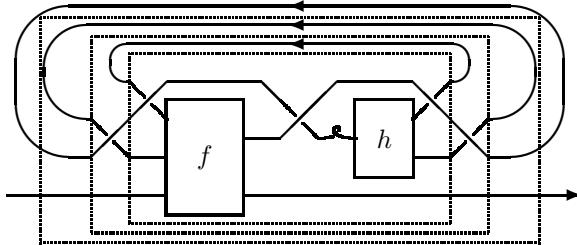
$$Tr_{A,B}^X(Tr_{A \otimes X, B \otimes X}^Y((id_B \otimes h) \circ f)) = Tr_{A,B}^Z(f \circ (id_A \otimes h))$$

for $f : A \otimes X \otimes Y \rightarrow B \otimes Z$ and $h : Z \rightarrow X \otimes Y$

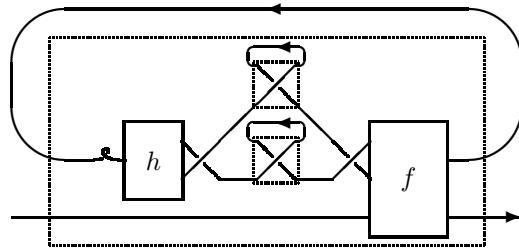
by the same technique used for deriving the sliding axiom above. From the left hand side of the equation, by yanking, superposing and tightening, we get



to which we apply the exchange axiom twice:



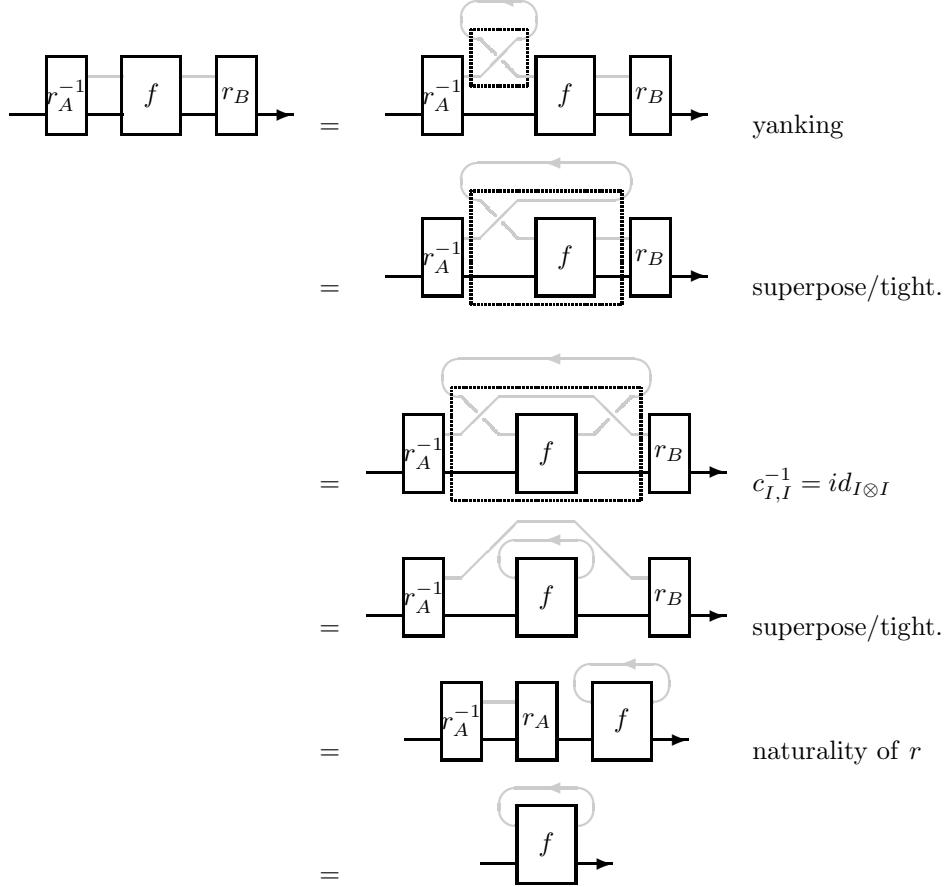
By the naturality and bilinearity of braidings, and by tightening, superposing, we have



The rest of proof is now easy, by yanking and the naturality of twisting; here we should recall the axiom $\theta_{A \otimes B} = c_{B,A} \circ (\theta_B \otimes \theta_A) \circ c_{A,B}$ of twists. The vanishing axiom for tensor follows from this equation, by letting Z be $X \otimes Y$ and h be $id_{X \otimes Y}$. \square

Remark A.1. Vanishing for unit $Tr_{A,B}^I f = r_B \circ f \circ r_A^{-1}$ ($f : A \otimes I \rightarrow B \otimes I$) is redundant

in original axioms in (Joyal *et al.* 1996). For completeness, here is a direct demonstration:

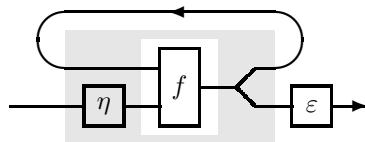


Appendix B. Linear Fixed-points Graphically

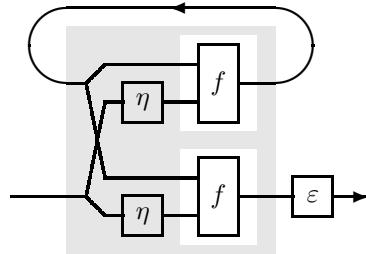
We show that, for $f : FA \otimes X \rightarrow X$,

$$f^\dagger = \varepsilon \circ Tr^{FUX}(m^{F^{-1}} \circ F(\Delta \circ Uf \circ m^U \circ (\eta \times id)) \circ m^F) : FA \rightarrow X$$

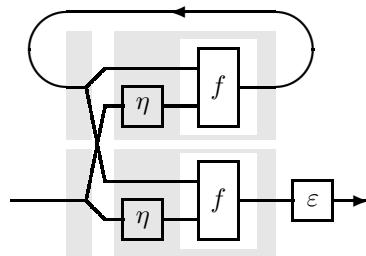
satisfies the (linear) fixed-point equation, using the box notation for monoidal functors (Cockett and Seely 1999; Melliès 2006).



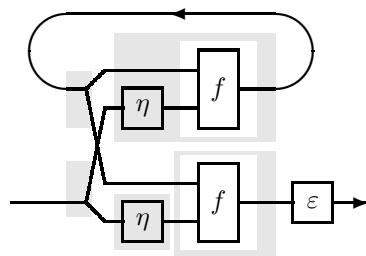
First, duplicate $Uf \circ m^U \circ (\eta \times id)$:



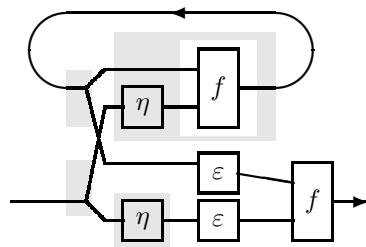
Since F is strong monoidal, the gray box can be decomposed —



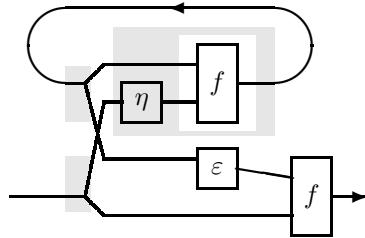
and further:



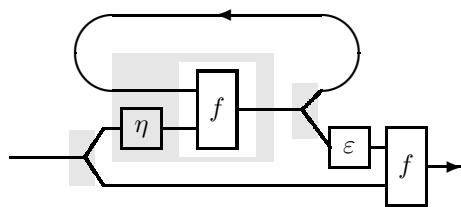
Since ε is a monoidal natural transformation from FU to the identity functor, we have



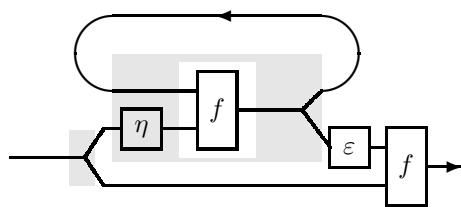
By adjointness, $\varepsilon \circ U\eta = id$:



By axioms of trace:



Now we have done: $f^\dagger = f \circ (id \otimes f) \circ m^{-1} \circ F\Delta$.



Acknowledgements

I am grateful to Paul-André Melliès and Shin-ya Katsumata for stimulating discussions and encouragements. This paper is based on a talk given at the workshop on Traced Monoidal Categories, Network Algebras and Applications (TMCNAA) held at Wrocław on July 2007. I thank the organizers and participants of the workshop for many helpful feedbacks. This work is partly supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 17700013.

References

- Abramsky, S. (1996) Retracing some paths in process algebra. In *Proc. Concurrency Theory*, Springer Lecture Notes in Comput. Sci. **1119**, pp. 1–17.
- Abramsky, S., Haghverdi, E. and Scott, P.J. (2002) Geometry of Interaction and linear combinatorial algebras. *Math. Struct. Comp. Sci.* **12**(5), 625–665.
- Abramsky, S. and Jagadeesan, R. (1994) New foundations for the Geometry of Interaction. *Inf. Comput.* **111**(1), 53–119.
- Barber, A. and Plotkin, G. (1997) Dual intuitionistic linear logic. Manuscript. An earlier version available as Technical Report ECS-LFCS-96-347, LFCS, University of Edinburgh.

- Barr, M. (1991) $*$ -autonomous categories and linear logic. *Math. Struct. Comp. Sci.* **1**, 159–178.
- Benton, N. (1995) A mixed linear non-linear logic: proofs, terms and models. In *Proc. Computer Science Logic*, Springer Lecture Notes in Comput. Sci. **933**, pp. 121–135.
- Bierman, G.M. (1995) What is a categorical model of intuitionistic linear logic? In *Proc. Typed Lambda Calculi and Applications*, Springer Lecture Notes in Comput. Sci. **902**, pp. 78–93.
- Bloom, S. and Ésik, Z. (1993) *Iteration Theories*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag.
- Blute, R.F., Cockett, J.R.B. and Seely, R.A.G. (2000) Feedback for linearly distributive categories: traces and fixpoints. *J. Pure Appl. Algebra* **154**, 27–69.
- Chirica, L.M. and Martin, D.F. (1979) An order-algebraic definition of knuthian semantics. *Mathematical Systems Theory* **13**, 1–27.
- Coccia, M., Gadducci, F. and Montanari, U. (2002) GS- Λ theories: a syntax for higher-order graphs. In *Proc. Category Theory and Computer Science*, Electr. Notes Theor. Comput. Sci. **69**.
- Cockett, J.R.B. and Seely, R.A.G. (1999) Linearly distributive functors. *J. Pure Appl. Algebra* **143**, 155–203.
- Girard, J.-Y. (1987) Linear logic. *Theoret. Comp. Sci.* **50**, 1–102.
- Girard, J.-Y. (1989) Geometry of Interaction I: interpretation of system F. In *Proc. Logic Colloquium '88*, pp. 221–260.
- Haghverdi, E. (2000) *A Categorical Approach to Linear Logic, Geometry of Interaction and Full Completeness*, PhD thesis, University of Ottawa.
- Haghverdi, E. and Scott, P.J. (2006) A categorical model for the geometry of interaction. *Theor. Comput. Sci.* **350**(2-3), 252–274.
- Hasegawa, M. (1997) Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *Proc. Typed Lambda Calculi and Applications*, Springer Lecture Notes in Comput. Sci. **1210**, pp. 196–213.
- Hasegawa, M. (1999) *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. Distinguished Dissertations Series, Springer-Verlag.
- Hasegawa, M. (2004) The uniformity principle on traced monoidal categories. *Publ. Res. Inst. Math. Sci.* **40**(3), 991–1014.
- Houston, R. (2006) Finite products are biproducts in a compact closed category. arXiv:math/0604542.
- Hyland, M. and Schalk, A. (2003) Glueing and orthogonality for models of linear logic. *Theoret. Comp. Sci.* **294**(1/2), 183–231.
- Joyal, A. and Street, R. (1991) The geometry of tensor calculus, I. *Adv. Math.* **88**, 55–113.
- Joyal, A. and Street, R. (1993) Braided tensor categories. *Adv. Math.* **102**, 20–78.
- Joyal, A., Street, R. and Verity, D. (1996) Traced monoidal categories. *Math. Proc. Cambridge Philos. Soc.* **119**, 447–468.
- Katsumata, S. (2007) A new foundation of attribute grammars in traced symmetric monoidal categories. Manuscript. Available from <http://www.kurims.kyoto-u.ac.jp/~sinya/index-e.html>
- Katsumata, S. and Nishimura, S. (2006) Algebraic fusion of functions with an accumulating parameter and its improvement. In *Proc. International Conference on Functional Programming*, pp. 227–238.
- Kelly, G.M. (1974) Doctorinal adjunction. In: *Proc. Sydney Category Theory Seminar*, Springer Lecture Notes in Math. **420**, pp. 257–280.
- Kelly, G.M. and Laplaza, M.L. (1980), Coherence for compact closed categories, *J. Pure Appl. Algebra* **19**, 193–213.

- Knuth, D.E. (1968) Semantics of context-free languages. *Mathematical Systems Theory* **2**(2), 127–145.
- Lambek, J. and Scott, P.J. (1986) *Introduction to Higher Order Categorical Logic*. Cambridge University Press.
- Mac Lane, S. (1971) *Categories for the Working Mathematician*. Graduate Texts in Mathematics **5**, Springer-Verlag.
- Melliès, P.-A. (2003) Categorical models of linear logic revisited. To appear in *Theoret. Comp. Sci.*
- Melliès, P.-A. (2004) Asynchronous games 3: an innocent model of linear logic. In *Proc. Category Theory and Computer Science*, Electr. Notes Theor. Comput. Sci. **122**, pp. 171–192.
- Melliès, P.-A. (2006) Functorial boxes in string diagrams. In *Proc. Computer Science Logic*, Springer Lecture Notes in Comput. Sci. **4207**, pp. 1–30.
- Melliès, P.-A. and Tabareau, N. (2007) Resource modalities in game semantics. In *Proc. 22nd Logic in Computer Science*, pp. 389–398.
- Milner, R. (1994) Action calculi V: reflexive molecular forms (with appendix by O. Jensen). Manuscript, LFCS, University of Edinburgh.
- Penrose, R. and Rindler, R. (1984) *Spinors and Space-Time*, Vol. 1. Cambridge University Press.
- Seely, R.A.G. (1989) Linear logic, *-autonomous categories and cofree coalgebras. In *Categories in Computer Science*, AMS Contemp. Math. **92**, pp. 371–389.
- Shum, M.-C. (1994) Tortile tensor categories, *J. Pure Appl. Algebra* **93**, 57–110.
- Simpson, A. and Plotkin, G., (2000) Complete axioms for categorical fixed-point operators, In *Proc. 15th Logic in Computer Science*, pp. 30–41.
- Stefănescu, G. (2000) *Network Algebra*. Discrete Mathematics and Theoretical Computer Science Series, Springer-Verlag.
- Yetter, D.N. (2001) *Functorial Knot Theory*. World Scientific.

Recursive Graphical Solution of Closed Schwinger-Dyson Equations in ϕ^4 -Theory – Part1: Generation of Connected and One-Particle Irreducible Feynman Diagrams

Axel Pelster and Konstantin Glaum

*Institut für Theoretische Physik, Freie Universität Berlin, Arnimallee 14, 14195 Berlin, Germany
pelster@physik.fu-berlin.de, glaum@physik.fu-berlin.de*

(Dated: November 11, 2018)

Using functional derivatives with respect to the free correlation function we derive a closed set of Schwinger-Dyson equations in ϕ^4 -theory. Its conversion to graphical recursion relations allows us to systematically generate all connected and one-particle irreducible Feynman diagrams for the two- and four-point function together with their weights.

PACS numbers: 05.70.Fh,64.60.-i

I. INTRODUCTION

Quantum and statistical field theory investigate the influence of field fluctuations on the n -point functions. Interactions lead to an infinite hierarchy of Schwinger-Dyson equations for the n -point functions [1–6]. These integral equations can only be closed approximately, for instance, by the well-established self-consistent method of Kadanoff and Baym [7].

Recently, it has been shown that the Schwinger-Dyson equations of QED can be closed in a certain functional-analytic sense [8]. Using functional derivatives with respect to the free propagators and the interaction [8–14] two closed sets of equations were derived. The first one involves the connected electron and two-point function as well as the connected three-point function, whereas the second one determines the electron and photon self-energy as well as the one-particle irreducible three-point function. Their conversion to graphical recursion relations leads to a systematic graphical generation of all connected and one-particle irreducible Feynman diagrams in QED, respectively.

The purpose of the present paper is to apply this functional-analytic approach to the ϕ^4 -theory of second-order phase transitions in the disordered, symmetric phase. A short outline of this program was already published in Ref. [15]. To this end we derive in Section II a closed set of equations for the connected two- and four-point function. Analogously, we determine in Section III a closed set of Schwinger-Dyson equations for the self-energy and the one-particle irreducible four-point function. In both cases, the closed set of Schwinger-Dyson equations can be converted into graphical recursion relations for the corresponding connected and one-particle irreducible Feynman diagrams in ϕ^4 -theory. From these the respective connected vacuum diagrams follow by short-circuiting external legs. Thus our present approach is complementary to Ref. [10] which was based on the observation that the complete knowledge of the vacuum energy implies the knowledge of the entire theory (“the vacuum is the world”) [16, 17]. In that paper the vacuum diagrams were constructed in a first step, together with their weights, as solutions of a graphical recursion relation derived from a nonlinear functional differential equation. In a second step, all diagrams with external lines are obtained from functional derivatives of the connected vacuum diagrams with respect to the free correlation function.

II. SCALAR ϕ^4 -THEORY

Euclidean ϕ^4 -theories in d dimensions are useful models for a large family of universality classes of continuous phase transitions [18]. In particular, the $O(N)$ -symmetric ϕ^4 -theory serves to describe the critical phenomena in dilute polymer solutions ($N = 0$), Ising- and Heisenberg-like magnets ($N = 1, 3$), and superfluids ($N = 2$). In all these systems, the thermal fluctuations of a self-interacting scalar order parameter field ϕ with N components are controlled by the Ginzburg-Landau energy functional

$$E[\phi] = \int d^d x \left\{ \frac{1}{2} \sum_{\alpha=1}^N \phi_\alpha(x) (-\partial_x^2 + m^2) \phi_\alpha(x) + \frac{g}{24} \left[\sum_{\alpha=1}^N \phi_\alpha^2(x) \right]^2 \right\}, \quad (1)$$

where the mass m^2 is proportional to the temperature deviation from the critical point, and g denotes the coupling constant. In the following it turns out to be advantageous to rewrite the Ginzburg-Landau energy functional (1) as

$$E[\phi] = \frac{1}{2} \int_{12} G_{12}^{-1} \phi_1 \phi_2 + \frac{1}{24} \int_{1234} V_{1234} \phi_1 \phi_2 \phi_3 \phi_4. \quad (2)$$

In this short-hand notation, the spatial and tensorial arguments of the order parameter field ϕ , the bilocal kernel G^{-1} , and the quartic interaction V are indicated by simple number indices, i.e.,

$$1 \equiv \{x_1, \alpha_1\}, \quad \int_1 \equiv \sum_{\alpha_1=1}^N \int d^d x_1, \quad \phi_1 \equiv \phi_{\alpha_1}(x_1). \quad (3)$$

The kernel G^{-1} represents the functional matrix

$$G_{12}^{-1} \equiv G_{\alpha_1, \alpha_2}^{-1}(x_1, x_2) = \delta_{\alpha_1, \alpha_2} (-\partial_{x_1}^2 + m^2) \delta(x_1 - x_2), \quad (4)$$

while the interaction V is given by the functional tensor

$$V_{1234} \equiv V_{\alpha_1, \alpha_2, \alpha_3, \alpha_4}(x_1, x_2, x_3, x_4) = \frac{g}{3} (\delta_{\alpha_1, \alpha_2} \delta_{\alpha_3, \alpha_4} + \delta_{\alpha_1, \alpha_3} \delta_{\alpha_2, \alpha_4} + \delta_{\alpha_1, \alpha_4} \delta_{\alpha_2, \alpha_3}) \delta(x_1 - x_2) \delta(x_1 - x_3) \delta(x_1 - x_4), \quad (5)$$

both being symmetric in their indices. For the purpose of this paper we shall leave the kernel G^{-1} in the energy functional (2) completely general, except for the symmetry with respect to its indices. By doing so, we regard the energy functional (2) as a functional of the kernel G^{-1} :

$$E[\phi] = E[\phi, G^{-1}]. \quad (6)$$

As a consequence, all global and local statistical quantities derived from (6) are also functionals of the bilocal kernel G^{-1} . In particular, we are interested in studying the functional dependence of the partition function, defined by a functional integral over a Boltzmann weight in natural units

$$Z[G^{-1}] = \int \mathcal{D}\phi e^{-E[\phi, G^{-1}]}, \quad (7)$$

and the (negative) vacuum energy

$$W[G^{-1}] = \ln Z[G^{-1}]. \quad (8)$$

For the sake of simplicity we restrict ourselves in the present paper to study the disordered, symmetric phase of the ϕ^4 -theory where the n -point functions

$$\mathbf{G}_{1\dots n}[G^{-1}] = \frac{1}{Z[G^{-1}]} \int \mathcal{D}\phi \phi_1 \dots \phi_n e^{-E[\phi, G^{-1}]}, \quad (9)$$

with odd n vanish. Thus the first nonvanishing n -point functions are the two-point function

$$\mathbf{G}_{12}[G^{-1}] = \frac{1}{Z[G^{-1}]} \int \mathcal{D}\phi \phi_1 \phi_2 e^{-E[\phi, G^{-1}]} \quad (10)$$

and the four-point function

$$\mathbf{G}_{1234}[G^{-1}] = \frac{1}{Z[G^{-1}]} \int \mathcal{D}\phi \phi_1 \phi_2 \phi_3 \phi_4 e^{-E[\phi, G^{-1}]} . \quad (11)$$

Further important statistical quantities are the correlation functions, i.e. the connected n -point functions. In the disordered, symmetric phase, the connected two-point function coincides with the two-point function

$$\mathbf{G}_{12}^c[G^{-1}] = \mathbf{G}_{12}[G^{-1}], \quad (12)$$

whereas the connected four-point function is defined by

$$\mathbf{G}_{1234}^c[G^{-1}] \equiv \mathbf{G}_{1234}[G^{-1}] - \mathbf{G}_{12}[G^{-1}] \mathbf{G}_{34}[G^{-1}] - \mathbf{G}_{13}[G^{-1}] \mathbf{G}_{24}[G^{-1}] - \mathbf{G}_{14}[G^{-1}] \mathbf{G}_{23}[G^{-1}] . \quad (13)$$

By expanding the functional integrals (7) and (9) in powers of the coupling constant g , the expansion coefficients of the partition function and the n -point functions consist of free-field expectation values. These are evaluated with the help of Wick's rule as a sum of Feynman integrals, which are pictured as diagrams constructed from lines and vertices.

Thereby the free correlation function G_{12} , which is the functional inverse of the kernel G^{-1} in the energy functional (2)

$$\int_2 G_{12} G_{23}^{-1} = \delta_{13} \quad (14)$$

with $\delta_{13} = \delta_{\alpha_1, \alpha_3} \delta(x_1 - x_3)$, is represented by a line in a Feynman diagram

$$G_{12} \equiv 1 \text{ --- } 2 , \quad (15)$$

and the interaction V is pictured as a vertex

$$- V_{1234} \equiv \begin{array}{c} 2 & 3 \\ \times \\ 1 & 4 \end{array} . \quad (16)$$

The graphical elements (15) and (16) are combined by an integral which graphically corresponds to the gluing prescription

$$- \int_4 V_{1234} G_{45} \equiv \begin{array}{c} 2 & 3 \\ \times \\ 1 & 4 & 4 \text{ --- } 5 \end{array} \equiv \begin{array}{c} 2 & 3 \\ \times \\ 1 & 4 & 5 \end{array} . \quad (17)$$

In this paper we analyze the resulting diagrams for the statistical quantities (7)–(13) by using their functional dependence on the kernel G_{12}^{-1} . To this end we introduce the functional derivative with respect to the kernel G_{12}^{-1} whose basic rule reflects the symmetry of its indices:

$$\frac{\delta G_{12}^{-1}}{\delta G_{34}^{-1}} = \frac{1}{2} (\delta_{13}\delta_{42} + \delta_{14}\delta_{32}) . \quad (18)$$

From the identity (14) and the functional product rule we find the effect of this derivative on the free correlation function

$$-\frac{\delta G_{12}}{\delta G_{34}^{-1}} = \frac{1}{2} (G_{13}G_{42} + G_{14}G_{32}) , \quad (19)$$

which has the graphical representation

$$-\frac{\delta}{\delta G_{34}^{-1}} 1 \text{ --- } 2 = \frac{1}{2} \left(1 \text{ --- } 3 \text{ --- } 4 \text{ --- } 2 + 1 \text{ --- } 4 \text{ --- } 3 \text{ --- } 2 \right) . \quad (20)$$

Thus a functional derivative with respect to the kernel G_{12}^{-1} is represented by a graphical operation which cuts a line of a Feynman diagram in all possible ways [8–14]. For practical purposes it is convenient to use also functional derivatives with respect to the free correlation function G_{12} whose basic rule reads

$$\frac{\delta G_{12}}{\delta G_{34}} \equiv \frac{\delta 1 \text{ --- } 2}{\delta 3 \text{ --- } 4} = \frac{1}{2} (\delta_{13}\delta_{42} + \delta_{14}\delta_{32}) . \quad (21)$$

Such functional derivatives are represented graphically by removing a line of a Feynman diagram in all possible ways [8–14]. The functional derivatives with respect to the kernel G_{12}^{-1} and the correlation function G_{12} are related via the functional chain rule

$$\frac{\delta}{\delta G_{12}^{-1}} = - \int_{34} G_{13}G_{24} \frac{\delta}{\delta G_{34}} . \quad (22)$$

These functional derivatives are used in Subsection II A to derive a closed set of Schwinger-Dyson equations for the connected two- and four-point functions. In Subsection II B they are converted into graphical recursion relations for the corresponding connected Feynman diagrams. Finally, the connected vacuum diagrams contributing to the vacuum energy are constructed in a graphical way in Subsection II C.

A. Closed Set of Equations for Connected Two- and Four-Point Function

In this subsection we apply the functional derivatives introduced so far to a functional identity which immediately follows from the definition of the functional integral. By doing so, we derive a closed set of Schwinger-Dyson equations determining the connected two- and four-point function.

1. Connected Two-Point Function

In order to derive an equation for the connected two-point function, we start with the trivial identity

$$\int \mathcal{D}\phi \frac{\delta}{\delta\phi_1} (\phi_2 e^{-E[\phi]}) = 0, \quad (23)$$

which follows via direct functional integration from the vanishing of the exponential at infinite fields. Taking into account the explicit form of the energy functional (2), we perform the functional derivative with respect to the field and obtain

$$\int \mathcal{D}\phi \left(\delta_{12} - \int_3 G_{13}^{-1} \phi_2 \phi_3 - \frac{1}{6} \int_{345} V_{1345} \phi_2 \phi_3 \phi_4 \phi_5 \right) e^{-E[\phi]} = 0. \quad (24)$$

Applying the definitions (7)–(11) and (13), this equation can be expressed in terms of the connected two- and four-point function as follows

$$\delta_{12} - \int_3 G_{13}^{-1} \mathbf{G}_{23} = \frac{1}{2} \int_{345} V_{1345} \mathbf{G}_{34} \mathbf{G}_{52} + \frac{1}{6} \int_{345} V_{1345} \mathbf{G}_{2345}^c. \quad (25)$$

Multiplying this equation with G_{16} and integrating with respect to the index 1 finally leads to the Schwinger-Dyson equation which determines the connected two-point function:

$$\mathbf{G}_{12} = G_{12} - \frac{1}{2} \int_{3456} G_{13} V_{3456} \mathbf{G}_{45} \mathbf{G}_{62} - \frac{1}{6} \int_{3456} G_{13} V_{3456} \mathbf{G}_{2456}^c. \quad (26)$$

When the connected two-point function is graphically represented in Feynman diagrams by a double line

$$\mathbf{G}_{12} \equiv \text{---} \overline{\text{---}} \text{---} 2, \quad (27)$$

and the connected four-point function is pictured by a vertex with an open dot with four legs

$$\mathbf{G}_{1234}^c \equiv \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \text{---} \text{---} \text{---} \\ | \quad | \quad | \\ 1 \quad 3 \quad 4 \end{array}, \quad (28)$$

this Schwinger-Dyson equation reads graphically:

$$\text{---} \overline{\text{---}} \text{---} 2 = \text{---} \overline{\text{---}} \text{---} 2 + \frac{1}{2} \text{---} \overline{\text{---}} \text{---} 2 + \frac{1}{6} \text{---} \overline{\text{---}} \text{---} 2. \quad (29)$$

It represents an integral equation for the connected two-point function on the left-hand side which turns out to appear also on the right-hand side. Iteratively solving the integral equation (29) for the connected two-point function \mathbf{G}_{12} necessitates, however, the knowledge of the connected four-point function \mathbf{G}_{1234}^c .

2. Connected Four-Point Function

In principle, one could determine the connected four-point function (13) from the connected two-point function (10) as follows. We obtain from (2) and (18)

$$\phi_1 \phi_2 = 2 \frac{\delta E[\phi]}{\delta G_{12}^{-1}}, \quad (30)$$

so that we yield from (7), (11), and (13)

$$\mathbf{G}_{1234}^c = -2 \frac{\delta \mathbf{G}_{12}}{\delta G_{34}^{-1}} - \mathbf{G}_{13} \mathbf{G}_{24} - \mathbf{G}_{14} \mathbf{G}_{23}. \quad (31)$$

This result reads graphically

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{1} \\ \diagdown \quad \diagup \\ 4 \end{array} = -2 \frac{\delta \text{---} 2}{\delta G_{34}^{-1}} - \text{---} 1 \text{---} 3 \text{---} 2 \text{---} 4 - \text{---} 1 \text{---} 4 \text{---} 2 \text{---} 3 . \quad (32)$$

However, such a procedure would have the disadvantage that cutting a line in the diagrams of the connected two-point function \mathbf{G}_{12} would also lead to disconnected diagrams which are later removed by the second and the third term on the right-hand side of (32). As the number of undesired disconnected diagrams occurring at an intermediate step of the calculation increases with the loop order, this procedure is quite inefficient to determine the connected four-point function \mathbf{G}_{1234}^c . Therefore we aim at deriving another equation for \mathbf{G}_{1234}^c whose iterative solution only involves connected diagrams. To this end we insert the Schwinger-Dyson equation (26) into relation (31) and obtain

$$\begin{aligned}
\mathbf{G}_{1234}^c = & \frac{1}{3} \int_{5678} G_{15} V_{5678} \frac{\delta \mathbf{G}_{6782}^c}{\delta G_{34}^{-1}} - \int_{5678} G_{15} V_{5678} \mathbf{G}_{62} \mathbf{G}_{73} \mathbf{G}_{84} - \frac{1}{2} \int_{5678} G_{15} V_{5678} \mathbf{G}_{6734}^c \mathbf{G}_{82} \\
& - \frac{1}{2} \int_{5678} G_{15} V_{5678} \mathbf{G}_{8234}^c \mathbf{G}_{67} + \frac{1}{6} \int_{5678} G_{15} V_{5678} \mathbf{G}_{6784}^c \mathbf{G}_{23} + \frac{1}{6} \int_{5678} G_{15} V_{5678} \mathbf{G}_{6783}^c \mathbf{G}_{24} .
\end{aligned} \quad (33)$$

The last two terms in (33) are still disconnected and cancel the disconnected diagrams which are generated by the functional derivative of the connected four-point function with respect to the kernel in the first term. In order to eliminate all disconnected terms from (33) we use the commutator relation

$$\begin{aligned}
\frac{\delta \mathbf{G}_{1234}^c}{\delta G_{56}^{-1}} - \frac{\delta \mathbf{G}_{1256}^c}{\delta G_{34}^{-1}} = & \frac{1}{2} (\mathbf{G}_{13} \mathbf{G}_{2456}^c + \mathbf{G}_{14} \mathbf{G}_{2356}^c + \mathbf{G}_{23} \mathbf{G}_{1456}^c + \mathbf{G}_{24} \mathbf{G}_{1356}^c \\
& - \mathbf{G}_{15} \mathbf{G}_{2346}^c - \mathbf{G}_{16} \mathbf{G}_{2345}^c - \mathbf{G}_{25} \mathbf{G}_{1346}^c - \mathbf{G}_{26} \mathbf{G}_{1345}^c) ,
\end{aligned} \quad (34)$$

which directly follows from (31) because of the identity that mixed second functional derivatives with respect to the kernel G^{-1} can be interchanged:

$$\frac{\delta^2 \mathbf{G}_{12}}{\delta G_{56}^{-1} \delta G_{34}^{-1}} = \frac{\delta^2 \mathbf{G}_{12}}{\delta G_{34}^{-1} \delta G_{56}^{-1}} . \quad (35)$$

The commutator relation (34) states that within the functional derivative of the connected four-point function with respect to the kernel indices might be interchanged at the expense of the additional terms on the right-hand side. Inserting (34) in (33) by taking into account the Schwinger-Dyson equation (26) and the functional chain rule (22), we yield the following functional integrodifferential equation for the connected four-point function:

$$\begin{aligned}
\mathbf{G}_{1234}^c = & - \int_{5678} G_{15} V_{5678} \mathbf{G}_{62} \mathbf{G}_{73} \mathbf{G}_{84} - \frac{1}{3} \int_{5678} G_{15} V_{5678} G_{69} G_{70} \frac{\delta \mathbf{G}_{8234}^c}{\delta G_{90}} - \frac{1}{6} \int_{5678} G_{15} V_{5678} \mathbf{G}_{8234}^c \mathbf{G}_{67} \\
& - \frac{1}{6} \int_{5678} G_{15} V_{5678} \mathbf{G}_{6723}^c \mathbf{G}_{84} - \frac{1}{6} \int_{5678} G_{15} V_{5678} \mathbf{G}_{6724}^c \mathbf{G}_{83} - \frac{1}{6} \int_{5678} G_{15} V_{5678} \mathbf{G}_{6734}^c \mathbf{G}_{82} .
\end{aligned} \quad (36)$$

Although Eq. (36) has the disadvantage of being more complex than Eq. (31), it has the advantage that it does not lead to disconnected diagrams at an intermediate stage of the calculation. This can be immediately seen in its graphical representation which reads

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{1} \\ \diagdown \quad \diagup \\ 4 \end{array} = \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ 1 \quad 4 \\ \diagdown \quad \diagup \\ 3 \end{array} + \frac{1}{3} \begin{array}{c} 5 \\ | \\ 1 \text{---} 6 \text{---} 7 \\ | \\ \delta \text{---} 4 \\ | \\ 6 \text{---} 7 \end{array} + \frac{1}{6} \begin{array}{c} 2 \\ | \\ 1 \text{---} \textcircled{5} \\ | \\ 4 \end{array} + \frac{1}{6} \begin{array}{c} 2 \\ | \\ 1 \text{---} \textcircled{5} \\ | \\ 4 \end{array} \\
+ \frac{1}{6} \begin{array}{c} 4 \\ | \\ 1 \text{---} \textcircled{5} \\ | \\ 3 \end{array} + \frac{1}{6} \begin{array}{c} 3 \\ | \\ 1 \text{---} \textcircled{5} \\ | \\ 4 \end{array} + \frac{1}{6} \begin{array}{c} 2 \\ | \\ 1 \text{---} \textcircled{5} \\ | \\ 4 \end{array} .
\end{math>$$

Thus the closed set of Schwinger-Dyson equations for the connected two- and four-point function is given by (29) and (37).

B. Graphical Recursion Relations

Now we demonstrate how the diagrams of the connected two- and four-point function are recursively generated in a graphical way. To this end we perform for both quantities a perturbative expansion

$$1 \overline{\overline{}} 2 \equiv \sum_{p=0}^{\infty} 1 \overset{(p)}{\overline{\overline{}}} 2 , \quad (38)$$

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{p} \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} \equiv \sum_{p=0}^{\infty} \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ p \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} , \quad (39)$$

where p denotes the number of interactions V which contribute. With these we obtain from (29) and (37) the following closed set of graphical recursion relations:

$$1 \overset{(p+1)}{\overline{\overline{}}} 2 = \frac{1}{2} \sum_{q=0}^p 1 \overset{(p-q)}{\overline{\overline{}}} \textcircled{(q)} 2 + \frac{1}{6} 1 \text{---} \textcircled{p} \text{---} 2 , \quad (40)$$

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{p+1} \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} = \sum_{q=0}^p \sum_{r=0}^q \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{(r)} \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} + \frac{1}{3} 1 \text{---} \textcircled{5} \text{---} \frac{\delta}{\delta 6} \text{---} \textcircled{7} + \frac{1}{6} \sum_{q=1}^p 1 \overset{(p-q)}{\overline{\overline{}}} \textcircled{q} \text{---} 3 \\ + \frac{1}{6} \sum_{q=1}^p \begin{array}{c} 4 \\ \diagup \quad \diagdown \\ \textcircled{q} \\ \diagdown \quad \diagup \\ 1 \quad 2 \end{array} + \frac{1}{6} \sum_{q=1}^p \begin{array}{c} 3 \\ \diagup \quad \diagdown \\ \textcircled{q} \\ \diagdown \quad \diagup \\ 1 \quad 2 \end{array} + \frac{1}{6} \sum_{q=1}^p \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{q} \\ \diagdown \quad \diagup \\ 1 \quad 3 \end{array} . \quad (41)$$

This is solved starting from

$$1 \overset{(0)}{\overline{\overline{}}} 2 = 1 \overline{\overline{}} 2 , \quad (42)$$

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ 0 \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} = 0 . \quad (43)$$

Note that these graphical recursion relations (40)–(43) allow to prove via complete induction that all diagrams contributing to the connected two- and four-point function are, indeed, connected [19].

The first few perturbative contributions to G_{12} and G_{1234}^c are determined as follows. Inserting (42) and (43) in (40) and (41), we obtain for $p = 1$ the connected two-point function

$$1 \overset{(1)}{\overline{\overline{}}} 2 = \frac{1}{2} 1 \text{---} \textcircled{1} \text{---} 2 \quad (44)$$

and the connected four-point function

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ 1 \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} = \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \times \\ \diagdown \quad \diagup \\ 1 \quad 4 \end{array} . \quad (45)$$

With this we get from (40) the second-order contribution to the connected two-point function

$$1 \overset{(2)}{\overline{\overline{}}} 2 = \frac{1}{4} 1 \text{---} \textcircled{2} \text{---} 2 + \frac{1}{4} 1 \text{---} \textcircled{1} \text{---} \textcircled{1} \text{---} 2 + \frac{1}{6} 1 \text{---} \textcircled{2} \text{---} 2 . \quad (46)$$

Amputating one line from (45),

$$\begin{aligned}
 \frac{\delta}{\delta 6} \text{ (Diagram 1)} &= \frac{1}{2} \delta_{26} \text{ (Diagram 2)} + \frac{1}{2} \delta_{36} \text{ (Diagram 3)} + \frac{1}{2} \delta_{46} \text{ (Diagram 4)} + \frac{1}{2} \delta_{27} \text{ (Diagram 5)} \\
 &\quad + \frac{1}{2} \delta_{37} \text{ (Diagram 6)} + \frac{1}{2} \delta_{47} \text{ (Diagram 7)} + \frac{1}{2} \delta_{56} \text{ (Diagram 8)} + \frac{1}{2} \delta_{57} \text{ (Diagram 9)}, \quad (47)
 \end{aligned}$$

we find the second-order contribution to the connected four-point function from (41):

$$\begin{aligned}
 \text{Diagram 2} &= \frac{1}{2} \left(\text{Diagram 1} + \text{Diagram 2} + \text{Diagram 3} + \text{Diagram 4} + \text{Diagram 5} \right) \\
 &\quad + \frac{1}{2} \left(\text{Diagram 6} + \text{Diagram 7} + \text{Diagram 8} + \text{Diagram 9} \right). \quad (48)
 \end{aligned}$$

The results (44)–(46), and (48) are listed in Table I and II which show all diagrams of the connected two- and four-point function up to the forth perturbative order irrespective of their spatial indices. However, to evaluate the recursion relations in higher orders, it becomes necessary to reassign the spatial indices to the end points of the diagrams of the connected two- and four-point function in Table I and II. This involves a decomposition of the weights shown in Table I and II due to symmetry considerations. To this end we characterize a diagram by its symmetry degree N . Thus reassigning the spatial indices to the end points of the diagrams of the connected two- and four-point function leads to $24/N$ different diagrams. As an example for determining the symmetry degree N we consider diagram #4.5 in Table II. Successively assigning the indices 1, 2, 3, 4 to the respective end points leads to $N = 4$:

$$\begin{aligned}
 \text{Diagram 24} &\mapsto \text{Diagram 24} \quad \text{Diagram 24} \mapsto \text{Diagram 8} + \text{Diagram 8} + \text{Diagram 8} \quad \text{Diagram 8} \mapsto \\
 &\mapsto \text{Diagram 4} + \text{Diagram 4}. \quad (49)
 \end{aligned}$$

Note that the weights of the diagrams of the connected two- and four-point function in Table I and II have been determined by solving the graphical recursion relations. However, as a cross-check, they also follow from the formula [10, 18, 20]

$$w^{(n)} = \frac{n!}{2^{S+D} 3^T P N}, \quad (50)$$

where n stands for the number of external legs and S, D, T denote the number of self-, double, triple connections between vertices. Furthermore, P stands for the number of vertex permutations leaving the diagrams unchanged.

C. Connected Vacuum Diagrams

The connected vacuum diagrams of ϕ^4 -theory can be generated order by order together with their weights in two ways. In this section we show that they follow from short-circuiting the external legs of the diagrams of the connected two- and four-point function, respectively. Thus our present approach is complementary to Ref. [10], where a nonlinear functional integrodifferential equation for the vacuum energy was recursively solved in a graphical way in order to directly generate the connected vacuum diagrams.

1. Relation to the Diagrams of the Connected Two-Point Function

Our first approach is based on the connected two-point function \mathbf{G}_{12} . We start with concluding

$$\mathbf{G}_{12} = -2 \frac{\delta W}{\delta G_{12}^{-1}}, \quad (51)$$

which follows from (7), (8), and (10) by taking into account (30). We can read off from (51) that cutting a line of the connected diagrams of the vacuum energy in all possible ways leads to all diagrams of the connected two-point function. Here, however, we want to regard (51) as a functional differential equation for the vacuum energy W . If the interaction V vanishes, Eq. (51) is solved by the free contribution of the vacuum energy

$$W^{(0)} = -\frac{1}{2} \text{Tr} \ln G^{-1}. \quad (52)$$

Here the trace of the logarithm of the kernel G^{-1} is defined by the series [22, p. 16]

$$\text{Tr} \ln G^{-1} \equiv -\sum_{n=1}^{\infty} \frac{1}{n} \int_{1\dots n} (\delta_{12} - G_{12}^{-1}) \dots (\delta_{n-1,n} - G_{n-1,n}^{-1}) (\delta_{n1} - G_{n1}^{-1}), \quad (53)$$

so that we get

$$-2 \frac{\delta W^{(0)}}{\delta G_{12}^{-1}} = G_{12}. \quad (54)$$

For a non-vanishing interaction V , the functional differential equation (51) produces corrections to (52), which we shall denote with $W^{(\text{int})}$. Thus the vacuum energy decomposes according to

$$W = W^{(0)} + W^{(\text{int})}, \quad (55)$$

and we obtain together with (51) and (54)

$$-\int_{12} G_{12}^{-1} \frac{\delta W^{(\text{int})}}{\delta G_{12}^{-1}} = \frac{1}{2} \int_{12} G_{12}^{-1} (\mathbf{G}_{12} - G_{12}). \quad (56)$$

In the following, we aim at recursively determining $W^{(\text{int})}$ in a graphical way. To this end we perform a perturbative expansion of the interaction part of the vacuum energy

$$W^{(\text{int})} = \sum_{p=1}^{\infty} W^{(p)}, \quad (57)$$

and a corresponding one of the connected two-point function

$$\mathbf{G}_{12} = G_{12} + \sum_{p=1}^{\infty} \mathbf{G}_{12}^{(p)}. \quad (58)$$

Inserting (57) and (58) into (56) and using the functional chain rule (22), we obtain for $p \geq 1$

$$\int_{12} G_{12} \frac{\delta W^{(p)}}{\delta G_{12}} = \frac{1}{2} \int_{12} G_{12}^{-1} \mathbf{G}_{12}^{(p)}. \quad (59)$$

The contributions $W^{(p)}$ of the vacuum energy obey the following eigenvalue problem for $p \geq 1$

$$\int_{12} G_{12} \frac{\delta W^{(p)}}{\delta G_{12}} = 2p W^{(p)}. \quad (60)$$

Indeed, in the p th perturbative order the functional derivative $\delta/\delta G_{12}$ generates diagrams in each of which one of the $2p$ lines of the original vacuum diagram is removed and, subsequently, the removed line is again reinserted. Thus the left-hand side of (60) results in counting the lines of the vacuum diagrams in $W^{(p)}$, so that we obtain the eigenvalue

$2p$. With (60) we can explicitly solve (59) for the respective perturbative contributions of the vacuum energy and obtain for $p \geq 1$

$$W^{(p)} = \frac{1}{4p} \int_{12} G_{12}^{-1} \mathbf{G}_{12}^{(p)} . \quad (61)$$

Now we supplement the above-mentioned Feynman rules with a graphical representation for the contributions of the vacuum energy

$$W^{(p)} \equiv \text{---} \circlearrowleft p \text{---} . \quad (62)$$

and for the kernel

$$G_{12}^{-1} \equiv \text{---} \circlearrowleft \text{---} . \quad (63)$$

The latter graphical element serves for gluing two lines together according to

$$\int_{12} G_{31} G_{12}^{-1} G_{24} \equiv \text{---} 3 \text{---} 1 \text{---} \circlearrowleft 2 \text{---} 4 \text{---} 4 = \text{---} 3 \text{---} 4 , \quad (64)$$

which follows from

$$\int_{12} G_{31} G_{12}^{-1} G_{24} = G_{34} . \quad (65)$$

Thus our result (61) can be depicted graphically as follows:

$$\text{---} \circlearrowleft p \text{---} = \frac{1}{4p} \text{---} \circlearrowleft (p) \text{---} . \quad (66)$$

It states that closing the perturbative contributions of the connected two-point function yields the corresponding contributions of the vacuum energy. From (44) and (66) we yield for $p = 1$

$$\text{---} \circlearrowleft 1 \text{---} = \frac{1}{8} \text{---} \circlearrowleft \text{---} \text{---} . \quad (67)$$

Correspondingly, the second-order result follows from (46) and (66):

$$\text{---} \circlearrowleft 2 \text{---} = \frac{1}{16} \text{---} \circlearrowleft \text{---} \text{---} \text{---} + \frac{1}{48} \text{---} \circlearrowleft \text{---} \text{---} \text{---} . \quad (68)$$

2. Relation to the Diagrams of the Connected Four-Point Function

Now we elaborate a second approach which is based on closing the diagrams of the connected four-point function \mathbf{G}_{1234}^c . To this end we insert (51) into the left-hand side of the Schwinger-Dyson equation (25) and use (22), (54) as well as the decomposition (55) to obtain a functional differential equation for the interaction part of the vacuum energy:

$$\int_{12} G_{12} \frac{\delta W^{(\text{int})}}{\delta G_{12}} = -\frac{1}{12} \int_{1234} V_{1234} \mathbf{G}_{1234}^c - \frac{1}{4} \int_{1234} V_{1234} \mathbf{G}_{12} \mathbf{G}_{34} . \quad (69)$$

Combining the perturbation expansions (57), (58) for $W^{(\text{int})}$, \mathbf{G}_{12} with the corresponding one for \mathbf{G}_{1234}^c , i.e.

$$\mathbf{G}_{1234}^c = \sum_{p=1}^{\infty} \mathbf{G}_{1234}^{c,(p)}, \quad (70)$$

we obtain together with the eigenvalue problem (60) from (69)

$$W^{(p+1)} = -\frac{1}{24(p+1)} \int_{1234} V_{1234} \mathbf{G}_{1234}^{c,(p)} - \frac{1}{8(p+1)} \sum_{q=0}^p \int_{1234} V_{1234} \mathbf{G}_{12}^{(q)} \mathbf{G}_{34}^{(p-q)} . \quad (71)$$

The graphical representation of this result reads

$$\textcircled{p+1} = \frac{1}{24(p+1)} \textcircled{p} + \frac{1}{8(p+1)} \sum_{q=0}^p \textcircled{(q)} \textcircled{(p-q)}. \quad (72)$$

Inserting the diagrams of the connected two- and four-point function from Tab. I and II, we obtain the corresponding vacuum diagrams. For instance, Eq. (72) reduces for $p = 2$

$$\textcircled{3} = \frac{1}{72} \textcircled{2} + \frac{1}{12} \textcircled{(0)} \textcircled{(2)} + \frac{1}{24} \textcircled{(1)} \textcircled{(1)} \quad (73)$$

with the respective terms

$$\textcircled{2} = \frac{3}{2} \textcircled{\triangle} + 2 \textcircled{\square}, \quad (74)$$

$$\textcircled{(0)} \textcircled{(2)} = \frac{1}{6} \textcircled{\square\square} + \frac{1}{4} \textcircled{\square\square\square\square} + \frac{1}{4} \textcircled{\square\square\square\square}, \quad (75)$$

$$\textcircled{(1)} \textcircled{(1)} = \frac{1}{4} \textcircled{\square\square\square\square}. \quad (76)$$

Thus we obtain the connected vacuum diagrams shown in Tab. III for $p = 3$:

$$\textcircled{3} = \frac{1}{48} \textcircled{\triangle} + \frac{1}{24} \textcircled{\square\square} + \frac{1}{32} \textcircled{\square\square\square\square} + \frac{1}{48} \textcircled{\square\square\square\square}. \quad (77)$$

The results (67), (68), and (77) are listed in Table III which shows also one further perturbative order. Note that the weights of the connected vacuum diagrams obey a formula similar to (50)

$$w = \frac{1}{2^{S+D} 3^T 4^F P}, \quad (78)$$

where S, D, T, F denote the number of self-, double, triple, fourfold connections and P stands for the number of vertex permutations leaving the vacuum diagram unchanged [10, 18, 20].

III. ONE-PARTICLE IRREDUCIBLE FEYNMAN DIAGRAMS

So far, we have explained how to generate connected Feynman diagrams of the ϕ^4 -theory. We shall now eliminate from them the reducible contributions. To this end we derive in Subsection III A a closed set of Schwinger-Dyson equations for the one-particle irreducible two- and four-point function. In Subsection III B they are converted into graphical recursion relations for the corresponding one-particle irreducible Feynman diagrams needed for renormalizing the ϕ^4 -theory. Subsection III C discusses how these Feynman diagrams of the one-particle irreducible two- and four-point functions are related to the connected vacuum diagrams which are also one-particle-irreducible.

A. Closed Set of Schwinger-Dyson Equations for One-Particle Irreducible Two- and Four-Point Functions

In this subsection we revisit the functional identity (25) which immediately followed from the definition of the functional integral. We show that it can be also used to derive a closed set of Schwinger-Dyson equations for the one-particle irreducible two- and four-point functions.

1. Field-Theoretic Definitions

From Table I we can distinguish two classes of diagrams contributing to the connected two-point function \mathbf{G}_{12} . The first one contains one-particle irreducible diagrams which remain connected after amputating one arbitrary line. The second one consists of the remaining diagrams which are called reducible. The diagrams # 2.1 and # 2.2, for instance, are one-particle irreducible, whereas # 2.3 is reducible. When considering the self-energy

$$\Sigma_{12} = G_{12}^{-1} - \mathbf{G}_{12}^{-1}, \quad (79)$$

where \mathbf{G}_{12}^{-1} is the functional inverse of \mathbf{G}_{12} :

$$\int_3 \mathbf{G}_{13} \mathbf{G}_{32}^{-1} \equiv \delta_{12}, \quad (80)$$

we will see later on that this quantity is graphically represented by all one-particle irreducible diagrams of the connected two-point function \mathbf{G}_{12} where the external legs are omitted. Reversely, all connected diagrams of the connected two-point function \mathbf{G}_{12} are reconstructed from the one-particle irreducible ones of the self-energy Σ_{12} according to the Dyson equation

$$\mathbf{G}_{12} = G_{12} + \int_{34} G_{13} \Sigma_{34} \mathbf{G}_{42}, \quad (81)$$

which immediately follows from (79) by taking into account (14) and (80). When the self-energy is graphically represented in Feynman diagrams by a big open dot with two legs

$$\Sigma_{12} \equiv 1 - \textcircled{O} - 2, \quad (82)$$

the Dyson equation (81) reads graphically

$$1 \text{ --- } 2 = 1 \text{ --- } 2 + 1 - \textcircled{O} - 2. \quad (83)$$

In a similar way, Table II shows that the diagrams of the connected four-point function \mathbf{G}_{1234} are either one-particle irreducible, as for instance diagram # 2.1, or reducible such as # 2.2. Defining

$$\Gamma_{1234} = - \int_{5678} \mathbf{G}_{5678}^c \mathbf{G}_{51}^{-1} \mathbf{G}_{62}^{-1} \mathbf{G}_{73}^{-1} \mathbf{G}_{84}^{-1}, \quad (84)$$

we will see later on that this quantity consists of all one-particle irreducible diagrams of the connected four-point function \mathbf{G}_{1234} where the external legs are omitted. Therefore this quantity Γ_{1234} is called the one-particle irreducible four-point function. Inverting relation (84) yields

$$\mathbf{G}_{1234}^c = - \int_{5678} \Gamma_{5678} \mathbf{G}_{51} \mathbf{G}_{62} \mathbf{G}_{73} \mathbf{G}_{84}. \quad (85)$$

When the one-particle four-point function is depicted by using a big open dot with four legs

$$-\Gamma_{1234} \equiv \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{O} \\ \diagdown \quad \diagup \\ 1 \quad 3 \\ | \\ 4 \end{array}, \quad (86)$$

the identity (85) reads graphically

$$\begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{O} \\ \diagdown \quad \diagup \\ 1 \quad 3 \\ | \\ 4 \end{array} = \begin{array}{c} 2 \\ \diagup \quad \diagdown \\ \textcircled{O} \\ \diagdown \quad \diagup \\ 1 \quad 3 \\ | \\ 4 \end{array}. \quad (87)$$

In the following we determine a closed set of Schwinger-Dyson equations for the self-energy and the one-particle irreducible four-point function.

2. Self-Energy

Multiplying the identity (25) with \mathbf{G}_{27}^{-1} and integrating with respect to the index 2, we take into account the self-energy (79) as well as the connected four-point function (85). Thus we yield the Schwinger-Dyson equation for the self-energy:

$$\Sigma_{12} = -\frac{1}{2} \int_{34} V_{1234} \mathbf{G}_{34} + \frac{1}{6} \int_{345678} V_{1345} \mathbf{G}_{36} \mathbf{G}_{47} \mathbf{G}_{58} \Gamma_{6782} . \quad (88)$$

Its graphical representation reads

$$1 - \textcircled{O} - 2 = \frac{1}{2} \underset{1}{\textcircled{O}} \underset{2}{\textcircled{O}} + \frac{1}{6} \underset{1}{\textcircled{O}} \underset{2}{\textcircled{O}} . \quad (89)$$

It contains on the right-hand side the connected two-point function which is determined by the Dyson equation (83). Iteratively solving the integral equations (83) and (89) necessitates, however, the knowledge of the one-particle irreducible four-point function Γ_{1234} .

3. One-Particle Irreducible Four-Point Function

In principle, one could determine Γ_{1234} from the self-energy Σ_{12} . To this end we insert relation (31) into the definition (84) of the one-particle irreducible four-point function:

$$\Gamma_{1234} = 2 \int_{5678} \frac{\delta \mathbf{G}_{56}}{\delta G_{78}^{-1}} \mathbf{G}_{51}^{-1} \mathbf{G}_{62}^{-1} \mathbf{G}_{73}^{-1} \mathbf{G}_{84}^{-1} + \mathbf{G}_{13}^{-1} \mathbf{G}_{24}^{-1} + \mathbf{G}_{14}^{-1} \mathbf{G}_{23}^{-1} . \quad (90)$$

The functional derivative on the right-hand side follows from applying a functional derivative with respect to the kernel G^{-1} to the identity (80):

$$\frac{\delta \mathbf{G}_{56}}{\delta G_{78}^{-1}} = - \int_{90} \mathbf{G}_{59} \frac{\delta \mathbf{G}_{90}^{-1}}{\delta G_{78}^{-1}} \mathbf{G}_{06} . \quad (91)$$

Using the definition of the self-energy (79) and Eq. (18), the one-particle irreducible four-point function (90) reduces to

$$\Gamma_{1234} = 2 \int_{56} \mathbf{G}_{35}^{-1} \frac{\delta \Sigma_{12}}{\delta G_{56}^{-1}} \mathbf{G}_{64}^{-1} . \quad (92)$$

Applying again (79) and the functional chain rule (22), this yields

$$-\Gamma_{1234} = 2 \frac{\delta \Sigma_{12}}{\delta G_{34}} - 2 \int_{56} \Sigma_{35} G_{56} \frac{\delta \Sigma_{12}}{\delta G_{64}} - 2 \int_{56} \frac{\delta \Sigma_{12}}{\delta G_{35}} G_{56} \Sigma_{64} + 2 \int_{5678} \Sigma_{35} G_{57} \frac{\delta \Sigma_{12}}{\delta G_{78}} G_{86} \Sigma_{64} , \quad (93)$$

which is represented graphically as

$$\begin{aligned} \text{Diagram with loop at } 1 &= 2 \frac{\delta \underset{1}{\textcircled{O}} \underset{4}{\textcircled{O}}}{\delta \underset{3}{\text{---}} \underset{4}{\text{---}}} + 2 \underset{3}{\text{---}} \underset{5}{\textcircled{O}} \underset{6}{\text{---}} \frac{\delta \underset{1}{\textcircled{O}} \underset{2}{\textcircled{O}}}{\delta \underset{5}{\text{---}} \underset{6}{\text{---}}} \underset{6}{\text{---}} \underset{4}{\textcircled{O}} \\ &\quad - 2 \underset{3}{\text{---}} \underset{5}{\textcircled{O}} \frac{\delta \underset{1}{\textcircled{O}} \underset{2}{\textcircled{O}}}{\delta \underset{5}{\text{---}} \underset{4}{\text{---}}} - 2 \frac{\delta \underset{1}{\textcircled{O}} \underset{2}{\textcircled{O}}}{\delta \underset{3}{\text{---}} \underset{5}{\text{---}}} \underset{5}{\text{---}} \underset{4}{\textcircled{O}} . \end{aligned} \quad (94)$$

Thus amputating a line from the self energy Σ_{12} leads to the one-particle irreducible diagrams of Γ_{1234} . However, this procedure has the disadvantage that the first two terms yield reducible diagrams which are later on removed in the last two terms in (94). As the number of undesired one-particle reducible diagrams occurring at an intermediate step of the calculation increases with the loop order, the procedure of determining Γ_{1234} via relation (94) is quite inefficient. Therefore we aim at deriving another equation for Γ_{1234} whose iterative solution only involves one-particle irreducible diagrams.

Going back to Eq. (92), we insert the Schwinger-Dyson equation (88) for the self-energy and obtain

$$\begin{aligned} \Gamma_{1234} = & V_{1234} - \frac{g}{2} \int_{5678} V_{1256} \mathbf{G}_{57} \mathbf{G}_{68} \Gamma_{7834} - \frac{1}{2} \int_{5678} V_{1356} \mathbf{G}_{57} \mathbf{G}_{68} \Gamma_{7824} \\ & - \frac{1}{2} \int_{5678} V_{1456} \mathbf{G}_{57} \mathbf{G}_{68} \Gamma_{7823} + \frac{1}{2} \int_{567890\bar{1}\bar{2}} V_{5167} \mathbf{G}_{69} \mathbf{G}_{70} \Gamma_{902\bar{1}} \mathbf{G}_{\bar{1}\bar{2}} \Gamma_{\bar{2}348} \mathbf{G}_{85} \\ & + \frac{1}{3} \int_{567890\bar{1}\bar{2}} V_{1567} \mathbf{G}_{58} \mathbf{G}_{69} \mathbf{G}_{70} \frac{\delta \Gamma_{8902}}{\delta G_{\bar{1}\bar{2}}^{-1}} \mathbf{G}_{\bar{1}\bar{3}}^{-1} \mathbf{G}_{\bar{2}\bar{4}}^{-1} . \end{aligned} \quad (95)$$

The last term is still problematic, as inserting the definition (79) of the self-energy would yield again an inefficient equation. This time, however, we can circumvent the inefficiency problem by deriving from (92) the commutator relation

$$\begin{aligned} \int_{78} \frac{\delta \Gamma_{1234}}{\delta G_{78}^{-1}} \mathbf{G}_{75}^{-1} \mathbf{G}_{86}^{-1} - \int_{78} \frac{\delta \Gamma_{1256}}{\delta G_{78}^{-1}} \mathbf{G}_{73}^{-1} \mathbf{G}_{84}^{-1} = & 2 \int_{7890\bar{1}\bar{2}} \mathbf{G}_{57}^{-1} \mathbf{G}_{68}^{-1} \frac{\delta}{\delta G_{78}^{-1}} (\mathbf{G}_{39}^{-1} \mathbf{G}_{40}^{-1}) \mathbf{G}_{9\bar{1}} \mathbf{G}_{0\bar{2}} \Gamma_{\bar{1}\bar{2}12} \\ & - 2 \int_{7890\bar{1}\bar{2}} \mathbf{G}_{37}^{-1} \mathbf{G}_{48}^{-1} \frac{\delta}{\delta G_{78}^{-1}} (\mathbf{G}_{59}^{-1} \mathbf{G}_{60}^{-1}) \mathbf{G}_{9\bar{1}} \mathbf{G}_{0\bar{2}} \Gamma_{\bar{1}\bar{2}12} . \end{aligned} \quad (96)$$

Applying then

$$\frac{\delta \mathbf{G}_{12}^{-1}}{\delta G_{34}^{-1}} = \frac{1}{2} (\delta_{13} \delta_{24} + \delta_{14} \delta_{23}) - \frac{1}{2} \int_{56} \mathbf{G}_{35} \Gamma_{5126} \mathbf{G}_{64} , \quad (97)$$

which follows from inserting (92) in (79), the intermediate commutator relation (96) is converted to the final one

$$\begin{aligned} \int_{78} \frac{\delta \Gamma_{1234}}{\delta G_{78}^{-1}} \mathbf{G}_{75}^{-1} \mathbf{G}_{86}^{-1} - \int_{78} \frac{\delta \Gamma_{1256}}{\delta G_{78}^{-1}} \mathbf{G}_{73}^{-1} \mathbf{G}_{84}^{-1} = & \frac{1}{2} \int_{78} \Gamma_{3457} \mathbf{G}_{78} \Gamma_{8126} + \frac{1}{2} \int_{78} \Gamma_{3467} \mathbf{G}_{78} \Gamma_{8125} \\ & - \frac{1}{2} \int_{78} \Gamma_{5637} \mathbf{G}_{78} \Gamma_{8124} - \frac{1}{2} \int_{78} \Gamma_{5647} \mathbf{G}_{78} \Gamma_{8123} . \end{aligned} \quad (98)$$

Note that the present commutator relation (98) is equivalent to the former one (34) in Section II. Indeed, Eq. (98) follows also from inserting (85) into (34) after a lengthy but straight-forward calculation. Now we can treat the last problematic term in (95) with the commutator relation (98) and thus yield a nonlinear functional integrodifferential equation for the one-particle irreducible four-point function:

$$\begin{aligned} -\Gamma_{1234} = & -V_{1234} + \frac{1}{3} \int_{567890} V_{1567} \mathbf{G}_{58} \mathbf{G}_{69} \mathbf{G}_{70} \frac{\delta \Gamma_{8234}}{\delta G_{90}} + \frac{1}{2} \int_{5678} V_{1256} \mathbf{G}_{57} \mathbf{G}_{68} \Gamma_{7834} \\ & + \frac{1}{2} \int_{5678} V_{1356} \mathbf{G}_{57} \mathbf{G}_{68} \Gamma_{7824} + \frac{1}{2} \int_{5678} V_{1456} \mathbf{G}_{57} \mathbf{G}_{68} \Gamma_{7823} - \frac{1}{6} \int_{567890\bar{1}\bar{2}} V_{5167} \mathbf{G}_{69} \mathbf{G}_{70} \Gamma_{902\bar{1}} \mathbf{G}_{\bar{1}\bar{2}} \Gamma_{\bar{2}348} \mathbf{G}_{85} \\ & - \frac{1}{6} \int_{567890\bar{1}\bar{2}} V_{5167} \mathbf{G}_{69} \mathbf{G}_{70} \Gamma_{903\bar{1}} \mathbf{G}_{\bar{1}\bar{2}} \Gamma_{\bar{2}248} \mathbf{G}_{85} - \frac{1}{6} \int_{567890\bar{1}\bar{2}} V_{5167} \mathbf{G}_{69} \mathbf{G}_{70} \Gamma_{904\bar{1}} \mathbf{G}_{\bar{1}\bar{2}} \Gamma_{\bar{2}238} \mathbf{G}_{85} . \end{aligned} \quad (99)$$

Its graphical representation reads

$$\begin{aligned} \text{Diagram 1} &= \text{Diagram 2} + \frac{1}{3} \text{Diagram 3} + \frac{1}{2} \text{Diagram 4} + \frac{1}{2} \text{Diagram 5} \\ &+ \frac{1}{2} \text{Diagram 6} + \frac{1}{6} \text{Diagram 7} + \frac{1}{6} \text{Diagram 8} + \frac{1}{6} \text{Diagram 9} . \end{aligned} \quad (100)$$

Thus the closed set of Schwinger-Dyson equations for the connected two-point function, the self-energy and the one-particle irreducible four-point function is given by (83), (89), and (100). In the subsequent section we show that its recursive graphical solution leads to all one-particle irreducible Feynman diagrams needed for renormalizing the ϕ^4 -theory.

B. Graphical Recursion Relations

To this end we supplement the perturbative expansion (38) of the connected two-point function with corresponding ones for the self-energy

$$1 - \textcircled{1} - 2 = \sum_{p=1}^{\infty} 1 - \textcircled{(p)} - 2 \quad (101)$$

and for the one-particle irreducible four-point function

$$\begin{array}{c} 2 \\ \diagup \\ \textcircled{1} \\ \diagdown \\ 1 \end{array} \quad 3 = \sum_{p=1}^{\infty} \begin{array}{c} 2 \\ \diagup \\ \textcircled{p} \\ \diagdown \\ 1 \end{array} \quad 3 . \quad (102)$$

As a result we obtain the following closed set of graphical recursion relations:

$$1 - \textcircled{(p)} - 2 = \frac{1}{2} 1 - \textcircled{(p-1)} - 2 + \frac{1}{6} \sum_{q=1}^{p-1} \sum_{r=1}^q \sum_{s=1}^r 1 - \textcircled{(q-r)} - \textcircled{(r-s)} - s - 2 , \quad (103)$$

$$1 - \textcircled{(p)} - 2 = \sum_{q=1}^p 1 - \textcircled{(q)} - \textcircled{(p-q)} - 2 , \quad (104)$$

$$\begin{array}{c} 2 \\ \diagup \\ \textcircled{p+1} \\ \diagdown \\ 1 \end{array} \quad 3 = \frac{1}{3} \sum_{q=1}^p 1 - \textcircled{(p-q)} - 5 - \frac{2}{\delta} \begin{array}{c} 2 \\ \diagup \\ \textcircled{q} \\ \diagdown \\ 5 \end{array} - 6 - \frac{2}{\delta} \begin{array}{c} 2 \\ \diagup \\ \textcircled{(q-r)} \\ \diagdown \\ 1 \end{array} - \textcircled{(q-r)} - 4 - 2 + \frac{1}{2} \sum_{q=1}^p \sum_{r=1}^q \begin{array}{c} 2 \\ \diagup \\ \textcircled{(p-q)} \\ \diagdown \\ 1 \end{array} - \textcircled{(q-r)} - 3 + \frac{1}{2} \sum_{q=1}^p \sum_{r=1}^q \begin{array}{c} 4 \\ \diagup \\ \textcircled{(p-q)} \\ \diagdown \\ 1 \end{array} - \textcircled{(q-r)} - 2 + \frac{1}{2} \sum_{q=1}^p \sum_{r=1}^q \begin{array}{c} 3 \\ \diagup \\ \textcircled{(p-q)} \\ \diagdown \\ 1 \end{array} - \textcircled{(q-r)} - 4 + \frac{1}{6} \sum_{q=2}^p \sum_{r=2}^q \sum_{s=2}^r \sum_{t=1}^{s-1} \sum_{k=0}^{t-1} \begin{array}{c} 2 \\ \diagup \\ \textcircled{(q-r)} \\ \diagdown \\ 1 \end{array} - \textcircled{(k)} - \textcircled{(s-t)} - (r-s) - 3 + \frac{1}{6} \sum_{q=2}^p \sum_{r=2}^q \sum_{s=2}^r \sum_{t=1}^{s-1} \sum_{k=0}^{t-1} \begin{array}{c} 4 \\ \diagup \\ \textcircled{(q-r)} \\ \diagdown \\ 1 \end{array} - \textcircled{(k)} - \textcircled{(s-t)} - (r-s) - 2 + \frac{1}{6} \sum_{q=2}^p \sum_{r=2}^q \sum_{s=2}^r \sum_{t=1}^{s-1} \sum_{k=0}^{t-1} \begin{array}{c} 3 \\ \diagup \\ \textcircled{(q-r)} \\ \diagdown \\ 1 \end{array} - \textcircled{(k)} - \textcircled{(s-t)} - (r-s) - 4 . \quad (105)$$

This is to be solved starting from

$$1 - \textcircled{(0)} - 2 = 1 - \textcircled{1} - 2 , \quad (106)$$

$$\begin{array}{c} 2 \\ \diagup \\ \textcircled{1} \\ \diagdown \\ 1 \end{array} \quad 3 = \begin{array}{c} 2 \\ \diagup \\ \times \\ \diagdown \\ 1 \end{array} \quad 3 . \quad (107)$$

Note that these graphical recursion relations (103)–(107) allow to prove via complete induction that all diagrams which contribute to the self-energy and the one-particle irreducible four-point function are, indeed, one-particle irreducible [19].

The first few perturbative contributions to Σ_{12} and Γ_{1234} are determined as follows. Inserting (106) in (103) and (104) yield for $p = 1$ the self-energy

$$1 - \textcircled{1} - 2 = \frac{1}{2} 1 - \textcircled{1} - 2 , \quad (108)$$

and the connected two-point function

$$1 - \textcircled{(1)} - 2 = 1 - \textcircled{1} - \textcircled{(0)} - 2 = \frac{1}{2} 1 - \textcircled{1} - 2 . \quad (109)$$

As (107) represent a bare vertex with no line, we read off

$$\frac{\delta}{\delta \overline{6-7}} \begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \\ | \\ 5 \end{array} = 0 . \quad (110)$$

the second-order contribution to the one-particle irreducible four-point function follows from (105) to be

$$\begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} = \frac{1}{2} \begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} + \frac{1}{2} \begin{array}{c} 4 \\ \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 3 \end{array} + \frac{1}{2} \begin{array}{c} 3 \\ \text{---} \\ 1 \\ \text{---} \\ 4 \\ | \\ 2 \end{array} . \quad (111)$$

The results in (108), (109), and (111) are then used to determine for $p = 2$ the self-energy (103)

$$1 - \begin{array}{c} \text{---} \\ 2 \\ \text{---} \end{array} 2 = \frac{1}{4} \begin{array}{c} \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 2 \end{array} + \frac{1}{6} \begin{array}{c} \text{---} \\ 1 \\ \text{---} \\ 2 \end{array} \quad (112)$$

and the connected two-point function (104)

$$1 \overline{\overline{2}} 2 = 1 \overline{\overline{2}} 2^{(0)} + 1 \overline{\overline{1}} 2^{(1)} = \frac{1}{4} \begin{array}{c} \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 2 \end{array} + \frac{1}{4} \begin{array}{c} \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 2 \end{array} + \frac{1}{6} \begin{array}{c} \text{---} \\ 1 \\ \text{---} \\ 2 \end{array} . \quad (113)$$

Amputating one line from (111),

$$\begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \\ | \\ 5 \\ \delta \\ \delta \overline{6-7} \end{array} = \frac{1}{2} \frac{\delta}{\delta \overline{6-7}} \left(\begin{array}{c} 2 \\ \text{---} \\ 5 \\ \text{---} \\ 4 \\ | \\ 3 \\ | \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} + \begin{array}{c} 4 \\ \text{---} \\ 5 \\ \text{---} \\ 3 \\ | \\ 2 \\ | \\ 1 \\ \text{---} \\ 2 \\ | \\ 3 \end{array} + \begin{array}{c} 3 \\ \text{---} \\ 5 \\ \text{---} \\ 2 \\ | \\ 1 \\ \text{---} \\ 4 \\ | \\ 2 \end{array} \right) \\ = \frac{1}{2} \left(\begin{array}{c} 2 \\ \text{---} \\ 5 \\ \text{---} \\ 4 \\ | \\ 6 \\ | \\ 7 \\ 3 \\ 4 \end{array} + \begin{array}{c} 2 \\ \text{---} \\ 5 \\ \text{---} \\ 4 \\ | \\ 7 \\ | \\ 6 \\ 3 \\ 4 \end{array} + \begin{array}{c} 4 \\ \text{---} \\ 5 \\ \text{---} \\ 2 \\ | \\ 6 \\ | \\ 7 \\ 3 \\ 2 \end{array} + \begin{array}{c} 4 \\ \text{---} \\ 5 \\ \text{---} \\ 2 \\ | \\ 7 \\ | \\ 6 \\ 3 \\ 2 \end{array} + \begin{array}{c} 3 \\ \text{---} \\ 5 \\ \text{---} \\ 4 \\ | \\ 6 \\ | \\ 7 \\ 2 \\ 3 \end{array} + \begin{array}{c} 3 \\ \text{---} \\ 5 \\ \text{---} \\ 4 \\ | \\ 6 \\ | \\ 7 \\ 2 \\ 3 \end{array} \right) , \quad (114)$$

we find the third-order contribution to the one-particle irreducible four-point function from (105):

$$\begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} = \frac{1}{4} \left(\begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} + \begin{array}{c} 4 \\ \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 3 \end{array} + \begin{array}{c} 3 \\ \text{---} \\ 1 \\ \text{---} \\ 4 \\ | \\ 2 \end{array} \right) \\ + \frac{1}{2} \left(\begin{array}{c} 3 \\ \text{---} \\ 1 \\ \text{---} \\ 4 \\ | \\ 2 \end{array} + \begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} + \begin{array}{c} 4 \\ \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 3 \end{array} + \begin{array}{c} 1 \\ \text{---} \\ 4 \\ \text{---} \\ 2 \\ | \\ 3 \end{array} + \begin{array}{c} 1 \\ \text{---} \\ 3 \\ \text{---} \\ 4 \\ | \\ 2 \end{array} + \begin{array}{c} 1 \\ \text{---} \\ 2 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} \right) \\ + \frac{1}{2} \left(\begin{array}{c} 2 \\ \text{---} \\ 1 \\ \text{---} \\ 3 \\ | \\ 4 \end{array} + \begin{array}{c} 4 \\ \text{---} \\ 1 \\ \text{---} \\ 2 \\ | \\ 3 \end{array} + \begin{array}{c} 3 \\ \text{---} \\ 1 \\ \text{---} \\ 4 \\ | \\ 2 \end{array} \right) . \quad (115)$$

As expected, the diagrams (109) and (113) for the connected two-point function coincide with the ones from Section II B which are shown in Table I. Furthermore, the diagrams (108), (112) and (107), (111), (115) for the self-energy and the one-particle irreducible four-point function are listed in Table IV and V which show all one-particle irreducible diagrams up to the order $p = 4$ irrespective of their spatial indices (compare the discussion before Eq. (49)). Note that these one-particle irreducible Feynman diagrams have to be evaluated in order to determine the critical exponents of the ϕ^4 -theory from renormalizing the field ϕ , the coupling constant g and the mass m^2 . So far, the results are available up to six and partly to seven loops in $d = 3$ [23–25] and up to five loops in $d = 4 - \epsilon$ dimensions whithin the minimal subtraction scheme [18, 26].

C. One-Particle Irreducible Vacuum Diagrams

The vacuum diagrams of ϕ^4 -theory are not only connected but also one-particle irreducible. In this section we elaborate how they can be generated from short-circuiting the external legs of the diagrams of the self-energy and the one-particle irreducible four-point function, respectively.

1. Relation to the Diagrams of the Self-Energy

At first, we consider the approach which is based on the self-energy. Combining the Dyson equation (81) with the perturbative expansion of the connected two-point function (58) and the corresponding one for the self-energy

$$\Sigma_{12} = \sum_{p=1}^{\infty} \Sigma_{12}^{(p)}, \quad (116)$$

we obtain

$$\int_{12} \mathbf{G}_{12}^{(p)} G_{12}^{-1} = \sum_{q=1}^p \int_{12} \Sigma_{12}^{(q)} \mathbf{G}_{12}^{(p-q)}. \quad (117)$$

Thus our previous equation (61) for determining the contributions of the vacuum energy $W^{(p)}$ from the connected two-point function is converted to

$$W^{(p)} = \frac{1}{4p} \sum_{q=1}^p \int_{12} \Sigma_{12}^{(q)} \mathbf{G}_{12}^{(p-q)}. \quad (118)$$

This result reads graphically

$$\text{(circle)}_p = \frac{1}{4p} \sum_{q=1}^p \text{(circle)}_{(p-q)}^q. \quad (119)$$

We consider one example how the diagrams of the self-energy lead to the corresponding diagrams of the vacuum energy. For $p = 3$ Eq. (119) reduces to

$$\text{(circle)}_3 = \frac{1}{12} \text{(circle)}_{(2)}^1 + \frac{1}{12} \text{(circle)}_{(1)}^2 + \frac{1}{12} \text{(circle)}_{(0)}^3, \quad (119)$$

with the respective terms

$$\text{(circle)}_{(2)}^1 = \frac{1}{12} \text{(circle)}_{(2)}^1 + \frac{1}{8} \text{(circle)}_{(2)}^1 + \frac{1}{8} \text{(circle)}_{(2)}^1, \quad (120)$$

$$\text{(circle)}_{(1)}^2 = \frac{1}{12} \text{(circle)}_{(1)}^2 + \frac{1}{8} \text{(circle)}_{(1)}^2, \quad (121)$$

$$\text{(circle)}_{(0)}^3 = \frac{1}{4} \text{(circle)}_{(0)}^3 + \frac{1}{3} \text{(circle)}_{(0)}^3 + \frac{1}{8} \text{(circle)}_{(0)}^3 + \frac{1}{8} \text{(circle)}_{(0)}^3. \quad (122)$$

Thus we reobtain the connected vacuum diagrams (77) shown in Tab. III for $p = 3$.

2. Relation to the Diagrams of the One-Particle Irreducible Four-Point Function

For the sake of completeness we also mention that the vacuum diagrams can be generated from closing the diagrams of the one-particle irreducible four-point function. To this end we insert in the equation (119) determining the contributions of the vacuum energy the recursion relation (103) for the self-energy and yield

$$\text{(circle)}_{p+1} = \frac{1}{8(p+1)} \sum_{q=0}^p \text{(circle)}_{(q)}^q \text{(circle)}_{(p-q)}^0 + \frac{1}{24(p+1)} \sum_{q=1}^p \sum_{r=1}^q \sum_{s=1}^r \sum_{t=1}^s \text{(circle)}_{(s-t)}^t \text{(circle)}_{(r-s)}^{(q-r)} \text{(circle)}_{(p-q)}^0. \quad (123)$$

Note that this equation follows also directly from the previous recursion relation (72) by taking into account the identity (87) and the perturbative expansions (38), (102).

IV. SUMMARY AND OUTLOOK

In this paper we have derived a closed set of Schwinger-Dyson equations in the disordered, symmetric phase of the ϕ^4 -theory. In particular, we supplemented the well-known integral equations (29) and (89) for the connected two-point function and the self-energy by the new functional integrodifferential equations (37) and (100) for the connected and one-particle irreducible four-point function. Their conversion to graphical recursion relations has allowed us to systematically generate the corresponding connected and one-particle irreducible Feynman diagrams. Furthermore, we have discussed how the short-circuiting of their external legs leads to the associated connected vacuum diagrams. In the subsequent paper [21] we shall elaborate how tadpoles and, more generally, corrections to the connected two-point function as well as the vertex can be successively eliminated by introducing higher Legendre transformations [27–29]. This will lead to graphical recursion relations for the skeleton Feynman diagrams in ϕ^4 -theory.

The recursive graphical solution of our closed set of Schwinger-Dyson equations in ϕ^4 -theory is straightforward and has been carried out in this paper up to the forth perturbative order by hand. Our iterative procedure can easily be automatized by computer algebra. In Ref. [10] it was demonstrated that the basic graphical operations as amputating a line or gluing two lines together can be formulated with the help of a unique matrix notation for Feynman diagrams. It would be interesting to compare the efficiency of our Schwinger-Dyson approach of generating Feynman diagrams of n -point functions together with their weights with already existing computer programs such as FEYNARTS [30–32] and QGRAF [33, 34]. Some of them are based on a combinatorial enumeration of all possible ways of connecting vertices by lines according to Feynman’s rules. Others use a systematic generation of homeomorphically irreducible star graphs [35, 36]. The latter approach is quite efficient and popular at higher orders. It has, however, the conceptual disadvantage that it renders at an intermediate stage numerous superfluous diagrams with different vertex degrees which have to be discarded at the end. Further promising methods have been proposed in Refs. [37, 38]. Whereas the first one is based on a bootstrap equation that uses only the free field value of the energy as an input, the second one combines Schwinger-Dyson equations with the two-particle irreducible (“skeleton”) expansion.

We believe that our closed set of Schwinger-Dyson equations for the connected and one-particle irreducible two- and four-point function will turn out to be useful for developing nonperturbative approximations. This may proceed, for instance, as in Ref. [39] which proposes a self-consistent solution of the Dyson equation in ϕ^4 -theory by using a scaling ansatz for the connected two-point function near the phase transition. A similar consideration in Ginzburg-Landau theory has allowed Ref. [40] to analyze the influence of the thermal fluctuations of the order parameter and the vector potential on the superconducting phase transition. A self-consistent solution of our closed Schwinger-Dyson equation may be found via a phenomenological ansatz for the connected and one-particle irreducible two- and four-point function, respectively. Within such an ansatz one has to find how the functional derivative with respect to the free correlation function is approximated, as this operation is crucial in the new Schwinger-Dyson equations (37) and (100).

Acknowledgement

We thank Hagen Kleinert for stimulating discussions and for reading the manuscript.

References

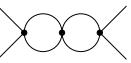
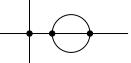
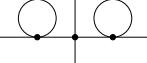
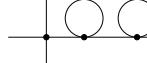
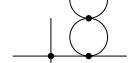
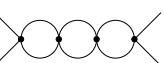
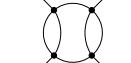
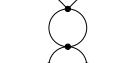
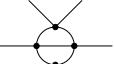
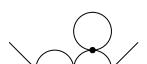
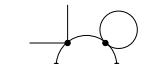
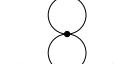
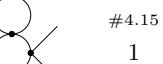
- [1]J.D. Bjorken and S.D. Drell, Vol. I *Relativistic Quantum Mechanics*, Vol. II *Relativistic Quantum Fields* (McGraw-Hill, New York, 1965).
- [2]D.J. Amit, *Field Theory, the Renormalization Group and Critical Phenomena* (McGraw-Hill, New York, 1978).
- [3]M. Le Bellac, *Quantum and Statistical Field Theory* (Oxford Science Publications, Oxford, 1991).
- [4]C. Itzykson and J.-B. Zuber, *Quantum Field Theory* (McGraw-Hill, New York, 1985).
- [5]J. Zinn-Justin, *Quantum Field Theory and Critical Phenomena*, Third Edition (Oxford University Press, Oxford, 1996).
- [6]M.E. Peskin and D.V. Schroeder, *Introduction to Quantum Field Theory* (Addison-Wesley, Reading, 1995).
- [7]L.P. Kadanoff and G. Baym, *Quantum Statistical Mechanics* (Benjamin, Menlo Park, 1962).
- [8]A. Pelster, H. Kleinert, and M. Bachmann, Ann. of Phys. (N.Y.) **297**, 363 (2002).
- [9]M. Bachmann, H. Kleinert, and A. Pelster, Phys. Rev. **D 61**, 085017 (2000).
- [10]H. Kleinert, A. Pelster, B. Kastening, and M. Bachmann, Phys. Rev. **E 62**, 1537 (2000).
- [11]B. Kastening, Phys. Rev. **E 61**, 3501 (2000).
- [12]A. Pelster and H. Kleinert, Physica **A** (in press); e-print: [hep-th/0006153](#).
- [13]H. Kleinert, A. Pelster, and B. Van den Bossche, Physica **A 312**, 141 (2002).
- [14]A. Pelster and K. Glaum, Phys. Stat. Sol. **B** (in press), eprint: [cond-mat/0211361](#).
- [15]A. Pelster and K. Glaum, *Recursive Graphical Construction of Tadpole-Free Feynman Diagrams and Their Weights in ϕ^4 -Theory*; in W. Janke, A. Pelster, H.-J. Schmidt, and M. Bachmann (Editors): *Fluctuating Paths and Fields – Dedicated to Hagen Kleinert on the Occasion of His 60th Birthday* (World Scientific, Singapore, 2001), p. 269; eprint: [hep-th/0105193](#).
- [16]R.F. Streater and A.S. Wightman, *PCT, Spin and Statistics, and All That* (W.A. Benjamin, Reading, Massachusetts, 1964).
- [17]J. Schwinger, *Particles, Sources, and Fields*, Vols. I and II (Addison-Wesley, Reading, 1973).
- [18]H. Kleinert and V. Schulte-Frohlinde, *Critical Properties of ϕ^4 -Theories* (World Scientific, Singapore, 2001).
- [19]K. Glaum, MS Thesis (in German), FU-Berlin (2001).
- [20]J. Neu, MS Thesis (in German), FU-Berlin (1990).
- [21]A. Pelster, K. Glaum, and H. Kleinert, to be published.
- [22]H. Kleinert, *Gauge Fields in Condensed Matter*, Vol. I, *Superflow and Vortex Lines* (World Scientific, Singapore, 1989).
- [23]B.G. Nickel, D.I. Meiron, and G.B. Baker Jr., University of Guelph, preprint (1977), http://www.physik.fu-berlin.de/~kleinert/kleiner_reb8/programs/programs.html.
- [24]S.A. Antonenko and A.I. Sokolov, Phys. Rev. **E 51**, 1894 (1995).
- [25]D.B. Murray and B.G. Nickel, University of Guelph, preprint (1998).
- [26]H. Kleinert, J. Neu, V. Schulte-Frohlinde, K.G. Chetyrkin, and S.A. Larin, Phys. Lett. **B 272**, 39 (1991); **319**, 545 (E) (1993).
- [27]H. Kleinert, Fortschr. Phys. **30**, 187 (1982).
- [28]H. Kleinert, Fortschr. Phys. **30**, 351 (1982).

- [29]A.N. Vasiliev, *Functional Methods in Quantum Field Theory and Statistical Physics* (Gordon and Breach Science Publishers, New York, 1998); translation from the Russian edition (St. Petersburg University Press, St. Petersburg, 1976).
- [30]J. Külbeck, M. Böhm, and A. Denner, Comp. Phys. Comm. **60**, 165 (1991).
- [31]T. Hahn, Comp. Phys. Comm. **140**, 418 (2001).
- [32]<http://www.feynarts.de>.
- [33]P. Nogueira, J. Comput. Phys. **105**, 279 (1993).
- [34]<ftp://gtae2.ist.utl.pt/pub/qgraf>.
- [35]B.R. Heap, J. Math. Phys. **7**, 1582 (1966).
- [36]J.F. Nagle, J. Math. Phys. **7**, 1588 (1966).
- [37]S. Schelstraete and H. Verschelde, Z. Phys. **C 67**, 343 (1995).
- [38]K. Kajantie, M. Laine, and Y. Schröder, Phys. Rev. **D 65**, 045008 (2002).
- [39]A.J. Bray, Phys. Rev. Lett. **32**, 1413 (1974).
- [40]L. Radzikovsky, Europhys. Lett. **29**, 227 (1995).

p	$\overline{\overline{(p)}}$			
0	#0.1 1 (0,0,0,1;2)			
1	#1.1 1/2 (1,0,0,1;2)			
2	#2.1 1/6 (0,0,1,1;2)	#2.2 1/4 (1,1,0,1;2)	#2.3 1/4 (2,0,0,1;2)	
3	#3.1 1/4 (0,2,0,1;2)	#3.2 1/12 (0,0,1,2;2)	#3.3 1/4 (1,1,0,1;2)	#3.4 1/8 (1,2,0,1;2)
	#3.5 1/8 (2,0,0,2;2)	#3.6 1/8 (3,0,0,1;2)	#3.7 1/6 (1,0,1,1;1)	#3.8 1/4 (2,1,0,1;1)
4	#4.1 1/8 (0,3,0,1;2)	#4.2 1/4 (0,1,0,2;2)	#4.3 1/4 (0,2,0,1;2)	#4.4 1/12 (0,1,1,1;2)
	#4.5 1/8 (0,2,0,2;2)	#4.6 1/24 (0,1,1,2;2)	#4.7 1/8 (2,1,0,1;2)	#4.8 1/8 (2,0,0,2;2)
	#4.9 1/8 (1,2,0,1;2)	#4.10 1/2 (1,1,0,1;1)	#4.11 1/8 (1,1,0,2;2)	#4.12 1/12 (1,0,1,1;2)
	#4.13 1/8 (1,2,0,1;2)	#4.14 1/16 (2,1,0,2;2)	#4.15 1/8 (2,1,0,1;2)	#4.16 1/16 (3,0,0,2;2)

	#4.17 1/16 (1,3,0,1;2)	#4.18 1/36 (0,0,2,1;2)	#4.19 1/4 (2,1,0,1;1)	#4.20 1/24 (2,0,1,1;2)
4	#4.21 1/12 (1,0,1,2;1)	#4.22 1/12 (1,1,1,1;1)	#4.23 1/16 (2,2,0,1;2)	#4.24 1/8 (2,2,0,1;1)
	#4.25 1/8 (3,0,0,2;1)	#4.26 1/12 (2,0,1,1;1)	#4.27 1/4 (1,2,0,1;1)	
	#4.28 1/16 (3,1,0,1;2)	#4.29 1/8 (3,1,0,1;1)	#4.30 1/16 (4,0,0,1;2)	

TABLE I: Diagrams of the connected two-point function and their weights of the ϕ^4 -theory up to four loops characterized by the vector $(S, D, T, P; N)$. Its components S, D, T specify the number of self-, double, triple connections, P stands for the number of vertex permutations leaving the diagram unchanged, and N denotes the symmetry degree.

p	
1	#1.1 1  (0,0,0,1;24)
2	#2.1 3/2  (0,1,0,1;8) #2.2 2  (1,0,0,1;6)
3	#3.1 3/4  (0,2,0,1;8) #3.2 3  (0,1,0,1;4) #3.3 3/2  (1,0,0,1;8) #3.4 2/3  (0,0,1,1;6) #3.5 3  (1,1,0,1;2) #3.6 3/2  (2,0,0,1;4) #3.7 1  (2,0,0,1;6) #3.8 1  (1,1,0,1;6)
4	#4.1 3/8  (0,3,0,1;8) #4.2 1  (0,0,0,1;24) #4.3 3/2  (0,2,0,1;4) #4.4 3/4  (0,1,0,2;8) #4.5 3/2  (0,2,0,1;4) #4.6 6  (0,1,0,1;2) #4.7 3/2  (0,2,0,1;4) #4.8 1/2  (0,0,1,1;8) #4.9 3  (1,0,0,1;4) #4.10 3/2  (1,1,0,1;4) #4.11 3  (1,1,0,1;2) #4.12 3/4  (2,0,0,1;8) #4.13 3/4  (1,1,0,1;8) #4.14 3/8  (2,0,0,2;8) #4.15 1  (0,2,0,1;6) #4.16 1  (0,1,1,1;8) #4.17 3  (1,1,0,1;2) #4.18 3  (1,1,0,1;2) #4.19 3  (2,0,0,1;2) #4.20 3/2  (1,2,0,1;2)

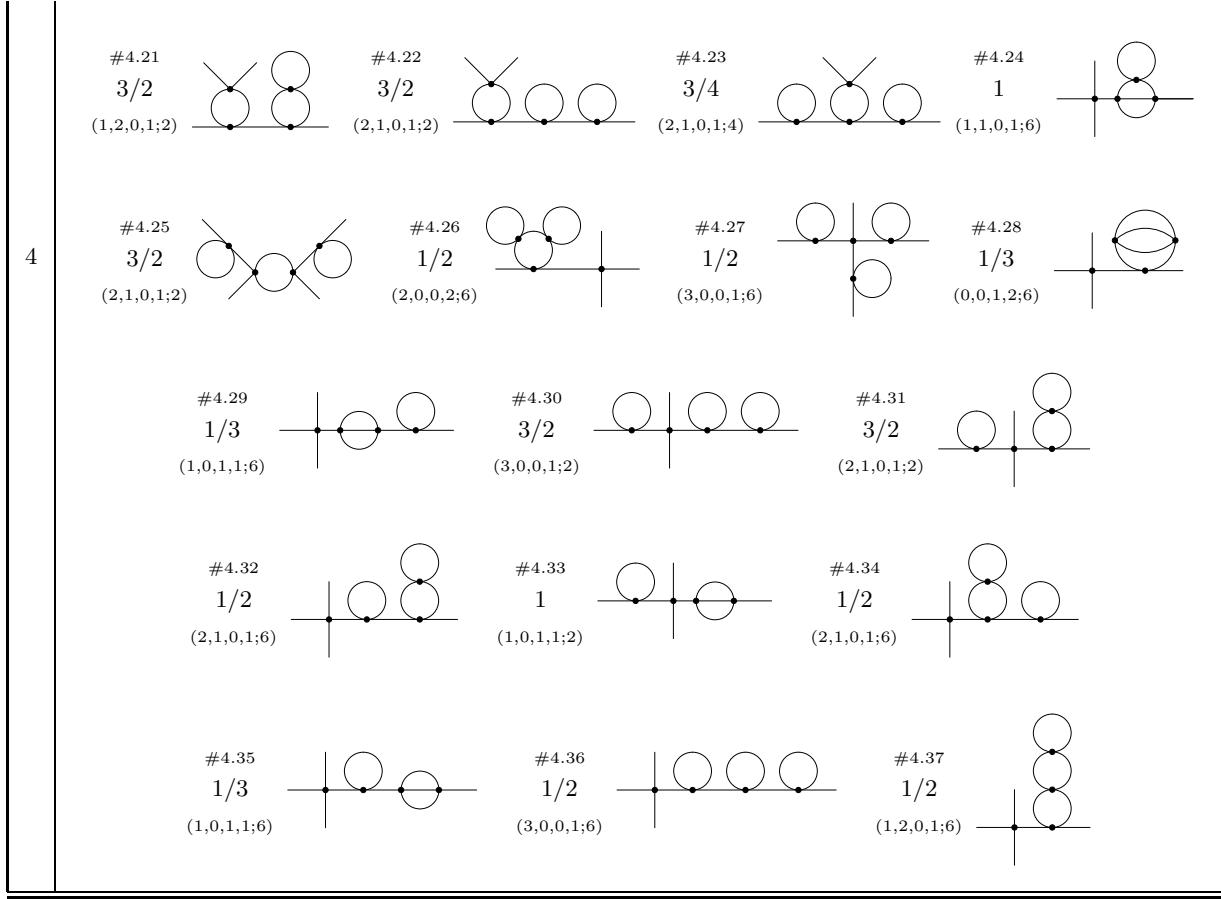


TABLE II: Diagrams of the connected four-point function and their weights of the ϕ^4 -theory up to three loops characterized by the vector $(S, D, T, P; N)$. Its components S, D, T specify the number of self-, double, triple connections, P stands for the number of vertex permutations leaving the diagram unchanged, and N denotes the symmetry degree.

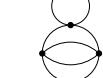
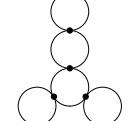
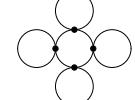
p	p									
1	#1.1 1/8 (2,0,0,0,1)									
2	#2.1 1/48 (0,0,0,1,2)		#2.2 1/16 (2,1,0,0,2)							
3	#3.1 1/48 (0,3,0,0,6)		#3.2 1/24 (1,0,1,0,2)		#3.3 1/48 (3,0,0,0,6)		#3.4 1/32 (2,2,0,0,2)			
4	#4.1 1/128 (0,4,0,0,8)		#4.2 1/32 (0,2,0,0,8)		#4.3 1/144 (0,0,2,0,4)		#4.4 1/16 (1,2,0,0,2)			
	#4.5 1/48 (2,0,1,0,2)		#4.6 1/32 (2,1,0,0,4)		#4.7 1/48 (1,1,1,0,2)		#4.8 1/32 (3,1,0,0,2)			
	#4.9 1/128 (4,0,0,0,8)		#4.10 1/64 (2,3,0,0,2)							

TABLE III: Vacuum diagrams and their weights of the ϕ^4 -theory up to five loops. Each diagram is characterized by the vector (S, D, T, F, P) whose components specify the number of self-, double, triple and fourfold connections, and of the vertex permutations leaving the vacuum diagram unchanged, respectively.

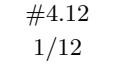
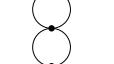
p										
1	#1.1 1/2 (1,0,0,1;2)									
2	#2.1 1/6 (0,0,1,1;2)		#2.2 1/4 (1,1,0,1;2)							
3	#3.1 1/4 (0,2,0,1;2)		#3.2 1/12 (0,0,1,2;2)		#3.3 1/4 (1,1,0,1;2)		#3.4 1/8 (1,2,0,1;2)		#3.5 1/8 (2,0,0,2;2)	
4	#4.1 1/8 (0,3,0,1;2)		#4.2 1/4 (0,1,0,2;2)		#4.3 1/4 (0,2,0,1;2)		#4.4 1/12 (0,1,1,1;2)		#4.5 1/8 (0,2,0,2;2)	
	#4.6 1/24 (0,1,1,2;2)		#4.7 1/8 (2,1,0,1;2)		#4.8 1/8 (2,0,0,2;2)		#4.9 1/8 (1,2,0,1;2)			
	#4.10 1/2 (1,1,0,1;1)		#4.11 1/8 (1,1,0,2;2)		#4.12 1/12 (1,0,1,1;2)		#4.13 1/8 (1,2,0,1;2)			
	#4.14 1/16 (2,1,0,2;2)		#4.15 1/8 (2,1,0,1;2)		#4.16 1/16 (3,0,0,2;2)		#4.17 1/16 (1,3,0,1;2)			

TABLE IV: One-particle irreducible diagrams of the self-energy and their weights of the ϕ^4 -theory up to four loops characterized by the vector $(S, D, T, P; N)$. Its components S, D, T specify the number of self-, double, triple connections, P stands for the number of vertex permutations leaving the diagram unchanged, and N denotes the symmetry degree.

p	
1	#1.1 1 (0,0,0,1;24)
2	#2.1 3/2 (0,1,0,1;8)
3	#3.1 3/4 (0,2,0,1;8) #3.2 3 (0,1,0,1;4) #3.3 3/2 (1,0,0,1;8)
4	#4.1 3/8 (0,3,0,1;8) #4.2 1 (0,0,0,1;24) #4.3 3/2 (0,2,0,1;4) #4.4 3/4 (0,1,0,2;8) #4.5 3/2 (0,2,0,1;4) #4.6 6 (0,1,0,1;2) #4.7 3/2 (0,2,0,1;4) #4.8 1/2 (0,0,1,1;8) #4.9 3 (1,0,0,1;4) #4.10 3/2 (1,1,0,1;4) #4.11 3 (1,1,0,1;2) #4.12 3/4 (2,0,0,1;8) #4.13 3/4 (1,1,0,1;8) #4.14 3/8 (2,0,0,2;8)

TABLE V: Diagrams of the one-particle four-point function and their weights of the ϕ^4 -theory up to four loops characterized by the vector $(S, D, T, P; N)$. Its components S, D, T specify the number of self-, double, triple connections, P stands for the number of vertex permutations leaving the diagram unchanged, and N denotes the symmetry degree.

Theoretical Informatics and Applications

Theoret. Informatics Appl. **37** (2003) 273–299

DOI: 10.1051/ita:2003020

TRACED PREMONOIDAL CATEGORIES

NICK BENTON¹ AND MARTIN HYLAND²

Abstract. Motivated by some examples from functional programming, we propose a generalization of the notion of trace to symmetric premonoidal categories and of Conway operators to Freyd categories. We show that in a Freyd category, these notions are equivalent, generalizing a well-known theorem relating traces and Conway operators in Cartesian categories.

Mathematics Subject Classification. 68N18, 03B70, 03G30.

1. INTRODUCTION

Monads were introduced into computer science by Moggi [25] as a structuring device in denotational semantics and soon became a popular abstraction for writing actual programs, particularly for expressing and controlling side-effects in “pure” functional programming languages such as Haskell [24, 33]. Power and Robinson subsequently introduced *premonoidal categories* as a generalization of Moggi’s computational models [30], whilst Hughes developed *arrows*, which are the equivalent programming abstraction [18].

Some uses of monads in functional programming seem to call for a kind of recursion operator on computations for which, informally, the recursion “only takes place over the values”. For example, the Haskell Prelude defines the (internally implemented) ST and IO monads for, respectively, potentially state-manipulating and input/output-performing computations. These come equipped with polymorphic functions

Keywords and phrases. Traces, fixed point operators, premonoidal categories, recursion, monads.

¹ Microsoft Research, Roger Needham Building, 7 J J Thomson Avenue, Cambridge CB3 0FB, UK; e-mail: nick@microsoft.com

² University of Cambridge, Department of Pure Mathematics and Mathematical Statistics, Wilberforce Road, Cambridge CB3 0WB, UK; e-mail: M.Hyland@dpmms.cam.ac.uk

```
fixST :: (a -> ST a) -> ST a
fixIO :: (a -> IO a) -> IO a
```

which allow computations to be recursively defined in terms of the values they produce. For example, the following program uses `fixIO`¹ to extend a cunning cyclic programming trick due to Bird [1] to the case of side-effecting computations. `replacemin` computes a tree in which every leaf of the argument has been replaced by the minimum of all the leaves. It does this in a single pass over the input and prints out each leaf as it encounters it²:

```
data Tree a = Leaf a | Branch (Tree a) (Tree a)

f :: Tree Int -> Int -> IO (Int,Tree Int)
f (Leaf n) m = do print n
                  return (n, Leaf m)
f (Branch t1 t2) m =
  do (m1,r1) <- f t1 m
     (m2,r2) <- f t2 m
  return (min m1 m2, Branch r1 r2)

replacemin :: Tree Int -> IO (Int, Tree Int)
-- m is argument to and part of the result of f
replacemin t = fixIO (\ ~(m,r) -> f t m)
```

As another (though still somewhat contrived) example, consider modeling the heap of a fictitious pure Scheme-like language at a fairly low level. One might interpret heap-manipulating computations using a monad `T` that is an instance of a type class something like this

```
class Monad T => HeapMonad T where
  alloc :: (Int, Int) -> T Int
  lookup :: Int -> T (Int,Int)
  free   :: Int -> T ()
```

The intention is that `alloc` takes two integers and returns a computation which finds a free cons cell in the heap, fills it with those two integers and returns the (strictly positive) address of the allocated cell. `lookup` takes an integer address and returns the contents of that cons cell, whilst `free` marks a particular address as available for future allocations. Since the values in the *car* and *cdr* of cells can be used as the addresses of other cells, we can interpret programs which build data structures such as lists in the heap. What if the language we are interpreting can create cyclic structures (for example, closures for recursive functions)? At the machine level, cyclic structures are created by allocating cells containing dummy

¹We should note that `fixIO` does not *actually* satisfy the axioms we will propose. However, the basic pattern would remain the same, though the code would be a little longer, if we had performed side-effects involving state instead.

²The tilde `~` on the last line specifies “lazy” pattern matching for the pair `(m,r)`. Haskell’s tuples are actually lifted products and pattern matching is, by default, strict. Without the tilde the function would diverge.

values and then “tying the knot” by overwriting those dummy values with the addresses returned by the allocator. Hence we could just provide destructive update operations

```
setcar :: Int -> Int -> T ()
setcdr :: Int -> Int -> T ()
```

and use those to create cycles. However, if the interpreted language itself does not include destructive assignment, but only creates cycles using higher-level constructs, then adding assignment operations to the monad breaks an abstraction barrier. One solution is to add a recursion operation to the monad

```
fixT :: (a -> T a) -> T a
```

with a definition such that the following code creates a two-element cyclic list (and returns the addresses of both cells):

```
onetwocycle :: T (int,int)
onetwocycle = fixT (\~(x,y)-
    do { x' <- alloc(1,y)
        y' <- alloc(2,x)
        return (x',y')
    })
```

Observe that although the computation is recursively defined, it should only perform the two allocation side-effects once.

Many of the real uses of this kind of recursion have the flavour of the previous example: they involve computations which create cyclic structures for which the identity, order of creation or multiplicity of creation of the objects in the structure is significant. An interesting example arises in work on using Haskell to model hardware. Early versions of both Lava [2] and Hawk [23] specified circuits in a monadic style, instantiating the monad differently for different applications (such as simulating the circuit, generating a netlist or interfacing with a theorem prover). Cyclic circuits (*i.e.* those with feedback) were defined in essentially the style used to define `onetwocycle` above. Lava has moved away from that style, in part because it is syntactically awkward³. Launchbury *et al.* [23] also noted that programming in a monadic style with `fixT` is uncomfortable, and suggested extending Haskell’s `do` notation to allow recursive bindings. That suggestion was followed up by Launchbury and Erkök, who proposed an axiomatization of operators like `fixT` (which they call `mfix`) and showed how the `do` notation can be extended to allow recursive bindings in the case that the underlying monad supports such an `mfix` operation [9, 22].

Launchbury and Erkök’s axiomatization of `mfix` is partly in terms of equations and partly in terms of inequations, intended to be interpreted in the “usual” (slightly informal) concrete domain theoretic model of Haskell. One striking feature of [22] is that it does not appear to build on any of the large body of existing

³Lava now uses a modified version of Haskell with “observable sharing”: allowing new name generation as an implicit side-effect of every expression and hence changing the equational theory of the language [6].

work on axiomatic/categorical treatments of recursion, even those (such as [8]) which consider fixed points in terms of monads. The authors cite some of this work but state, quite correctly, that the non-standard kind of recursion in which they are interested is different from that covered in the literature. Although the presence of a fixpoint object [8], for example, allows an operator with the same type as `mfix` to be defined, it is not of the kind we want.

From a categorical perspective, we seem to want a notion of recursion or feedback on the Kleisli category of a CCC with a strong monad. There is a special case of this situation in which earlier work *does* provide an answer. Although none of Launchbury and Erkök's examples are of commutative monads, in that case the Kleisli category will be symmetric monoidal and Joyal, Street and Verity's notion of *trace* seems to fit the bill [21].

In the general case of a non-commutative monad, however, the Kleisli category will only be symmetric *premonoidal*. The work described here grew firstly from the natural mathematical question of what the right definition of traced premonoidal category might be, and secondly from wondering whether an answer might provide a sensible categorical semantics for the kind of fixpoint operators described in [22]. We give a natural, straightforward and well-behaved answer to the first question, though it only accounts for a rather special subset of the cases considered by Launchbury and Erkök.

2. BACKGROUND

2.1. PREMONOIDAL CATEGORIES

For a careful definition of the notion of (*symmetric*) *premonoidal category* and (*symmetric*) *premonoidal functor*, see Power and Robinson's paper [30]. Briefly, a premonoidal category is a monoidal category except that the tensor product \otimes need only be a functor in each of the two variables separately. Thus if $f : A \rightarrow B$ and $g : A' \rightarrow B'$ in a premonoidal category \mathbb{K} then the two evident morphisms $A \otimes A' \rightarrow B \otimes B'$

$$\begin{aligned} f \times g &= A \otimes A' \xrightarrow{f \otimes A'} B \otimes A' \xrightarrow{B \otimes g} B \otimes B' \\ f \times g &= A \otimes A' \xrightarrow{A \otimes g} A \otimes B' \xrightarrow{f \otimes B'} B \otimes B' \end{aligned}$$

are not generally equal.

We generally write I for the unit of the tensor in a (pre)monoidal category, σ for the symmetry if there is one, and λ , ρ , α for the natural isomorphisms

$$\begin{aligned} \lambda &: I \otimes A \rightarrow A \\ \rho &: A \otimes I \rightarrow A \\ \alpha &: (A \otimes B) \otimes C \rightarrow A \otimes (B \otimes C). \end{aligned}$$

However, since we have coherence theorems for (symmetric) (pre)monoidal categories [30], we will usually elide the structural isomorphisms.

Definition 2.1. A morphism $f : A \rightarrow B$ in a premonoidal category \mathbb{K} is *central* if for all $g : A' \rightarrow B'$ in \mathbb{K} , $f \times g = f \times g$. If at least one of f and g is central, then we may unambiguously write $f \otimes g$. The *centre* $Z(\mathbb{K})$ of a premonoidal category \mathbb{K} is the monoidal subcategory of \mathbb{K} with the same objects but only the central morphisms.

The inclusion functor $Z(\mathbb{K}) \rightarrow \mathbb{K}$ is a strict, identity-on-objects premonoidal functor (and symmetric if \mathbb{K} is). In more recent work Power in particular has stressed the importance from the algebraic point of view in having an explicit choice of centre. That is, one is interested in the situation where one has a functor $J : \mathbb{M} \rightarrow \mathbb{K}$ from a specified (symmetric) monoidal subcategory of a (symmetric) premonoidal \mathbb{K} ; J factors through $Z(\mathbb{K})$, so this amounts to specifying a particular subcategory of central morphisms. (For many results J does not even need to be faithful, but we do not consider that generality here.) We call a $J : \mathbb{M} \rightarrow \mathbb{K}$ as above a *centered premonoidal category*, but since this is our preferred notion we usually drop the “centered”. In this context, by *central* morphisms we shall mean the morphisms of \mathbb{M} . One should think of \mathbb{M} as a category of *values* and \mathbb{K} as a category of possibly-effectful *computations*. An important special case is the following:

Definition 2.2. A *Freyd category* [31] is specified by a Cartesian category \mathbb{C} , a symmetric premonoidal category \mathbb{K} and an identity-on-objects strict symmetric premonoidal functor $J : \mathbb{C} \rightarrow \mathbb{K}$.

Note that morphisms in the specified centre of a Freyd category are “pure” not merely in the sense of commuting with arbitrary effectful computations, but also in being copyable and discardable.

Example 2.3. If T is a strong monad on a symmetric monoidal category \mathbb{M} , then the Kleisli category \mathbb{M}_T is symmetric premonoidal and the canonical functor from \mathbb{M} to \mathbb{M}_T is strict symmetric premonoidal. Thus in the case that \mathbb{M} is Cartesian, we have a Freyd category. If the monad is commutative, then \mathbb{M}_T is symmetric monoidal and $J : \mathbb{M} \rightarrow \mathbb{M}_T$ is strict symmetric monoidal.

2.2. TRACES AND FIXPOINTS

The notion of traced monoidal category was introduced in [21]. The use of traces to interpret recursion in programming languages and the relationship between traces and fixpoints have attracted much attention in recent years, beginning with Hasegawa’s thesis [16]. Categorical axiomatizations of fixpoint operators have been extensively studied, see [4, 8, 27] for example; a particularly crisp and up-to-date account appears in [32].

Definition 2.4. A *trace* on a symmetric monoidal category $(\mathbb{M}, \otimes, I, \lambda, \rho, \alpha, \sigma)$ is a family of functions

$$\text{tr}_{A,B}^U : \mathbb{M}(A \otimes U, B \otimes U) \rightarrow \mathbb{M}(A, B)$$

satisfying the following conditions:

Naturality in A (Left Tightening).

If $f : A' \otimes U \rightarrow B \otimes U$, $g : A \rightarrow A'$ then

$$\text{tr}_{A,B}^U((g \otimes U); f) = g; \text{tr}_{A',B}^U(f) : A \rightarrow B.$$

Naturality in B (Right Tightening).

If $f : A \otimes U \rightarrow B' \otimes U$, $g : B' \rightarrow B$ then

$$\text{tr}_{A,B}^U(f; (g \otimes U)) = \text{tr}_{A,B'}^U(f); g : A \rightarrow B.$$

Dinaturality (Sliding). If $f : A \otimes U \rightarrow B \otimes V$, $g : V \rightarrow U$ then

$$\text{tr}_{A,B}^U(f; (B \otimes g)) = \text{tr}_{A,B}^V((A \otimes g); f) : A \rightarrow B.$$

Action (Vanishing). If $f : A \rightarrow B$ then

$$\text{tr}_{A,B}^I(\rho; f; \rho^{-1}) = f : A \rightarrow B$$

and if $f : A \otimes (U \otimes V) \rightarrow B \otimes (U \otimes V)$ then

$$\text{tr}_{A,B}^{U \otimes V}(f) = \text{tr}_{A,B}^U \left(\text{tr}_{A \otimes U, B \otimes U}^V(\alpha; f; \alpha^{-1}) \right).$$

Superposing. If $f : A \otimes U \rightarrow B \otimes U$ then

$$\begin{aligned} & \text{tr}_{C \otimes A, C \otimes B}^U(\alpha; C \otimes f; \alpha^{-1}) \\ &= C \otimes \text{tr}_{A,B}^U(f) : C \otimes A \rightarrow C \otimes B. \end{aligned}$$

Yanking. For all U , $\text{tr}_{U,U}^U(\sigma_{U,U}) = U : U \rightarrow U$.

Monoidal categories provide a formal basis for reasoning about many of the graphical ‘‘boxes and wires’’ notations used in computer science. A precise foundation for this has been sketched by Hyland and Power [19]. Traced monoidal categories provide a formal basis for circuit-like notations involving feedback or cycles. In one reading at least, they are the building blocks for Girard’s Geometry of Interaction [15]. We recall a result of Joyal, Street and Verity [21]:

Theorem 2.5. *The forgetful 2-functor from the 2-category of compact closed categories to that of traced symmetric monoidal categories has a left biadjoint.*

The basic Geometry of Interaction construction is an explicit form of this biadjoint 2-functor.

We present the trace axioms graphically in Figure 1, though we do not consider the formal semantics of such diagrams here.

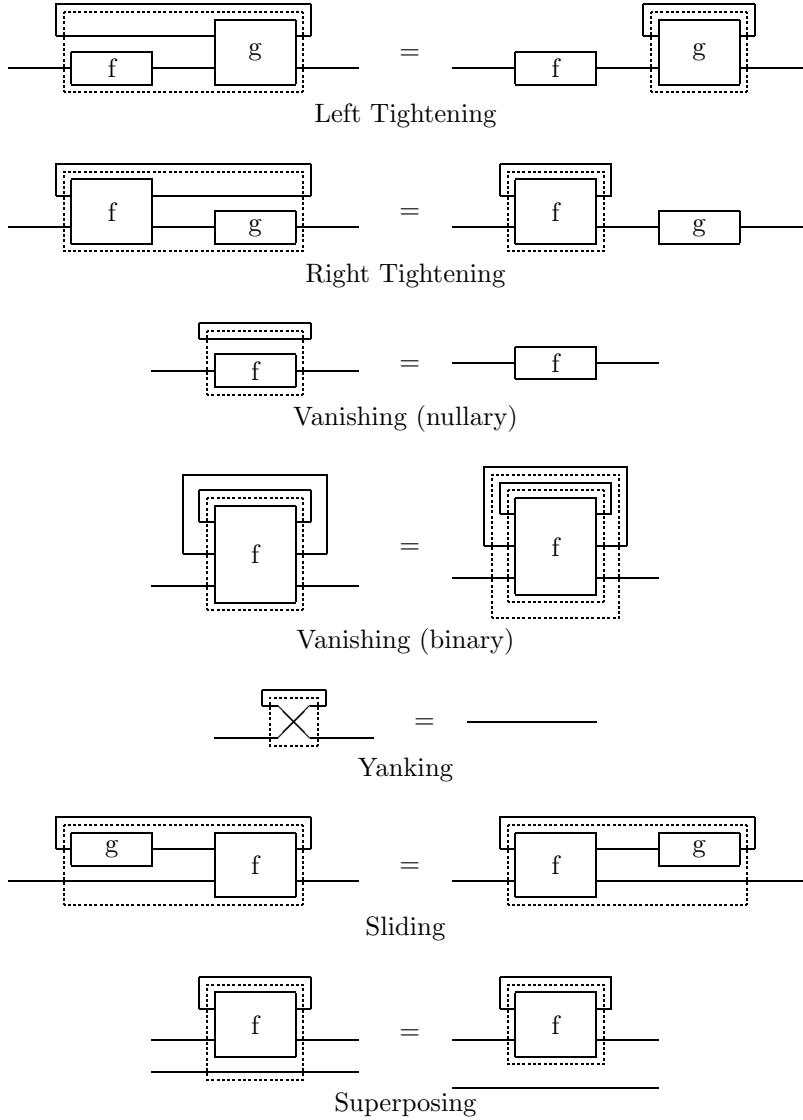


FIGURE 1. Trace axioms.

Definition 2.6. A *parameterized fixpoint operator* on a Cartesian category \mathbb{C} is a family of functions

$$(\cdot)^\dagger : \mathbb{C}(A \times U, U) \rightarrow \mathbb{C}(A, U)$$

satisfying

Naturality. If $f : B \times U \rightarrow U$ and $g : A \rightarrow B$ then

$$g; f^\dagger = ((g \times U); f)^\dagger : A \rightarrow U.$$

Fixed Point Property. If $f : A \times U \rightarrow U$ then

$$\langle A, f^\dagger \rangle; f = f^\dagger : A \rightarrow U.$$

The above definition is rather weak. Well-behaved fixpoint operators typically satisfy other interesting conditions.

Definition 2.7. A *Conway operator* is a parameterized fixpoint operator which additionally satisfies

Dinaturality. If $f : A \times U \rightarrow V$ and $g : V \rightarrow U$ then

$$(f; g)^\dagger = (A \times g; f)^\dagger; g : A \rightarrow U.$$

Diagonal Property. If $f : A \times U \times U \rightarrow U$ then

$$((A \times \Delta); f)^\dagger = (f^\dagger)^\dagger : A \rightarrow U.$$

The Conway operator axioms are shown graphically in Figure 2. Our terminology follows that of Bloom and Ésik [4], though they consider the opposite situation. They work with categories with co-products “generated by a single object”, that is, they consider the opposite of Lawvere theories. For them a Conway theory is the opposite of a Lawvere theory equipped with a Conway operator in our sense. Readers familiar with [4] will observe the following:

- Our *naturality* corresponds to their *parameter identity*;
- Our *fixed point property* corresponds to their *fixed point equation*;
- Our *dinaturality* corresponds to their *simplified composition identity*;
- Our *diagonal property* corresponds to their *double dagger identity*.

In all cases, the extension from Lawvere theories to general Cartesian categories is routine. To the extent that our terminology differs from that of [4], it draws attention to familiar category-theoretic notions.

We make a few remarks on axiomatization. The axioms for a Conway operator imply various other useful properties including the *Bekić property* (allowing simultaneous fixed points to be reduced to sequential ones) and *parameterized dinaturality* (corresponding to the *composition identity* of [4]):

Definition 2.8. A fixpoint operator on a Cartesian category satisfies *parameterized dinaturality* if whenever $f : A \times U \rightarrow V$ and $g : A \times V \rightarrow U$ then

$$\langle A, (\langle \pi_1, g \rangle; f)^\dagger \rangle; g = (\langle \pi_1, f \rangle; g)^\dagger : A \rightarrow U.$$

There are a number of sets of axioms equivalent to Definition 2.7 which can be gleaned from [4], see also [16]. We mention in particular the axiomatization which consists of naturality, parameterized dinaturality and the diagonal property.

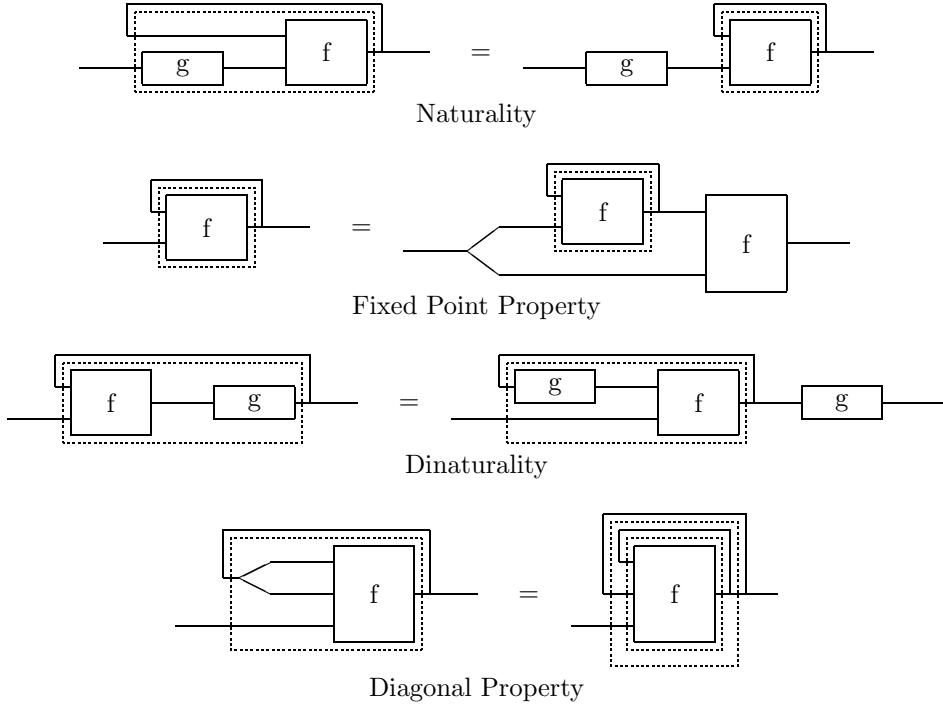


FIGURE 2. Conway axioms.

One focus of other work is the complete axiomatization of identities holding in special classes of models. Conway operators (Conway theories) are a stepping stone on the way. Adding the commutative identities of [4] produces a complete axiomatization, *inter alia*, for the least fixpoint operator in domains. These apparently mysterious identities follow most usefully from “uniformity properties” [4, 11, 12]. Simpson and Plotkin [32] explain the generality of the approach *via* uniformity, which also has other potential uses [17]. We, however, do not consider issues of uniformity further here.

An important theorem about traces and fixpoints is the following, which is due (independently) to Hasegawa and to Hyland, though its essential combinatorial content had been observed earlier in a slightly different context [3–5]:

Theorem 2.9. *To give a trace on a Cartesian category \mathbb{C} is to give a Conway operator on \mathbb{C} .*

For us the notion of trace is paramount. It is completely general and encapsulates the idea of feedback (cyclic graphs) in computer science; and it supports the Geometry of Interaction construction which models the dynamics of proofs.

3. TRACES AND FIXPOINT OPERATORS ON PREMONOIDAL CATEGORIES

So, what is an appropriate generalization of the notions of trace and fixpoint operator to the premonoidal case? We want definitions which make sense, have useful concrete instances, give the monoidal versions as special cases and lead to a generalization of Theorem 2.9.

3.1. SYMMETRIC PREMONOIDAL TRACES

We start by trying to generalize the definition of trace to a centered symmetric premonoidal category $J : \mathbb{M} \rightarrow \mathbb{K}$. Although none of the conditions in Definition 2.4 are expressed in terms of tensoring arbitrary morphisms (in which case we would certainly have to re-examine them), we cannot simply leave the definition unchanged:

Proposition 3.1. *A symmetric premonoidal category with a trace as defined in Definition 2.4 is actually monoidal.*

Proof. Assume $f : A \rightarrow B$ and $g : A' \rightarrow B'$. Then

$$\begin{aligned}
f \times g &= f \otimes A'; B \otimes g \quad (\text{def}) \\
&= f \otimes A'; B \otimes (g; \text{tr}_{B', B'}^{B'}(\sigma)) \quad (\text{yank}) \\
&= f \otimes A'; B \otimes \left(\text{tr}_{A', B'}^{B'}(g \otimes B'; \sigma) \right) \quad (\text{leftt}) \\
&= f \otimes A'; \text{tr}_{B \otimes A', B \otimes B'}^{B'}(\alpha; B \otimes (g \otimes B'; \sigma); \alpha^{-1}) \quad (\text{super}) \\
&= \text{tr}_{A \otimes A', B \otimes B'}^{B'}(((f \otimes A') \otimes B'); \alpha; B \otimes (g \otimes B'; \sigma); \alpha^{-1}) \quad (\text{leftt}) \\
&= \text{tr}_{A \otimes A', B \otimes B'}^{B'}((f \otimes A') \otimes B'; \alpha; B \otimes \sigma; \alpha^{-1}; (B \otimes B') \otimes g) \quad (\text{struct}) \\
&= \text{tr}_{A \otimes A', B \otimes B'}^{A'}((A \otimes A') \otimes g; (f \otimes A') \otimes B'; \alpha; B \otimes \sigma; \alpha^{-1}) \quad (\text{slide}) \\
&= \text{tr}_{A \otimes A', B \otimes B'}^{A'}((A \otimes A') \otimes g; \alpha; A \otimes \sigma; \alpha^{-1}; (f \otimes B') \otimes A') \quad (\text{struct}) \\
&= \text{tr}_{A \otimes A', A \otimes B'}^{A'}((A \otimes A') \otimes g; \alpha; A \otimes \sigma; \alpha^{-1}); f \otimes B' \quad (\text{rightt}) \\
&= \text{tr}_{A \otimes A', A \otimes B'}^{A'}(\alpha; A \otimes (\sigma; (g \otimes A')); \alpha^{-1}); f \otimes B' \quad (\text{struct}) \\
&= A \otimes \text{tr}_{A', B'}^{A'}(\sigma; g \otimes A'); f \otimes B' \quad (\text{super}) \\
&= A \otimes \left(\text{tr}_{A', A'}^{A'}(\sigma); g \right); f \otimes B' \quad (\text{rightt}) \\
&= A \otimes g; f \otimes B' \quad (\text{yank}) \\
&= f \rtimes g \quad (\text{def}). \tag*{\square}
\end{aligned}$$

The key step in the above proof uses the Sliding axiom to commute the side-effects of two computations. This observation motivates the following definition:

Definition 3.2. A *trace* on a centered symmetric premonoidal category $J : \mathbb{M} \rightarrow \mathbb{K}$ is a family of functions

$$\text{tr}_{A,B}^U : \mathbb{K}(A \otimes U, B \otimes U) \rightarrow \mathbb{K}(A, B)$$

satisfying the same conditions given in Definition 2.4 *except* that the Sliding axiom is replaced by

Central Sliding. If $f : A \otimes U \rightarrow B \otimes V$ and if $g : V \rightarrow U$ is a central morphism then

$$\text{tr}_{A,B}^U(f; (B \otimes g)) = \text{tr}_{A,B}^V((A \otimes g); f) : A \rightarrow B$$

and we impose the further requirement

Centre Preservation. If $f : A \otimes U \rightarrow B \otimes U$ is central then so is $\text{tr}_{A,B}^U f : A \rightarrow B$.

The axioms of a premonoidal trace are shown in Figure 3, where we follow Jeffrey [20] in using a heavy line to indicate the sequencing of effects in \mathbb{K} (and that line runs outside those boxes intended to represent central morphisms). Clearly, if $J : \mathbb{M} \rightarrow \mathbb{K}$ has a premonoidal trace on \mathbb{K} , the restriction of that trace to \mathbb{M} is a trace operator in the traditional sense of Definition 2.4. In particular, Definition 3.2 really is a generalization of Definition 2.4.

Requiring the trace to preserve the distinguished centre \mathbb{M} is largely a matter of taste: we prefer to keep our equations algebraic. Even without the condition it is still easy to see that the trace preserves $Z(\mathbb{K})$:

Proposition 3.3. If $f : A \otimes U \rightarrow B \otimes U$ is in $Z(\mathbb{K})$ and $g : C \rightarrow D$ then $g \ltimes \text{tr}_{A,B}^U(f) = g \rtimes \text{tr}_{A,B}^U(f)$.

Proof.

$$\begin{aligned} g \ltimes \text{tr}_{A,B}^U(f) &= g \otimes A; D \otimes \text{tr}_{A,B}^U(f) \\ &= g \otimes A; \text{tr}_{D \otimes A, D \otimes B}^U(D \otimes f) \quad (\text{super}) \\ &= \text{tr}_{C \otimes A, D \otimes B}^U(g \otimes A \otimes U; D \otimes f) \quad (\text{lefttt}) \\ &= \text{tr}_{C \otimes A, D \otimes B}^U(C \otimes f; g \otimes B \otimes U) \quad (\text{centrality}) \\ &= \text{tr}_{C \otimes A, C \otimes B}^U(C \otimes f); g \otimes B \quad (\text{righttt}) \\ &= C \otimes \text{tr}_{A,B}^U(f); g \otimes B \quad (\text{super}) \\ &= g \rtimes \text{tr}_{A,B}^U(f). \end{aligned} \quad \square$$

Thus there is an obvious (non-algebraic) notion of traced premonoidal category. If \mathbb{K} is such and we choose $J : \mathbb{M} \rightarrow \mathbb{K}$ giving a centered premonoidal category then we may fail to get a trace on $J : \mathbb{M} \rightarrow \mathbb{K}$ since \mathbb{M} may fail to be closed under trace (though one can then just close it).

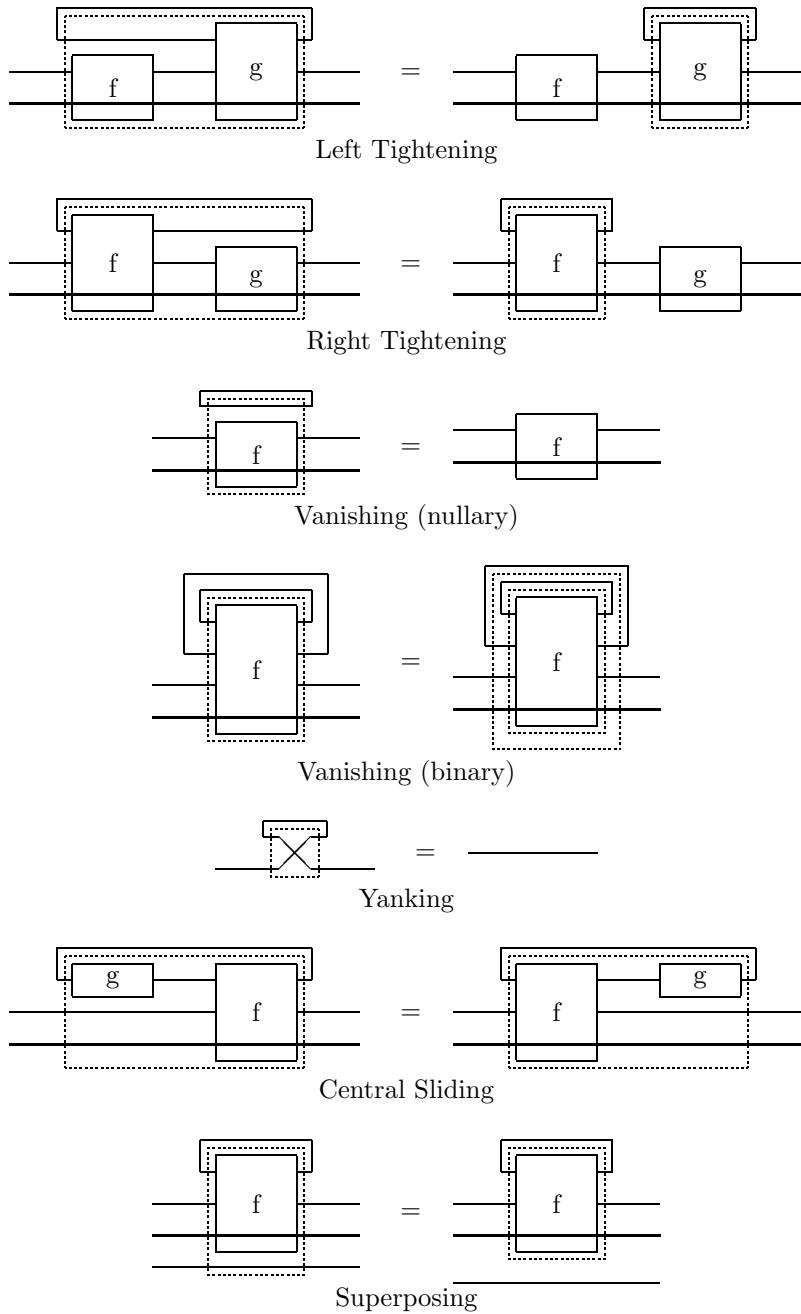


FIGURE 3. Premonoidal trace axioms.

It might also be remarked that the premonoidal sliding condition appears somewhat asymmetric, since it requires that g , rather than one of f and g , be central. However, a little calculation shows that the symmetric case is a consequence:

Proposition 3.4. *Assume $f : A \otimes U \rightarrow B \otimes V$ is central and $g : V \rightarrow U$, then $\text{tr}_{A,B}^V((A \otimes g); f) = \text{tr}_{A,B}^U(f; B \otimes g)$.*

Proof.

$$\begin{aligned}
\text{tr}_{A,B}^V((A \otimes g); f) &= \text{tr}_{A,B}^V((A \otimes g); f); \text{tr}_{B,B}^B(\sigma) \quad (\text{yank}) \\
&= \text{tr}_{A,B}^B(\text{tr}_{A,B}^V((A \otimes g); f) \otimes B; \sigma) \quad (\text{rightt}) \\
&= \text{tr}_{A,B}^B(\sigma; B \otimes \text{tr}_{A,B}^V((A \otimes g); f)) \quad (\text{nat}) \\
&= \text{tr}_{A,B}^B(\sigma; \text{tr}_{B \otimes A, B \otimes B}^V(\alpha; B \otimes ((A \otimes g); f); \alpha^{-1})) \quad (\text{super}) \\
&= \text{tr}_{A,B}^B(\text{tr}_{A \otimes B, B \otimes B}^V(\sigma \otimes V; \alpha; B \otimes ((A \otimes g); f); \alpha^{-1})) \quad (\text{leftt}) \\
&= \text{tr}_{A,B}^{B \otimes V}(\alpha^{-1}; \sigma \otimes V; \alpha; B \otimes (A \otimes g); B \otimes f) \quad (\text{action}) \\
&= \text{tr}_{A,B}^{A \otimes U}(A \otimes f; \alpha^{-1}; \sigma \otimes V; \alpha; B \otimes (A \otimes g)) \\
&\quad (\text{premon sliding, as } B \otimes f \text{ is central}) \\
&= \text{tr}_{A,B}^{A \otimes U}(A \otimes (f; (B \otimes g)); \sigma; \alpha; B \otimes \sigma) \quad (\text{struct}) \\
&= \text{tr}_{A,B}^A(\text{tr}_{A \otimes A, B \otimes A}^U(\alpha; A \otimes (f; B \otimes g); \sigma; \alpha; B \otimes \sigma; \alpha^{-1})) \quad (\text{action}) \\
&= \text{tr}_{A,B}^A(\text{tr}_{A \otimes A, B \otimes A}^U(\alpha; A \otimes (f; B \otimes g); \alpha^{-1}; \sigma \otimes U)) \quad (\text{struct}) \\
&= \text{tr}_{A,B}^A(\text{tr}_{A \otimes A, A \otimes B}^U(\alpha; A \otimes (f; B \otimes g); \alpha^{-1}); \sigma) \quad (\text{rightt}) \\
&= \text{tr}_{A,B}^A(A \otimes \text{tr}_{A,B}^U(f; B \otimes g); \sigma) \quad (\text{super}) \\
&= \text{tr}_{A,B}^A(\sigma; \text{tr}_{A,B}^U(f; B \otimes g) \otimes A) \quad (\text{nat}) \\
&= \text{tr}_{A,A}^A(\sigma); \text{tr}_{A,B}^U(f; B \otimes g) \quad (\text{leftt}) \\
&= \text{tr}_{A,B}^U(f; B \otimes g) \quad (\text{yank}). \tag*{\square}
\end{aligned}$$

3.2. SYMMETRIC PREMONOIDAL FIXPOINTS

We now turn to generalizing the notion of fixpoint operator to the premonoidal case. Since some of the axioms involve duplication and discarding, we will assume that we are working in a Freyd category $J : \mathbb{C} \rightarrow \mathbb{K}$. We also use Δ , π_1 , $\langle \cdot, \cdot \rangle$, etc. as shorthand notation for the lifting of the appropriate operations from \mathbb{C} to \mathbb{K} (*i.e.* we elide uses of J). The notation $\langle f, g \rangle$ is ambiguous unless we specify the order in which the components are computed, but we shall only use it in the case one of the maps is central.

Definition 3.5. A *parameterized fixpoint operator* on a Freyd category $J : \mathbb{C} \rightarrow \mathbb{K}$ is a family of functions

$$(\cdot)^* : \mathbb{K}(A \otimes U, U) \rightarrow \mathbb{K}(A, U)$$

which satisfies:

Centre Preservation. If $f : A \otimes U \rightarrow U$ is central then so is $f^* : A \rightarrow U$.

Naturality. If $f : B \otimes U \rightarrow U$ and $g : A \rightarrow B$ then

$$g; f^* = ((g \otimes U); f)^* : A \rightarrow U.$$

Central Fixed Point Property. If $f : A \otimes U \rightarrow U$ is central, then

$$\langle A, f^* \rangle; f = f^* : A \rightarrow U.$$

Just as in the Cartesian case, this is the bare minimum one might require of a fixpoint operator. We are interested in rather stronger conditions, and propose the following as an appropriate generalization of Conway operators on Cartesian categories:

Definition 3.6. A parameterized fixpoint operator $(\cdot)^*$ on a Freyd category is a *Conway operator* if it satisfies the following conditions.

Parallel Property. If $f : A \otimes U \rightarrow U$ and $g : B \otimes V \rightarrow V$ with one of f and g central then

$$(A \otimes \sigma \otimes V; f \otimes g)^* = f^* \otimes g^* : A \otimes B \rightarrow U \otimes V.$$

Withering Property. If $f : A \otimes U \rightarrow B \otimes U$ and $g : B \rightarrow C$ then

$$(\langle \pi_1, \pi_3 \rangle; f; g \otimes U)^* = (\langle \pi_1, \pi_3 \rangle; f)^*; g \otimes U : A \rightarrow C \otimes U.$$

Diagonal Property. If $f : A \otimes U \otimes U \rightarrow U$ then

$$((A \otimes \Delta); f)^* = (f^*)^* : A \rightarrow U.$$

Dinaturality. If $f : A \otimes U \rightarrow V$ and $g : V \rightarrow U$ with g central then

$$(f; g)^* = (A \otimes g; f)^*; g : A \rightarrow U.$$

The axioms of a premonoidal Conway operator are shown graphically in Figure 4. Dinaturality and the diagonal property are essentially the same as in the Cartesian

case, but the parallel and withering properties are more unusual. There is a natural generalization of parameterized dinaturality to Freyd categories:

Definition 3.7. A parameterized fixpoint operator $(\cdot)^*$ on a Freyd category satisfies *parameterized central dinaturality* if, given $f : A \otimes U \rightarrow V$ and $g : A \otimes V \rightarrow U$ with g central

$$(\langle \pi_1, f \rangle; g)^* = \langle A, (\langle \pi_1, g \rangle; f)^* \rangle; g.$$

As in the Cartesian case, parameterized central dinaturality clearly implies the central fixed point property. But in the case of Freyd categories, parameterized dinaturality does not seem sufficient (along with the diagonal property) to establish the equivalence between traces and Conway operators, which is what motivated our parallel and withering axioms. These do imply parameterized dinaturality, however, though we defer the formal statement of this (Prop. 3.15) as our proof uses the trace axioms.

Furthermore, our definition of a Conway operator on a Freyd category does generalize the standard one:

Proposition 3.8. *Definition 3.6 is equivalent to Definition 2.7 in the case that the category is Cartesian.*

3.3. RELATING FIXPOINTS AND TRACES IN FREYD CATEGORIES

We now show our main result: in a Freyd category, to give a premonoidal trace is equivalent to giving a premonoidal Conway operator.

Theorem 3.9. *Let $J : \mathbb{C} \rightarrow \mathbb{K}$ be a Freyd category such that \mathbb{K} is traced, as in Definition 3.2. Then the operation*

$$(\cdot)^* : \mathbb{K}(A \otimes U, U) \rightarrow \mathbb{K}(A, U)$$

defined by, for $f : A \otimes U \rightarrow U$:

$$f^* = \text{tr}_{A,U}^U(f; \Delta)$$

is a Conway operator in the sense of Definition 3.6.

Proof. Naturality and centre preservation are immediate.

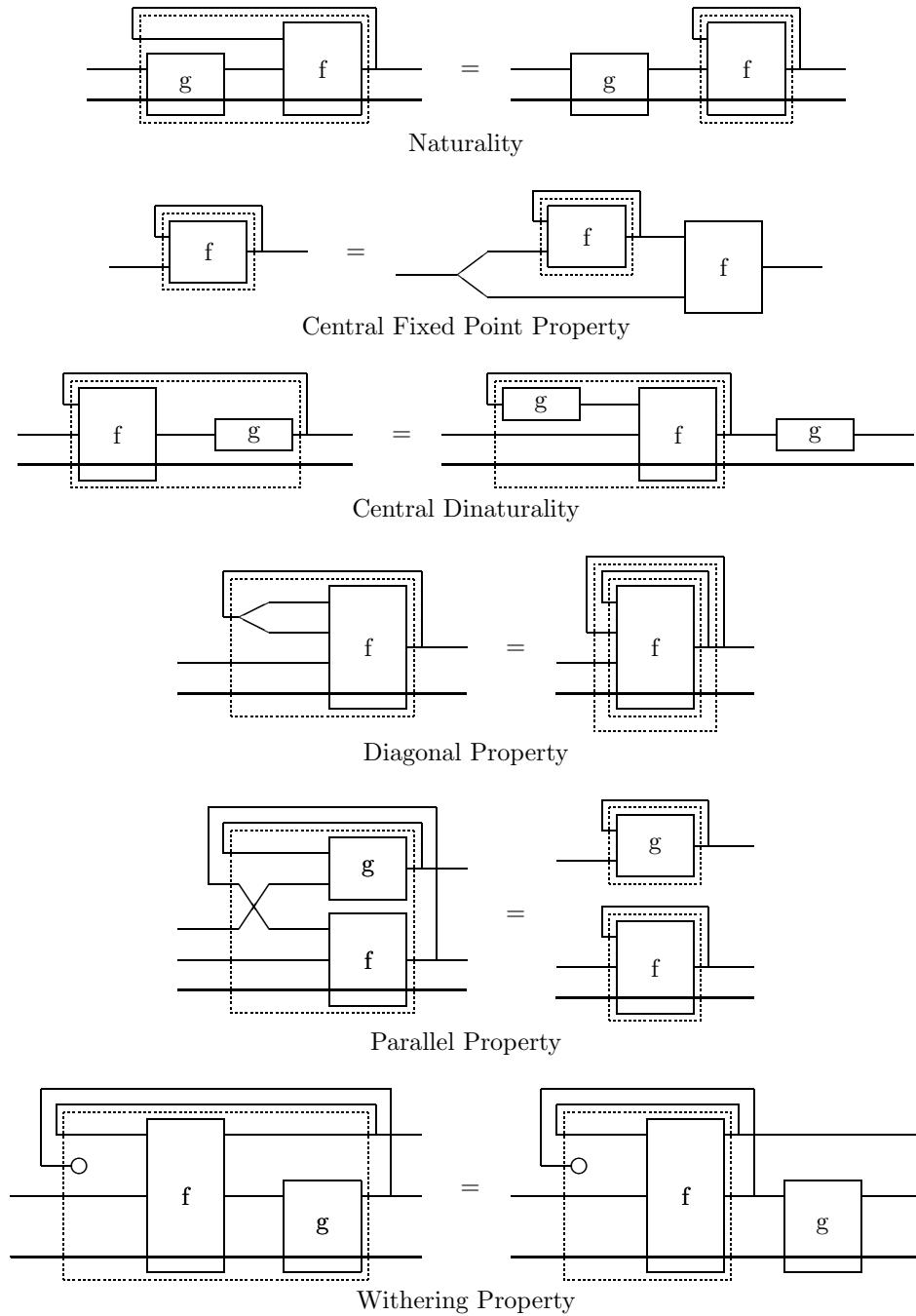


FIGURE 4. Premonoidal conway axioms.

For the central fixed point property, assume that $f : A \otimes U \rightarrow U$ is central. Then

$$\begin{aligned}
\langle A, f^* \rangle; f &= \Delta; A \otimes \text{tr}_{A,U}^U(f; \Delta); f \\
&= \Delta; A \otimes (\text{tr}_{A,A}^A(\sigma); \text{tr}_{A,U}^U(f; \Delta)); f \quad (\text{yank}) \\
&= \Delta; A \otimes (\text{tr}_{A,U}^A(\sigma; \text{tr}_{A,U}^U(f; \Delta) \otimes A)); f \quad (\text{rightt}) \\
&= \Delta; A \otimes (\text{tr}_{A,U}^A(A \otimes \text{tr}_{A,U}^U(f; \Delta); \sigma)); f \\
&= \Delta; A \otimes (\text{tr}_{A,U}^A(\text{tr}_{A \otimes A, A \otimes U}^U(A \otimes f; A \otimes \Delta); \sigma)); f \quad (\text{super}) \\
&= \Delta; A \otimes (\text{tr}_{A,U}^A(\text{tr}_{A \otimes A, A \otimes U}^U(A \otimes f; A \otimes \Delta; \sigma \otimes U))); f \quad (\text{rightt}) \\
&= \Delta; A \otimes (\text{tr}_{A,U}^{A \otimes U}(A \otimes f; A \otimes \Delta; \sigma \otimes U)); f \quad (\text{action}) \\
&= \Delta; A \otimes (\text{tr}_{A,U}^U(A \otimes \Delta; \sigma \otimes U; U \otimes f)); f \quad (\text{slide}) \\
&= \Delta; \text{tr}_{A \otimes A, A \otimes U}^U(A \otimes A \otimes \Delta; A \otimes \sigma \otimes U; A \otimes U \otimes f); f \quad (\text{super}) \\
&= \Delta; \text{tr}_{A \otimes A, U}^U(A \otimes A \otimes \Delta; A \otimes \sigma \otimes U; A \otimes U \otimes f; f \otimes U) \quad (\text{rightt}) \\
&= \text{tr}_{A,U}^U(\Delta \otimes U; A \otimes A \otimes \Delta; A \otimes \sigma \otimes U; A \otimes U \otimes f; f \otimes U) \quad (\text{leftt}) \\
&= \text{tr}_{A,U}^U(\Delta; f \otimes f) \quad (f \text{ central}) \\
&= \text{tr}_{A,U}^U(f; \Delta) \\
&= f^*.
\end{aligned}$$

For central dinaturality, assume $f : A \otimes U \rightarrow V$ and $g : V \rightarrow U$ with g central. Then

$$\begin{aligned}
(f; g)^* &= \text{tr}_{A,U}^U(f; g; \Delta) \\
&= \text{tr}_{A,U}^U(f; \Delta; g \otimes V; U \otimes g) \quad (g \text{ central}) \\
&= \text{tr}_{A,U}^V(A \otimes g; f; \Delta; g \otimes V) \quad (\text{sliding}) \\
&= \text{tr}_{A,U}^V(A \otimes g; f; \Delta); g \quad (\text{rightt}) \\
&= (A \otimes g; f)^*; g.
\end{aligned}$$

For the diagonal property, assume $f : A \otimes U \otimes U \rightarrow U$, then

$$\begin{aligned}
(f^*)^* &= \text{tr}_{A,U}^U(\text{tr}_{A \otimes U, U}^U(f; \Delta); \Delta) \\
&= \text{tr}_{A,U}^U(\text{tr}_{A \otimes U, U \otimes U}^U(f; \Delta; \Delta \otimes U)) \quad (\text{rightt}) \\
&= \text{tr}_{A,U}^{U \otimes U}(f; \Delta; \Delta \otimes U) \quad (\text{action}) \\
&= \text{tr}_{A,U}^{U \otimes U}(f; \Delta; U \otimes \Delta) \quad (\text{struct}) \\
&= \text{tr}_{A,U}^U(A \otimes \Delta; f; \Delta) \quad (\text{slide, } \Delta \text{ central}) \\
&= (A \otimes \Delta; f)^*.
\end{aligned}$$

To show the withering property, assume $f : A \otimes U \rightarrow B \otimes U$ and $g : B \rightarrow C$. Then

$$\begin{aligned}
(\langle \pi_1, \pi_3 \rangle; f; g \otimes U)^* &= \text{tr}_{A,C \otimes U}^{C \otimes U}(A \otimes \pi_2; f; g \otimes U; \Delta) \\
&= \text{tr}_{A,C \otimes U}^U(f; g \otimes U; \Delta; C \otimes U \otimes \pi_2) \quad (\text{slide}) \\
&= \text{tr}_{A,C \otimes U}^U(f; g \otimes U; C \otimes \Delta) \\
&= \text{tr}_{A,C \otimes U}^U(f; B \otimes \Delta; g \otimes U \otimes U) \\
&= \text{tr}_{A,B \otimes U}^U(f; B \otimes \Delta); g \otimes U \quad (\text{rightt}) \\
&= \text{tr}_{A,B \otimes U}^U(f; \Delta; B \otimes U \otimes \pi_2); g \otimes U \\
&= \text{tr}_{A,B \otimes U}^{B \otimes U}(A \otimes \pi_2; f; \Delta); g \otimes U \quad (\text{slide}) \\
&= (\langle \pi_1, \pi_3 \rangle; f)^*; g \otimes U.
\end{aligned}$$

To show the parallel property, assume $f : A \otimes U \rightarrow U$, $g : B \otimes V \rightarrow V$ and, without loss of generality, that f is central. Then

$$\begin{aligned}
&(A \otimes \sigma \otimes V; f \otimes g)^* \\
&= \text{tr}_{A \otimes B, U \otimes V}^{U \otimes V}(A \otimes \sigma \otimes V; f \otimes g; \Delta) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(\text{tr}_{A \otimes B \otimes U, U \otimes V \otimes U}^V(A \otimes \sigma \otimes V; f \otimes g; \Delta)) \quad (\text{action}) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; \text{tr}_{A \otimes U \otimes B, U \otimes V \otimes U}^V(f \otimes B \otimes V; U \otimes g; \Delta)) \quad (\text{leftt}) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \text{tr}_{U \otimes B, U \otimes V \otimes U}^V(U \otimes g; \Delta)) \quad (\text{leftt}) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \text{tr}_{U \otimes B, U \otimes V \otimes U}^V(U \otimes g; \Delta \otimes \Delta; U \otimes \sigma \otimes V)) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \text{tr}_{U \otimes B, U \otimes U \otimes V}^V(U \otimes g; \Delta \otimes \Delta); U \otimes \sigma) \quad (\text{rightt}) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \text{tr}_{U \otimes B, U^2 \otimes V}^V(\Delta \otimes B \otimes V; U^2 \otimes (g; \Delta)); U \otimes \sigma) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \Delta \otimes B; \text{tr}_{U^2 \otimes B, U^2 \otimes V}^V(U \otimes U \otimes (g; \Delta)); U \otimes \sigma) \\
&\quad (\text{leftt}) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \Delta \otimes B; U \otimes U \otimes \text{tr}_{B,V}^V(g; \Delta); U \otimes \sigma) \quad (\text{super}) \\
&= \text{tr}_{A \otimes B, U \otimes V}^U(A \otimes \sigma; f \otimes B; \Delta \otimes B; U \otimes \sigma; U \otimes \text{tr}_{B,V}^V(g; \Delta) \otimes U) \\
&= \text{tr}_{A \otimes B, U \otimes B}^U(A \otimes \sigma; f \otimes B; \Delta \otimes B; U \otimes \sigma); U \otimes \text{tr}_{B,V}^V(g; \Delta) \quad (\text{rightt}) \\
&= \text{tr}_{A \otimes B, U \otimes B}^U(\sigma \otimes U; B \otimes (f; \Delta); \sigma \otimes U); U \otimes \text{tr}_{B,V}^V(g; \Delta) \\
&= \sigma; \text{tr}_{B \otimes A, B \otimes U}^U(B \otimes (f; \Delta)); \sigma; U \otimes \text{tr}_{B,V}^V(g; \Delta) \quad (\text{l/r t}) \\
&= \sigma; B \otimes \text{tr}_{A,B}^U(f; \Delta); \sigma; U \otimes \text{tr}_{B,V}^V(g; \Delta) \quad (\text{super}) \\
&= \text{tr}_{A,B}^U(f; \Delta) \otimes B; U \otimes \text{tr}_{B,V}^V(g; \Delta) \\
&= f^* \otimes g^*. \quad \square
\end{aligned}$$

Remark 3.10. Hasegawa has also given a construction for a fixpoint operator from a trace in the special case of the Kleisli category of a commutative strong monad on a Cartesian category (a case in which the premonoidal structure is

monoidal) [16, Th. 7.2.1]. However, restricting our construction to this special case does not generally give the same fixpoint operator. Hasegawa's construction uses the adjunction in an essential way and repeats side-effects.

Theorem 3.11. *Let $J : \mathbb{C} \rightarrow \mathbb{K}$ be a Freyd category where \mathbb{K} has a Conway operator $(\cdot)^*$ as defined in Definition 3.6. Then the operation*

$$\text{tr}_{A,B}^U(\cdot) : \mathbb{K}(A \otimes U, B \otimes U) \rightarrow \mathbb{K}(A, B)$$

defined by, for $f : A \otimes U \rightarrow B \otimes U$

$$\text{tr}_{A,B}^U(f) = (\langle \pi_1, \pi_3 \rangle; f)^*; \pi_1 : A \rightarrow B$$

is a premonoidal trace in the sense of Definition 3.2.

Proof. Centre preservation is immediate, and left tightening follows directly from naturality.

For right tightening, let $f : A \otimes U \rightarrow B \otimes U$ and $g : B \rightarrow C$. Then

$$\begin{aligned} \text{tr}_{A,C}^U(f; g \otimes U) &= (\langle \pi_1, \pi_3 \rangle; f; g \otimes U)^*; \pi_1 \\ &= (\langle \pi_1, \pi_3 \rangle; f;)^*; g \otimes U; \pi_1 \quad (\text{wither}) \\ &= (\langle \pi_1, \pi_3 \rangle; f;)^*; \pi_1; g \\ &= \text{tr}_{A,B}^U(f); g. \end{aligned}$$

For yanking, observe that $\langle \pi_1, \pi_3 \rangle; \sigma$ is central, then

$$\begin{aligned} \text{tr}_{U,U}^U(\sigma) &= (\langle \pi_1, \pi_3 \rangle; \sigma)^*; \pi_1 \\ &= \langle U, (\langle \pi_1, \pi_3 \rangle; \sigma)^* \rangle; \langle \pi_1, \pi_3 \rangle; \sigma; \pi_1 \quad (\text{cfpp}) \\ &= \langle U, (\langle \pi_1, \pi_3 \rangle; \sigma)^* \rangle; \pi_3 \\ &= (\langle \pi_1, \pi_3 \rangle; \sigma)^*; \pi_2 \\ &= \langle U, (\langle \pi_1, \pi_3 \rangle; \sigma)^* \rangle; \langle \pi_1, \pi_3 \rangle; \sigma; \pi_2 \quad (\text{cfpp}) \\ &= \langle U, (\langle \pi_1, \pi_3 \rangle; \sigma)^* \rangle; \pi_1 \\ &= U. \end{aligned}$$

For the nullary case of action, assume $f : A \rightarrow B$ then

$$\begin{aligned} \text{tr}_{A,B}^I(f \otimes I) &= (\langle \pi_1, \pi_3 \rangle; f \otimes I)^*; \pi_1 \\ &= \langle \pi_1, \pi_3 \rangle^*; f \otimes I; \pi_1 \quad (\text{wither}) \\ &= \langle A, \langle \pi_1, \pi_3 \rangle^* \rangle; \langle \pi_1, \pi_3 \rangle; f \otimes I; \pi_1 \quad (\text{cfpp}) \\ &= \langle f, \langle \pi_1, \pi_3 \rangle^*; \pi_2 \rangle; \pi_1 \\ &= f. \end{aligned}$$

For the binary case of action, assume $f : A \otimes U \otimes V \rightarrow B \otimes U \otimes V$, then

$$\begin{aligned}
\text{tr}_{A,B}^U(\text{tr}_{A \otimes U, B \otimes U}^V(f)) &= ((\langle \pi_1, \pi_3 \rangle; (\langle \pi_1, \pi_2, \pi_5 \rangle; f)^*; \langle \pi_1, \pi_2 \rangle)^*; \pi_1 \\
&= (A \otimes \langle \pi_1, \pi_2 \rangle; \langle \pi_1, \pi_3 \rangle; (\langle \pi_1, \pi_2, \pi_5 \rangle; f)^*)^*; \langle \pi_1, \pi_2 \rangle; \pi_1 \\
&\quad (\text{dinaturality}) \\
&= ((\langle \pi_1, \pi_3 \rangle; (\langle \pi_1, \pi_2, \pi_5 \rangle; f)^*)^*; \pi_1 \\
&= (((\langle \pi_1, \pi_3, \pi_7 \rangle; f)^*)^*; \pi_1 \quad (\text{naturality}) \\
&= (A \otimes \Delta; \langle \pi_1, \pi_3, \pi_7 \rangle; f)^*; \pi_1 \quad (\text{diagonal}) \\
&= ((\langle \pi_1, \pi_3, \pi_4 \rangle; f)^*; \pi_1 \\
&= \text{tr}_{A,B}^{U \otimes V}(f).
\end{aligned}$$

To show sliding, let $f : A \otimes U \rightarrow B \otimes V$ and $g : V \rightarrow U$ with g central. Then

$$\begin{aligned}
\text{tr}_{A,B}^U(f; (B \otimes g)) &= ((\langle \pi_1, \pi_3 \rangle; f; B \otimes g)^*; \pi_1 \\
&= (A \otimes B \otimes g; \langle \pi_1, \pi_3 \rangle; h)^*; B \otimes g; \pi_1 \quad (\text{dinaturality}) \\
&= (A \otimes B \otimes g; \langle \pi_1, \pi_3 \rangle; h)^*; \pi_1 \quad (g \text{ central}) \\
&= ((\langle \pi_1, \pi_3 \rangle; A \otimes g; h)^*; \pi_1 \\
&= \text{tr}_{A,B}^V((A \otimes g); f).
\end{aligned}$$

For superposing, assume $f : A \otimes U \rightarrow B \otimes U$ then

$$\begin{aligned}
\text{tr}_{C \otimes A, C \otimes B}^U(f) &= ((\langle \pi_1, \pi_2, \pi_5 \rangle; C \otimes f)^*; \langle \pi_1, \pi_2 \rangle \\
&= (C \otimes \sigma \otimes B \otimes U; \langle \pi_1, \pi_3, \pi_5 \rangle; C \otimes f)^*; \langle \pi_1, \pi_2 \rangle \\
&= (C \otimes \sigma \otimes B \otimes U; \pi_1 \otimes (\langle \pi_1, \pi_3 \rangle; f))^*; \langle \pi_1, \pi_2 \rangle \\
&= (\pi_1^* \otimes ((\langle \pi_1, \pi_3 \rangle; f)^*); \langle \pi_1, \pi_2 \rangle \quad (\text{parallel}) \\
&= (((C, \pi_1^*); \pi_1) \otimes ((\langle \pi_1, \pi_3 \rangle; f)^*); \langle \pi_1, \pi_2 \rangle \quad (\text{cfpp}) \\
&= (C \otimes ((\langle \pi_1, \pi_3 \rangle; f)^*); \langle \pi_1, \pi_2 \rangle \\
&= C \otimes (((\langle \pi_1, \pi_3 \rangle; f)^*; \pi_1) \\
&= C \otimes \text{tr}_{A,B}^U(f). \quad \square
\end{aligned}$$

Proposition 3.12. *The constructions of trace from Conway operator and of Conway operator from trace given in Theorems 3.11 and 3.9 respectively are mutually inverse.*

Proof. Assume we have a premonoidal Conway operator $(\cdot)^*$. The Conway operator obtained from the trace obtained from $(\cdot)^*$ maps $f : A \otimes U \rightarrow U$ to

$(\langle \pi_1, \pi_3 \rangle; f; \Delta)^*; \pi_1$. Then

$$\begin{aligned} (\langle \pi_1, \pi_3 \rangle; f; \Delta)^*; \pi_1 &= (A \otimes \pi_2; f; \Delta)^*; \pi_1 \\ &= (A \otimes \Delta; A \otimes \pi_2; f)^*; \Delta; \pi_1 \quad (\text{dinaturality}) \\ &= (f)^*; \Delta; \pi_1 \\ &= (f)^* \end{aligned}$$

as required.

Going the other way around, assume we have a premonoidal trace $\text{tr} \cdot$ on a Freyd category. Then we have another trace which maps $h : A \otimes U \rightarrow B \otimes U$ to $\text{tr}_{A, B \otimes U}^{B \otimes U} (A \otimes \pi_2; h; \Delta); \pi_1$ and we calculate

$$\begin{aligned} \text{tr}_{A, B \otimes U}^{B \otimes U} (A \otimes \pi_2; h; \Delta); \pi_1 &= \text{tr}_{A, B \otimes U}^U (h; \Delta; B \otimes U \otimes \pi_2); \pi_1 \quad (\text{sliding}) \\ &= \text{tr}_{A, B \otimes U}^U (h; B \otimes \Delta); \pi_1 \\ &= \text{tr}_{A, B}^U (h; B \otimes \Delta; B \otimes \pi_1) \quad (\text{rightt}) \\ &= \text{tr}_{A, B}^U (h). \end{aligned} \quad \square$$

Thus we have succeeded in establishing a premonoidal generalization of Theorem 2.9.

Remark 3.13. Starting from a fixpoint operator, there is another candidate for the definition of a trace, *viz*

$$\text{tr}'(f) = \langle A, (f; \pi_2)^* \rangle; f; \pi_1 : A \rightarrow B$$

where $f : A \otimes U \rightarrow B \otimes U$. If \mathbb{K} is monoidal this is the same as the construction used in Theorem 3.11, but in the general premonoidal case they are different, and tr' does not seem to have useful properties.

Given the equivalence between traces and Conway operators in Freyd categories, we can now use the trace axioms to prove a useful lemma about Conway operators. We can then prove our earlier claim that our axioms for Conway operators imply the parameterized form of central dinaturality.

Lemma 3.14. *If $h : C \otimes A \otimes U \rightarrow U$ then*

$$(A \otimes h)^* = \Delta \otimes C; A \otimes \sigma; A \otimes h.^*$$

Proof.

$$\begin{aligned}
(A \otimes h)^* &= \text{tr}_{A \otimes C, A \otimes U}^{A \otimes U}(A \otimes h; \Delta) \quad (\text{equivalence}) \\
&= \text{tr}_{A \otimes C, A \otimes U}^A(\text{tr}_{A \otimes C \otimes A, A \otimes U \otimes A}^U(A \otimes h; \Delta)) \quad (\text{action}) \\
&= \text{tr}_{A \otimes C, A \otimes U}^A(\text{tr}_{A \otimes C \otimes A, A \otimes U \otimes A}^U(A \otimes h; A \otimes \Delta; \Delta \otimes U \otimes U; A \otimes \sigma \otimes U)) \\
&= \text{tr}_{A \otimes C, A \otimes U}^A(\text{tr}_{A \otimes C \otimes A, A \otimes U}^U(A \otimes h; A \otimes \Delta); \Delta \otimes U; A \otimes \sigma) \quad (\text{rightt}) \\
&= \text{tr}_{A \otimes C, A \otimes U}^A(A \otimes \text{tr}_{C \otimes A, U}^U(h; \Delta); \Delta \otimes U; A \otimes \sigma) \quad (\text{super}) \\
&= \text{tr}_{A \otimes C, A \otimes U}^A(A \otimes h^*; \Delta \otimes U; A \otimes \sigma) \quad (\text{equivalence}) \\
&= \text{tr}_{A \otimes C, A \otimes U}^A(\Delta \otimes C \otimes A; A \otimes \sigma \otimes A; A \otimes C \otimes \sigma; A \otimes h^* \otimes A) \\
&= \Delta \otimes C; A \otimes \sigma; \text{tr}_{A \otimes C \otimes A, A \otimes C \otimes A}^A(A \otimes C \otimes \sigma); A \otimes h^* \quad (\text{lt/rt}) \\
&= \Delta \otimes C; A \otimes \sigma; A \otimes C \otimes \text{tr}_{A, A}^A(\sigma); A \otimes h^* \quad (\text{super}) \\
&= \Delta \otimes C; A \otimes \sigma; A \otimes h^* \quad (\text{yank}). \tag*{\square}
\end{aligned}$$

Proposition 3.15. *A Conway operator on a Freyd category satisfies parameterized central dinaturality.*

Proof. Assume $f : A \otimes U \rightarrow V$ and $g : A \otimes V \rightarrow U$ with g central. Then

$$\begin{aligned}
(\langle \pi_1, f \rangle; g)^* &= (\Delta \otimes U; A \otimes f; g)^* \\
&= (A \otimes g; \Delta \otimes U; A \otimes f)^*; g \quad (\text{dinaturality}) \\
&= (\Delta \otimes A \otimes V; A \otimes A \otimes g; A \otimes f)^*; g \\
&= \Delta; (A \otimes (A \otimes g; f))^*; g \quad (\text{naturality}) \\
&= \Delta; \Delta \otimes A; A \otimes \sigma; A \otimes (A \otimes g; f)^*; g \quad (\text{Lemma 3.14}) \\
&= \Delta; A \otimes \Delta; A \otimes (A \otimes g; f)^*; g \\
&= \Delta; A \otimes (\Delta \otimes V; A \otimes g; f)^*; g \quad (\text{naturality}) \\
&= \langle A, (\langle \pi_1, g \rangle; f)^* \rangle; g. \tag*{\square}
\end{aligned}$$

4. EXAMPLES

4.1. MONOIDS

Let \mathbb{M} be a traced symmetric monoidal category as in Definition 2.4 and $(M, \mu : M \otimes M \rightarrow M, \eta : I \rightarrow M)$ be a monoid in \mathbb{M} . Let \mathbb{K} be the Kleisli category of the monad $TA = M \otimes A$ on \mathbb{M} , so $\mathbb{K}(A, B) = \mathbb{M}(A, M \otimes B)$. Then the tensor on \mathbb{M} lifts so that $J : \mathbb{M} \rightarrow \mathbb{K}$ is a centered symmetric premonoidal category (it is monoidal iff M is a commutative monoid).

Proposition 4.1. *In the above situation, the operation*

$$\hat{\text{tr}}_{A,B}^U : \mathbb{K}(A \otimes U, B \otimes U) \rightarrow \mathbb{K}(A, B)$$

defined by $\hat{\text{tr}}_{A,B}^U(f) = \text{tr}_{A,M \otimes B}^U(f)$ is a premonoidal trace.

Notions of computation based on monoids are fairly common. Commutative monoids such as the natural numbers under addition can be used for modeling resource usage (*e.g.* timed computations) whereas non-commutative monoids model, for example, side-effecting output. In Haskell syntax, the signature could look like this:

```
class Monoid m where
    mult :: (m,m) -> m
    unit :: m

newtype Cross m a = Cross (m,a) deriving Show

instance Monoid m => Monad (Cross m) where
    return a = Cross (unit,a)
    Cross(m,a) >>= f = let Cross(m',b) = f a
                           in Cross(mult (m,m'), b)

instance Monoid [a] where
    mult (s,t) = s ++ t
    unit = []

-- command which writes to the output
output s = Cross(s,())
```

If we then apply our construction of a premonoidal Conway operator from the trace defined in Proposition 4.1 then we end up with an `mfix` operation of the type described by Launchbury and Erkök:

```
instance Monoid m => MonadRec (Cross m) where
    mfix f = let Cross(m,a) = f a
              in Cross(m,a)
```

And this does have the expected behaviour:

```
nats_output =
    mfix (\ys -> do output "first "
                           output "second."
                           return (0 : map succ ys))

> nats_output
Cross ("first second.",[0,1,2,3,4,5,6,7,8,9,...]
```

The two side effects have happened once only and in the order specified.

4.2. STATE

Let \mathbb{M} be a traced symmetric monoidal category, S be an object of \mathbb{M} and \mathbb{K} be the category with the same objects as \mathbb{M} and $\mathbb{K}(A, B) = \mathbb{M}(S \otimes A, S \otimes B)$ with the evident composition. If \mathbb{M} is closed then \mathbb{K} is equivalent to the Kleisli category of the state monad $TA = S \multimap S \otimes A$. Then $J : \mathbb{M} \rightarrow \mathbb{K}$ is premonoidal.

Proposition 4.2. *In the above situation, the operation*

$$\hat{\text{tr}}_{A,B}^U : \mathbb{K}(A \otimes U, B \otimes U) \rightarrow \mathbb{K}(A, B)$$

defined by $\hat{\text{tr}}_{A,B}^U(f) = \text{tr}_{S \otimes A, S \otimes B}^U(f)$ is a premonoidal trace.

Again, the derived fixed point operator on the Kleisli category is easily defined in Haskell:

```
newtype State s a = State (s -> (s,a))

instance Monad (State s) where
    return a = State (\s ->(s,a))
    State m >>= f = State (\s -> let (s',a) = m s
                                State m' = f a
                                in m' s')

instance MonadRec (State s) where
    mfix f = State (\s -> let State m = f a
                            (s',a) = m s
                            in (s',a))
```

Note how the final value, a is recursively defined, but the final state s' is not – operationally, each time we go around the loop, the initial state is “snapped back” to s .

5. RELATED WORK

Compared with Launchbury and Erkök’s work on axiomatizing `mfix`, our definitions and results are in a rather more general setting, but account for rather fewer concrete examples. The axioms in [22] are almost identical to our definition of a premonoidal Conway operator except that they weaken some of our equations to inequations (interpreted in a concrete category of domains), add a strictness condition on one and regard some as additional properties which may hold in some cases (*i.e.* not part of the basic definition of what they call a “recursive monad”). These weaker conditions admit definitions of `mfix` for monads such as `Maybe` ($1 + (\cdot)$), lazy lists and Haskell’s `IO` monad [10] which do not have Conway operators satisfying our conditions.

Paterson has designed a convenient syntax for programming with Hughes’s arrows (just as Haskell adds `do` to simplify programming with monads) [28, 29].

Paterson gives axioms for an `ArrowLoop` operation which are the same as our definition of a premonoidal trace; our results thus prove an equivalence between `ArrowLoop` and a particular (newly identified) class of `mfixs`.

Jeffrey [20] has also considered a variant of traces in a premonoidal setting, though his construction is rather different from ours: he considers a *partial* trace (only applicable to certain maps) on the category of values rather than on that of computations.

Friedman and Sabry have also investigated value recursion [13], though their approach is rather different from the axiomatic one which we and the others cited here have taken. They view the ability to define computations recursively as an additional effect and give a “monad transformer” which adds a state-based updating implementation of recursion to an arbitrary monad. Lifting the operations of the underlying monad to the new one is left to the programmer (and can generally be done in different ways). Moggi and Sabry have further developed this operational approach [26], giving a semantics for `mfix` in which the side-effecting body is first evaluated (without using the bound variable) and the resulting value is closed up by wrapping it in an ordinary (value) `fix`.

6. CONCLUSIONS AND FURTHER WORK

We have formulated and proved a natural generalization of the theorem relating traces and Conway operators to the case of premonoidal categories. This has applications to the semantics of some non-standard recursion and feedback operations on computations which have been found useful in functional programming.

It would be interesting to see if one could explain Launchbury and Erkök’s weaker axiomatization in a more general setting. The natural thing to try here is to be more explicit about the presence of an abstract lifting monad, along the lines of [14]. This may also help establish a connection with the partial trace operation used by Jeffrey [20]. We would also like to have some more examples.

We are in the process of investigating the premonoidal version of the “geometry of interaction” construction, which traditionally embeds a traced monoidal category into a compact closed one. This is interesting from a mathematical view and may also have some connection to the building of layered protocol stacks from stateful components.

A. WHY “CONWAY” OPERATORS?

We give a brief explanation in category-theoretic terms of the link between Conway operators and the work of John Horton Conway. Consider a category \mathbb{C} with biproducts. The notions of biproduct and of traced symmetric monoidal category are self-dual. It follows therefore from Theorem 2.9 that to give a trace on such a \mathbb{C} is equally to give a Conway operator in our sense and to give one in the dual sense favoured by Bloom and Ésik. Now restrict to the case where \mathbb{C} is a Lawvere theory

(or the opposite of one as in [4], but it amounts to the same thing). Let the generating object be U . A category with biproducts is enriched in commutative monoids (so we can add maps) and it follows that $\text{End}_{\mathbb{C}}(U) = \mathbb{C}(U, U)$ is a rig (ring without negatives) in the terminology promoted by Lawvere and Schanuel. (Bloom and Ésik use the older “semi-ring”.) \mathbb{C} is completely determined by $\text{End}_{\mathbb{C}}(U)$: maps from U^n to U^m are given by $m \times n$ matrices over $\text{End}_{\mathbb{C}}(U)$ (using the usual column vector conventions).

We consider a Conway operator on such a \mathbb{C} , that is, a matrix Conway theory for Bloom and Ésik [4]. They show the following

Theorem A.1. *To give a Conway operator on a Lawvere theory \mathbb{C} with biproducts is to equip the rig $\text{End}_{\mathbb{C}}(U)$ with a unary operator $a \mapsto a^*$ satisfying the Conway identities*

$$\begin{aligned} (ab)^* &= 1 + a(ba)^*b \\ (a+b)^* &= (a^*b)^*a^*. \end{aligned}$$

The Conway identities appear in [7] in the course of an analysis of the theory of regular languages.

Acknowledgements. We thank the referees and Zoltán Ésik for their helpful feedback.

REFERENCES

- [1] R.S. Bird, Using circular programs to eliminate multiple traversals of data. *Acta Informatica* **21** (1984) 239-250.
- [2] P. Bjesse, K. Claessen, M. Sheeran and S. Singh, Lava: Hardware design in Haskell, in *International Conference on Functional Programming* (1998).
- [3] S.L. Bloom and Z. Ésik, Axiomatizing schemes and their behaviors. *J. Comput. Syst. Sci.* **31** (1985) 375-393.
- [4] S.L. Bloom and Z. Ésik, *Iteration Theories*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1993).
- [5] V.E. Cazanescu and Gh. Stefanescu, A general result on abstract flowchart schemes with applications to the study of accessibility, reduction and minimization. *Theor. Comput. Sci.* **99** (1992) 1-63.
- [6] K. Claessen and D. Sands, Observable sharing for functional circuit description, in *Asian Computing Science Conference* (1999).
- [7] J.H. Conway, *Regular Algebra and Finite Machines*. Chapman Hall (1971).
- [8] R.L. Crole and A.M. Pitts, New foundations for fixpoint computations: FIX-hyperdoctrines and the FIX-logic. *Inf. Comput.* **98** (1992) 171-210.
- [9] L. Erköl, *Value Recursion in Monadic Computations*. Ph.D. thesis, Oregon Graduate Institute, OHSU (October 2002).
- [10] L. Erköl, J. Launchbury and A. Moran, Semantics of value recursion for monadic input/output. *RAIRO Theoret. Informatics Appl.* **36** (2002) 155-180.
- [11] Z. Ésik, Axiomatizing iteration categories. *Acta Cybernet.* **14** (1999).
- [12] Z. Ésik, Group axioms for iteration. *Inf. Comput.* **148** (1999) 131-180
- [13] D.P. Friedman and A. Sabry, Recursion is a computational effect. Technical Report 546, Computer Science Department, Indiana University (December 2000).

- [14] C. Fuhrmann, A. Bucalo and A. Simpson, An equational notion of lifting monad. *Theor. Comput. Sci.* **294** (2003) 31-60.
- [15] J.-Y. Girard, Towards a geometry of interaction, in *Categories in Computer Science and Logic*, edited by J.W. Gray and A. Scedrov. *Contemp. Math.* **92** (1989) 69-108.
- [16] M. Hasegawa, *Models of Sharing Graphs (A Categorical Semantics of Let and Letrec)*. Distinguished Dissertations in Computer Science, Springer-Verlag (1999).
- [17] M. Hasegawa, The uniformity principle on traced monoidal categories, edited by R. Blute and P. Selinger. Elsevier, *Electronic Notes in Theor. Comput. Sci.* (2003).
- [18] J. Hughes, Generalising monads to arrows. *Sci. Comput. Program.* **37** (2000) 67-112.
- [19] M. Hyland and A.J. Power, Symmetric monoidal sketches, in *Proc. of the 2nd Conference on Principles and Practice of declarative Programming* (2000) 280-288.
- [20] A. Jeffrey, Premondoidal categories and a graphical view of programs. <http://www.cogs.susx.ac.uk/users/alanje/premon/> (June 1998).
- [21] A. Joyal, R. Street and D. Verity, Traced monoidal categories. *Math. Proc. Cambridge Philoso. Soc.* **119** (1996).
- [22] J. Launchbury and L. Erkök, Recursive monadic bindings, in *International Conference on Functional Programming* (2000).
- [23] J. Launchbury, J.R. Lewis and B. Cook, On embedding a microarchitectural design language within Haskell. *International Conference on Functional Programming* (1999).
- [24] J. Launchbury and S.L. Peyton Jones, State in Haskell. *Lisp and Symbolic Computation* **8** (1995) 293-341.
- [25] E. Moggi, Notions of computation and monads. *Inf. Comput.* **93** (1991) 55-92.
- [26] E. Moggi and A. Sabry, An abstract monadic semantics for value recursion, in *Proc. of the 2003 Workshop on Fixed Points in Computer Science* (April 2003).
- [27] P.S. Mulry, Strong monads, algebras and fixed points, in *Applications of Categories in Computer Science*, edited by M.P. Fournier, P.T. Johnstone and A.M. Pitts. *LMS Lecture Notes* **177** (1992) 202-216.
- [28] R. Paterson, A new notation for arrows, in *Proc. of the International Conference on Functional Programming*. ACM Press (September 2001).
- [29] R. Paterson, Arrows and computation, in *The Fun of Programming*, edited by J. Gibbons and O. de Moor. Palgrave (2003) 201-222.
- [30] A.J. Power and E.P. Robinson, Premonoidal categories and notions of computation. *Math. Struct. Comput. Sci.* **7** (1997) 453-468.
- [31] A.J. Power and H. Thielecke, Closed Freyd and κ -categories. *International Conference on Automata, Languages and Programming* (1999).
- [32] A.K. Simpson and G.D. Plotkin, Complete axioms for categorical fixed-point operators, in *Proc. of 15th Annual Symposium on Logic in Computer Science*. IEEE Computer Society (2000).
- [33] P. Wadler, The essence of functional programming, in *Proc. of the 19th Symposium on Principles of Programming Languages*. ACM (1992).

Received November 12, 2002. Accepted in final form August 14, 2003.

Notions of computation and monads

Eugenio Moggi*

Abstract

The λ -calculus is considered an useful mathematical tool in the study of programming languages, since programs can be *identified* with λ -terms. However, if one goes further and uses $\beta\eta$ -conversion to prove equivalence of programs, then a gross simplification is introduced (programs are identified with total functions from *values* to *values*), that may jeopardise the applicability of theoretical results. In this paper we introduce calculi based on a categorical semantics for *computations*, that provide a correct basis for proving equivalence of programs, for a wide range of *notions of computation*.

Introduction

This paper is about logics for reasoning about programs, in particular for proving equivalence of programs. Following a consolidated tradition in theoretical computer science we identify programs with the closed λ -terms, possibly containing extra constants, corresponding to some features of the programming language under consideration. There are three semantic-based approaches to proving equivalence of programs:

- The **operational** approach starts from an **operational semantics**, e.g. a partial function mapping every program (i.e. closed term) to its resulting value (if any), which induces a congruence relation on open terms called **operational equivalence** (see e.g. [Plo75]). Then the problem is to prove that two terms are operationally equivalent.
- The **denotational** approach gives an interpretation of the (programming) language in a mathematical structure, the **intended model**. Then the problem is to prove that two terms denote the same object in the intended model.
- The **logical** approach gives a class of **possible models** for the (programming) language. Then the problem is to prove that two terms denotes the same object in all possible models.

The operational and denotational approaches give only a theory: the operational equivalence \approx or the set Th of formulas valid in the intended model respectively. On the other hand, the logical approach gives a consequence relation \vdash , namely $Ax \vdash A$ iff the formula A is true in all models of the set of formulas Ax , which can deal with different programming languages (e.g. functional, imperative, non-deterministic) in a rather *uniform* way, by simply changing the set of axioms Ax , and possibly extending the language with new constants. Moreover, the relation \vdash is often semidecidable, so it is possible to give a sound and complete formal system for it, while Th and \approx are semidecidable only in oversimplified cases.

We do not take as a starting point for proving equivalence of programs the theory of $\beta\eta$ -conversion, which identifies the denotation of a program (procedure) of type $A \rightarrow B$ with a total function from A to B , since this identification wipes out completely behaviours like non-termination, non-determinism or side-effects, that can be exhibited by real programs. Instead, we proceed as follows:

1. We take category theory as a general theory of functions and develop on top a **categorical semantics of computations** based on monads.

*Research partially supported by EEC Joint Collaboration Contract # ST2J-0374-C(EDB).

2. We consider simple formal systems matching the categorical semantics of computation.
3. We extend stepwise categorical semantics and formal system in order to interpret richer languages, in particular the λ -calculus.
4. We show that w.l.o.g. one may consider only (monads over) toposes, and we exploit this fact to establish conservative extension results.

The methodology outlined above is inspired by [Sco80]¹, and it is followed in [Ros86, Mog86] to obtain the λ_p -calculus. The view that “category theory comes, logically, before the λ -calculus” led us to consider a categorical semantics of computations first, rather than to modify directly the rules of $\beta\eta$ -conversion to get a *correct* calculus.

Related work

The operational approach to find *correct* λ -calculi w.r.t. an operational equivalence, was first considered in [Plo75] for call-by-value and call-by-name operational equivalence. This approach was later extended, following a similar methodology, to consider other features of computations like nondeterminism (see [Sha84]), side-effects and continuations (see [FFKD86, FF89]). The calculi based only on operational considerations, like the λ_v -calculus, are sound and complete w.r.t. the operational semantics, i.e. a program M has a value according to the operational semantics iff it is provably equivalent to a value (not necessarily the same) in the calculus, but they are too weak for proving equivalences of programs.

Previous work on axiom systems for proving equivalence of programs with side effects has shown the importance of the *let*-constructor (see [Mas88, MT89a, MT89b]). In the framework of the computational lambda-calculus the importance of *let* becomes even more apparent.

The denotational approach may suggest important principles, e.g. fix-point induction (see [Sco69, GMW79]), that can be found only after developing a semantics based on mathematical structures rather than term models, but it does not give clear criteria to single out the general principles among the properties satisfied by the model. Moreover, the theory at the heart of Denotational Semantics, i.e. Domain Theory (see [GS89, Mos89]), has focused on the mathematical structures for giving semantics to recursive definitions of types and functions (see [SP82]), while other structures, that might be relevant to a better understanding of programming languages, have been overlooked. This paper identify one of such structures, i.e. *monads*, but probably there are others just waiting to be discovered.

The categorical semantic of computations presented in this paper has been strongly influenced by the reformulation of Denotational Semantics based on the category of cpos, possibly without bottom, and partial continuous functions (see [Plo85]) and the work on categories of partial morphisms in [Ros86, Mog86]. Our work generalises the categorical account of partiality to other notions of computations, indeed *partial cartesian closed categories* turn out to be a special case of λ_c -models (see Definition 3.9).

A type theoretic approach to partial functions and computations is proposed in [CS87, CS88] by introducing a type-constructor \bar{A} , whose intuitive meaning is the set of *computations* of type A . Our categorical semantics is based on a similar idea. Constable and Smith, however, do not adequately capture the general axioms for computations (as we do), since their notion of model, based on an untyped partial applicative structure, accounts only for partial computations.

1 A categorical semantics of computations

The basic idea behind the categorical semantics below is that, in order to interpret a programming language in a category \mathcal{C} , we distinguish the object A of values (of type A) from the object TA of

¹“I am trying to find out where λ -calculus *should* come from, and the fact that the notion of a cartesian closed category is a late developing one (Eilenberg & Kelly (1966)), is not relevant to the argument: I shall try to explain in my own words in the next section why we should look to it *first*”.

computations (of type A), and take as denotations of programs (of type A) the *elements* of TA . In particular, we identify the type A with the object of values (of type A) and obtain the object of computations (of type A) by applying an unary type-constructor T to A . We call T a *notion of computation*, since it abstracts away from the type of values computations may produce. There are many choices for TA corresponding to different notions of computations.

Example 1.1 We give few notions of computation in the category of sets.

- **partiality** $TA = A_{\perp}$ (i.e. $A + \{\perp\}$), where \perp is the *diverging computation*
- **nondeterminism** $TA = \mathcal{P}_{fin}(A)$
- **side-effects** $TA = (A \times S)^S$, where S is a set of states, e.g. a set U^L of stores or a set of input/output sequences U^*
- **exceptions** $TA = (A + E)$, where E is the set of exceptions
- **continuations** $TA = R^{(R^A)}$, where R is the set of results
- **interactive input** $TA = (\mu\gamma.A + \gamma^U)$, where U is the set of characters.
More explicitly TA is the set of U -branching trees with finite branches and A -labelled leaves
- **interactive output** $TA = (\mu\gamma.A + (U \times \gamma))$.
More explicitly TA is (isomorphic to) $U^* \times A$.

Further examples (in a category of cpos) could be given based on the denotational semantics for various programming languages (see [Sch86, GS89, Mos89]).

Rather than focusing on a specific T , we want to find the general properties common to all notions of computation, therefore we impose as only requirement that *programs* should form a category. The aim of this section is to convince the reader, with a sequence of informal argumentations, that such a requirement amounts to say that T is part of a Kleisli triple $(T, \eta, _^*)$ and that the category of programs is the Kleisli category for such a triple.

Definition 1.2 ([Man76]) A **Kleisli triple** over a category \mathcal{C} is a triple $(T, \eta, _^*)$, where $T: \text{Obj}(\mathcal{C}) \rightarrow \text{Obj}(\mathcal{C})$, $\eta_A: A \rightarrow TA$ for $A \in \text{Obj}(\mathcal{C})$, $f^*: TA \rightarrow TB$ for $f: A \rightarrow TB$ and the following equations hold:

- $\eta_A^* = \text{id}_{TA}$
- $\eta_A; f^* = f$ for $f: A \rightarrow TB$
- $f^*; g^* = (f; g)^*$ for $f: A \rightarrow TB$ and $g: B \rightarrow TC$.

A Kleisli triple satisfies the **mono requirement** provided η_A is mono for $A \in \mathcal{C}$.

Intuitively η_A is the *inclusion* of values into computations (in several cases η_A is indeed a mono) and f^* is the *extension* of a function f from values to computations to a function from computations to computations, which first evaluates a computation and then applies f to the resulting value. In summary

$$\begin{array}{c} a: A \xrightarrow{\eta_A} [a]: TA \\[10pt] \hline a: A \xrightarrow{f} f(a): TB \\ c: TA \xrightarrow{f^*} (\text{let } x \Leftarrow c \text{ in } f(x)): TB \end{array}$$

In order to justify the axioms for a Kleisli triple we have first to introduce a category \mathcal{C}_T whose morphisms correspond to programs. We proceed by analogy with the categorical semantics for terms, where types are interpreted by objects and terms of type B with a parameter (free variable) of type A are interpreted by morphisms from A to B . Since the denotation of programs of type B are supposed to be elements of TB , programs of type B with a parameter of type A ought to be

interpreted by morphisms with codomain TB , but for their domain there are two alternatives, either A or TA , depending on whether parameters of type A are identified with values or computations of type A . We choose the first alternative, because it entails the second. Indeed computations of type A are the same as values of type TA . So we take $\mathcal{C}_T(A, B)$ to be $\mathcal{C}(A, TB)$. It remains to define composition and identities in \mathcal{C}_T (and show that they satisfy the unit and associativity axioms for categories).

Definition 1.3 Given a Kleisli triple $(T, \eta, -^*)$ over \mathcal{C} , the **Kleisli category** \mathcal{C}_T is defined as follows:

- the objects of \mathcal{C}_T are those of \mathcal{C}
- the set $\mathcal{C}_T(A, B)$ of morphisms from A to B in \mathcal{C}_T is $\mathcal{C}(A, TB)$
- the identity on A in \mathcal{C}_T is $\eta_A: A \rightarrow TA$
- $f \in \mathcal{C}_T(A, B)$ followed by $g \in \mathcal{C}_T(B, C)$ in \mathcal{C}_T is $f; g^*: A \rightarrow TC$.

It is natural to take η_A as the identity on A in the category \mathcal{C}_T , since it maps a parameter x to $[x]$, i.e. to x viewed as a computation. Similarly composition in \mathcal{C}_T has a simple explanation in terms of the intuitive meaning of f^* , in fact

$$\frac{x: A \xrightarrow{f} f(x): TB \quad y: B \xrightarrow{g} g(y): TC}{x: A \xrightarrow{f; g^*} (\text{let } y = f(x) \text{ in } g(y)): TC}$$

i.e. f followed by g in \mathcal{C}_T with parameter x is the program which first evaluates the program $f(x)$ and then feed the resulting value as parameter to g . At this point we can give also a simple justification for the three axioms of Kleisli triples, namely they are equivalent to the unit and associativity axioms for \mathcal{C}_T :

- $f; \eta_B^* = f$ for $f: A \rightarrow TB$
- $\eta_A; f^* = f$ for $f: A \rightarrow TB$
- $(f; g^*); h^* = f; (g; h^*)^*$ for $f: A \rightarrow TB$, $g: B \rightarrow TC$ and $h: C \rightarrow TD$.

Example 1.4 We go through the notions of computation given in Example 1.1 and show that they are indeed part of suitable Kleisli triples.

- **partiality** $TA = A_\perp (= A + \{\perp\})$
 η_A is the inclusion of A into A_\perp
if $f: A \rightarrow TB$, then $f^*(\perp) = \perp$ and $f^*(a) = f(a)$ (when $a \in A$)
- **nondeterminism** $TA = \mathcal{P}_{fin}(A)$
 η_A is the singleton map $a \mapsto \{a\}$
if $f: A \rightarrow TB$ and $c \in TA$, then $f^*(c) = \cup_{x \in c} f(x)$
- **side-effects** $TA = (A \times S)^S$
 η_A is the map $a \mapsto (\lambda s: S. \langle a, s \rangle)$
if $f: A \rightarrow TB$ and $c \in TA$, then $f^*(c) = \lambda s: S. (\text{let } \langle a, s' \rangle = c(s) \text{ in } f(a)(s'))$
- **exceptions** $TA = (A + E)$
 η_A is the injection map $a \mapsto \text{inl}(a)$
if $f: A \rightarrow TB$, then $f^*(\text{inr}(e)) = e$ (when $e \in E$) and $f^*(\text{inl}(a)) = f(a)$ (when $a \in A$)
- **continuations** $TA = R^{(R^A)}$
 η_A is the map $a \mapsto (\lambda k: R^A. k(a))$
if $f: A \rightarrow TB$ and $c \in TA$, then $f^*(c) = (\lambda k: R^B. c(\lambda a: A. f(a)(k)))$

- **interactive input** $TA = (\mu\gamma.A + \gamma^U)$
 η_A maps a to the tree consisting only of one leaf labelled with a
if $f: A \rightarrow TB$ and $c \in TA$, then $f^*(c)$ is the tree obtained by replacing leaves of c labelled by a with the tree $f(a)$
- **interactive output** $TA = (\mu\gamma.A + (U \times \gamma))$
 η_A is the map $a \mapsto \langle \epsilon, a \rangle$
if $f: A \rightarrow TB$, then $f^*(\langle s, a \rangle) = \langle s * s', b \rangle$, where $f(a) = \langle s', b \rangle$ and $s * s'$ is the concatenation of s followed by s' .

Kleisli triples are just an alternative description for monads. Although the formers are easy to justify from a computational perspective, the latters are more widely used in the literature on Category Theory and have the advantage of being defined only in terms of functors and natural transformations, which make them more suitable for abstract manipulation.

Definition 1.5 ([Mac71]) *A monad over a category \mathcal{C} is a triple (T, η, μ) , where $T: \mathcal{C} \rightarrow \mathcal{C}$ is a functor, $\eta: \text{Id}_{\mathcal{C}} \rightarrow T$ and $\mu: T^2 \rightarrow T$ are natural transformations and the following diagrams commute:*

$$\begin{array}{ccc} T^3A & \xrightarrow{\mu_{TA}} & T^2A \\ T\mu_A \downarrow & & \downarrow \mu_A \\ T^2A & \xrightarrow{\mu_A} & TA \end{array} \quad \begin{array}{ccc} TA & \xrightarrow{\eta_{TA}} & T^2A \\ & \searrow \text{id}_{TA} & \downarrow \mu_A \\ & TA & \swarrow \text{id}_{TA} \end{array}$$

Proposition 1.6 ([Man76]) *There is a one-one correspondence between Kleisli triples and monads.*

Proof Given a Kleisli triple $(T, \eta, -^*)$, the corresponding monad is (T, η, μ) , where T is the extension of the function T to an endofunctor by taking $T(f) = (f; \eta_B)^*$ for $f: A \rightarrow B$ and $\mu_A = \text{id}_{TA}^*$. Conversely, given a monad (T, η, μ) , the corresponding Kleisli triple is $(T, \eta, -^*)$, where T is the restriction of the functor T to objects and $f^* = (Tf); \mu_B$ for $f: A \rightarrow TB$. ■

Remark 1.7 In general the categorical semantics of partial maps, based on a category \mathcal{C} equipped with a *dominion* \mathcal{M} (see [Ros86]), cannot be reformulated in terms of a Kleisli triple over \mathcal{C} satisfying some additional properties, unless \mathcal{C} has *lifting*, i.e. the inclusion functor from \mathcal{C} into the category of partial maps $P(\mathcal{C}, \mathcal{M})$ has a right adjoint \perp characterised by the natural isomorphism

$$\mathcal{C}(A, B\perp) \cong P(\mathcal{C}, \mathcal{M})(A, B)$$

This mismatch disappears when considering partial cartesian closed categories.

2 Simple languages for monads

In this section we consider two formal systems motivated by different objectives: reasoning about programming languages and reasoning about programs in a fixed programming language. When reasoning about programming languages one has different monads (for simplicity we assume that they are over the same category), one for each programming language, and the main aim is to study how they relate to each other. So it is natural to base a formal system on a *metalanguage* for a category and treat monads as unary type-constructors. When reasoning about programs one has only one monad, because the programming language is fixed, and the main aim is to prove properties of programs. In this case the obvious choice for the term language is the *programming language* itself, which is more naturally interpreted in the Kleisli category.

Remark 2.1 We regard the metalanguage as more fundamental. In fact, its models are more general, as they don't have to satisfy the mono requirement, and the interpretation of programs (of some given programming language) can be defined simply by translation into (a suitable extension of) the metalanguage. It should be pointed out that the mono requirement cannot be axiomatised in the metalanguage, as we would need conditional equations $[x]_T = [y]_T \rightarrow x = y$, and that existence assertions cannot be translated into formulas of the metalanguage, as we would need existentially quantified formulas $(e \downarrow_\sigma)^\circ \equiv (\exists!x: \sigma. e^\circ = [x]_T)^2$.

In Section 2.3 we will explain once for all the correspondence between theories of a simple programming language and categories with a monad satisfying the mono requirement. For other programming languages we will give only their translation in a suitable extension of the metalanguage. In this way, issues like call-by-value versus call-by-name affect the translation, but not the metalanguage.

In Categorical Logic it is common practice to identify a *theory* \mathcal{T} with a category $\mathcal{F}(\mathcal{T})$ with additional structure such that there is a one-one correspondence between *models* of \mathcal{T} in a category \mathcal{C} with additional structure and structure preserving functors from $\mathcal{F}(\mathcal{T})$ to \mathcal{C} (see [KR77])³. This identification was originally proposed by Lawvere, who also showed that algebraic theories can be viewed as categories with finite products.

In Section 2.2 we give a class of theories that can be viewed as categories with a monad, so that any category with a monad is, up to *equivalence* (of categories with a monad), one of such theories. Such a reformulation in terms of theories is more suitable for formal manipulation and more appealing to those unfamiliar with Category Theory. However, there are other advantages in having an alternative presentation of monads. For instance, natural extensions of the syntax may suggest extensions of the categorical structure that may not be immediate to motivate and justify otherwise (we will exploit this in Section 3). In Section 2.3 we take a programming language perspective and establish a correspondence between theories (with equivalence and existence assertions) for a simple programming language and categories with a monad satisfying the mono requirement, i.e. η_A mono for every A .

As starting point we take *many sorted monadic equational logic*, because it is more primitive than many sorted equational logic, indeed monadic theories are equivalent to categories without any additional structure.

2.1 Many sorted monadic equational logic

The language and formal system of many sorted monadic equational logic are parametric in a signature, i.e. a set of base types A and unary function symbols $f: A_1 \rightarrow A_2$. The language is made of types $\vdash A$ type, terms $x: A_1 \vdash e: A_2$ and equations $x: A_1 \vdash e_1 =_{A_2} e_2$ defined by the following formation rules:

$$\begin{array}{c}
A \quad \boxed{\vdash A \text{ type}} \quad \text{A base type} \\
\text{var} \quad \boxed{x: A \vdash x: A} \quad \vdash A \text{ type} \\
f \quad \boxed{x: A \vdash e_1: A_1} \quad \boxed{x: A \vdash f(e_1): A_2} \quad f: A_1 \rightarrow A_2 \\
\text{eq} \quad \boxed{x: A_1 \vdash e_1: A_2} \quad \boxed{x: A_1 \vdash e_2: A_2} \quad \boxed{x: A_1 \vdash e_1 =_{A_2} e_2}
\end{array}$$

²The uniqueness of x s.t. $e^\circ = [x]_T$ follows from the mono requirement.

³In [LS86] a stronger relation is sought between theories and categories with additional structure, namely an equivalence between the category of theories and translations and the category of small categories with additional structure and structure preserving functors. In the case of typed λ -calculus, for instance, such an equivalence between λ -theories and cartesian closed categories requires a modification in the definition of λ -theory, which allows not only equations between λ -terms but also equations between type expressions.

RULE	SYNTAX	SEMANTICS
A	$\vdash A \text{ type}$	$= \llbracket A \rrbracket$
var	$\vdash A \text{ type}$	$= c$
	$x: A \vdash x: A$	$= \text{id}_c$
$f: A_1 \rightarrow A_2$	$x: A \vdash e_1: A_1$ $x: A \vdash f(e_1): A_2$	$= g$ $= g; \llbracket f \rrbracket$
eq	$x: A_1 \vdash e_1: A_2$ $x: A_1 \vdash e_2: A_2$ $x: A_1 \vdash e_1 =_{A_2} e_2$	$= g_1$ $= g_2$ $\iff g_1 = g_2$

Table 1: Interpretation of Many Sorted Monadic Equational Language

Remark 2.2 Terms of (many sorted) monadic equational logic have exactly one free variable (the one declared in the context) which occurs exactly once, and equations are between terms with the same free variable.

An interpretation $\llbracket - \rrbracket$ of the language in a category \mathcal{C} is parametric in an interpretation of the symbols in the signature and is defined by induction on the derivation of well-formedness for (types,) terms and equations (see Table 1) according to the following general pattern:

- the interpretation $\llbracket A \rrbracket$ of a base type A is an object of \mathcal{C}
- the interpretation $\llbracket f \rrbracket$ of an unary function $f: A_1 \rightarrow A_2$ is a morphism from $\llbracket A_1 \rrbracket$ to $\llbracket A_2 \rrbracket$ in \mathcal{C} ; similarly for the interpretation of a term $x: A_1 \vdash e: A_2$
- the interpretation of an assertion $x: A \vdash \phi$ (in this case just an equation) is either true or false.

Remark 2.3 The interpretation of equations is standard. However, if one want to consider more complex assertions, e.g. formulas of first order logic, then they should be interpreted by subobjects; in particular equality $_ = : A$ should be interpreted by the diagonal $\Delta_{\llbracket A \rrbracket}$.

The formal consequence relation on the set of equations is generated by the inference rules for equivalences ((refl), (simm) and (trans)), congruence and substitutivity (see Table 2). This formal consequence relation is *sound and complete* w.r.t. interpretation of the language in categories, i.e. an equation is formally derivable from a set of equational axioms if and only if all the interpretations satisfying the axioms satisfy the equation. Soundness follows from the admissibility of the inference rules in any interpretation, while completeness follows from the fact that any theory \mathcal{T} (i.e. a set of equations closed w.r.t. the inference rules) is the set of equations satisfied by the *canonical interpretation* in the category $\mathcal{F}(\mathcal{T})$, i.e. \mathcal{T} viewed as a category.

Definition 2.4 Given a monadic equational theory \mathcal{T} , the category $\mathcal{F}(\mathcal{T})$ is defined as follows:

- objects are (base) types A ,
- morphisms from A_1 to A_2 are equivalence classes $[x: A_1 \vdash e: A_2]_{\mathcal{T}}$ of terms w.r.t. the equivalence relation induced by the theory \mathcal{T} , i.e.

$$(x: A_1 \vdash e_1: A_2) \equiv (x: A_1 \vdash e_2: A_2) \iff (x: A_1 \vdash e_1 =_{A_2} e_2) \in \mathcal{T}$$

refl	$\frac{x: A \vdash e: A_1}{x: A \vdash e =_{A_1} e}$
symm	$\frac{x: A \vdash e_1 =_{A_1} e_2}{x: A \vdash e_2 =_{A_1} e_1}$
trans	$\frac{x: A \vdash e_1 =_{A_1} e_2 \quad x: A \vdash e_2 =_{A_1} e_3}{x: A \vdash e_2 =_{A_1} e_3}$
congr	$\frac{x: A \vdash e_1 =_{A_1} e_2}{x: A \vdash f(e_1) =_{A_2} f(e_2)} \quad f: A_1 \rightarrow A_2$
subst	$\frac{x: A \vdash e: A_1 \quad x: A_1 \vdash \phi}{x: A \vdash [e/x]\phi}$

Table 2: Inference Rules of Many Sorted Monadic Equational Logic

- *composition is substitution, i.e.*

$$[x: A_1 \vdash e_1: A_2]_{\mathcal{T}}; [x: A_2 \vdash e_2: A_3]_{\mathcal{T}} = [x: A_1 \vdash [e_1/x]e_2: A_3]_{\mathcal{T}}$$

- *identity over A is $[x: A \vdash x: A]_{\mathcal{T}}$.*

There is also a correspondence in the opposite direction, namely every category \mathcal{C} (with additional structure) can be viewed as a theory $\mathcal{T}_{\mathcal{C}}$ (i.e. the *theory* of \mathcal{C} over the language for \mathcal{C}), so that \mathcal{C} and $\mathcal{F}(\mathcal{T}_{\mathcal{C}})$ are equivalent as categories (with additional structure). Actually, in the case of monadic equational theories and categories, \mathcal{C} and $\mathcal{F}(\mathcal{T}_{\mathcal{C}})$ are isomorphic.

In the sequel we consider other equational theories. They can be viewed as categories in the same way described above for monadic theories; moreover, these categories are equipped with additional structure, depending on the specific nature of the theories under consideration.

2.2 The Simple metalanguage

We extend many sorted monadic equational logic to match categories equipped with a monad (or equivalently a Kleisli triple). Although we consider only one monad, it is conceptually straightforward to have several monads at once.

The first step is to extend the language. This could be done in several ways without affecting the correspondence between theories and monads, we choose a presentation inspired by Kleisli triples, more specifically we introduce an unary type-constructor T and the two term-constructors, $[.]$ and let , used informally in Section 1. The definition of signature is slightly modified, since the domain and codomain of an unary function symbol $f: \tau_1 \rightarrow \tau_2$ can be any type, not just base types (the fact is that in many sorted monadic logic the only types are base types). An interpretation $\llbracket \cdot \rrbracket$ of the language in a category \mathcal{C} with a Kleisli triple $(T, \eta, _^*)$ is parametric in an interpretation of the symbols in the signature and is defined by induction on the derivation of well-formedness for types, terms and equations (see Table 3). Finally we add to many sorted monadic equational logic appropriate inference rules capturing axiomatically the properties of the new type- and term-constructors after interpretation (see Table 4).

Proposition 2.5 *Every theory \mathcal{T} of the simple metalanguage, viewed as a category $\mathcal{F}(\mathcal{T})$, is equipped with a Kleisli triple $(T, \eta, _^*)$:*

- $T(\tau) = T\tau,$
- $\eta_{\tau} = [x: \tau \vdash_{ml} [x]_T: T\tau]_{\mathcal{T}},$
- $([x: \tau_1 \vdash_{ml} e: T\tau_2]_{\mathcal{T}})^* = [x': T\tau_1 \vdash_{ml} (\text{let}_T x \Leftarrow x' \text{ in } e): T\tau_2]_{\mathcal{T}}.$

RULE	SYNTAX	SEMANTICS
A	$\vdash_{ml} A \text{ type}$	$= \llbracket A \rrbracket$
T	$\vdash_{ml} \tau \text{ type}$	$= c$
	$\vdash_{ml} T\tau \text{ type}$	$= Tc$
var	$\vdash_{ml} \tau \text{ type}$	$= c$
	$x: \tau \vdash_{ml} x: \tau$	$= \text{id}_c$
f: $\tau_1 \rightarrow \tau_2$	$x: \tau \vdash_{ml} e_1: \tau_1$	$= g$
	$x: \tau \vdash_{ml} f(e_1): \tau_2$	$= g; \llbracket f \rrbracket$
[.]_T	$x: \tau \vdash_{ml} e: \tau'$	$= g$
	$x: \tau \vdash_{ml} [e]_T: T\tau'$	$= g; \eta_{\llbracket \tau' \rrbracket}$
let	$x: \tau \vdash_{ml} e_1: T\tau_1$	$= g_1$
	$x_1: \tau_1 \vdash_{ml} e_2: T\tau_2$	$= g_2$
	$x: \tau \vdash_{ml} (\text{let}_T x_1 \Leftarrow e_1 \text{ in } e_2): T\tau_2$	$= g_1; g_2^*$
eq	$x: \tau_1 \vdash_{ml} e_1: \tau_2$	$= g_1$
	$x: \tau_1 \vdash_{ml} e_2: \tau_2$	$= g_2$
	$x: \tau_1 \vdash_{ml} e_1 =_{\tau_2} e_2$	$\Leftrightarrow g_1 = g_2$

Table 3: Interpretation of the Simple Metalanguage

$$\begin{array}{l}
 [\cdot].\xi \frac{x: \tau \vdash_{ml} e_1 =_{\tau_1} e_2}{x: \tau \vdash_{ml} [e_1]_T =_{T\tau_1} [e_2]_T} \\
 \text{let.}\xi \frac{x: \tau \vdash_{ml} e_1 =_{T\tau_1} e_2 \quad x': \tau_1 \vdash_{ml} e'_1 =_{T\tau_2} e'_2}{x: \tau \vdash_{ml} (\text{let}_T x' \Leftarrow e_1 \text{ in } e'_1) =_{T\tau_2} (\text{let}_T x' \Leftarrow e_2 \text{ in } e'_2)} \\
 \text{ass} \frac{x: \tau \vdash_{ml} e_1: T\tau_1 \quad x_1: \tau_1 \vdash_{ml} e_2: T\tau_2 \quad x_2: \tau_2 \vdash_{ml} e_3: T\tau_3}{x: \tau \vdash_{ml} (\text{let}_T x_2 \Leftarrow (\text{let}_T x_1 \Leftarrow e_1 \text{ in } e_2) \text{ in } e_3) =_{T\tau_3} (\text{let}_T x_1 \Leftarrow e_1 \text{ in } (\text{let}_T x_2 \Leftarrow e_2 \text{ in } e_3))} \\
 \text{T.}\beta \frac{x: \tau \vdash_{ml} e_1: \tau_1 \quad x_1: \tau_1 \vdash_{ml} e_2: T\tau_2}{x: \tau \vdash_{ml} (\text{let}_T x_1 \Leftarrow [e_1]_T \text{ in } e_2) =_{T\tau_2} [e_1/x_1]e_2} \\
 \text{T.}\eta \frac{x: \tau \vdash_{ml} e_1: T\tau_1}{x: \tau \vdash_{ml} (\text{let}_T x_1 \Leftarrow e_1 \text{ in } [x_1]_T) =_{T\tau_1} e_1}
 \end{array}$$

Table 4: Inference Rules of the Simple Metalanguage

Proof We have to show that the three axioms for Kleisli triples are valid. The validity of each axiom amounts to the derivability of an equation. For instance, $\eta_\tau^* = \text{id}_{T\tau}$ is valid provided $x': T\tau \vdash_{ml} (\text{let}_T x \Leftarrow x' \text{ in } [x]_T) =_{T\tau} x'$ is derivable, indeed it follows from $(T.\eta)$. The reader can check that the equations corresponding to the axioms $\eta_\tau; f^* = f$ and $f^*, g^* = (f; g^*)^*$ follow from $(T.\beta)$ and (ass) respectively. ■

2.3 A Simple Programming Language

In this section we take a programming language perspective by introducing a simple programming language, whose terms are interpreted by morphisms of the Kleisli category for a monad. Unlike the metalanguage of Section 2.2, the programming language does not allow to consider more than one monad at once.

The interpretation in the Kleisli category can also be given indirectly via a translation in the simple metalanguage of Section 2.2 mapping *programs* of type τ into *terms* of type $T\tau$. If we try to establish a correspondence between *equational theories* of the simple programming language and categories with one monad (as done for the metalanguage), then we run into problems, since there is no way (in general) to recover \mathcal{C} from \mathcal{C}_T . What we do instead is to establish a correspondence between theories with *equivalence* and *existence* assertions and categories with one monad satisfying the *mono requirement*, i.e. η_A is mono for every object A (note that η_{TA} is always a mono, because $\eta_{TA}; \mu_A = \text{id}_{TA}$). The intended extension of the existence predicate on computations of type A is the set of computations of the form $[v]$ for some value v of type A , so it is natural to require η_A to be mono and interpret the existence predicate as the *subobject* corresponding to η_A .

The simple programming language is parametric in a signature, i.e. a set of base types and unary command symbols. To stress that the interpretation is in \mathcal{C}_T rather than \mathcal{C} , we use unary command symbols $p: \tau_1 \rightarrow \tau_2$ (instead of unary function symbols $f: \tau_1 \rightarrow \tau_2$), we call $x: \tau_1 \vdash_{pl} e: \tau_2$ a program (instead of a term) and write $_ \equiv_\tau _$ (instead of $_ =_{T\tau} _$) as equality of computations of type τ . Given a category \mathcal{C} with a Kleisli triple $(T, \eta, _^*)$ satisfying the mono requirement, an interpretation $\llbracket _ \rrbracket$ of the programming language is parametric in an interpretation of the symbols in the signature and is defined by induction on the derivation of well-formedness for types, terms and equations (see Table 5) following the same pattern given for many sorted monadic equational logic, but with \mathcal{C} replaced by \mathcal{C}_T , namely:

- the interpretation $\llbracket \tau \rrbracket$ of a (base) type τ is an object of \mathcal{C}_T , or equivalently an object of \mathcal{C}
- the interpretation $\llbracket p \rrbracket$ of an unary command $p: \tau_1 \rightarrow \tau_2$ is a morphism from $\llbracket \tau_1 \rrbracket$ to $\llbracket \tau_2 \rrbracket$ in \mathcal{C}_T , or equivalently a morphism from $\llbracket \tau_1 \rrbracket$ to $T\llbracket \tau_2 \rrbracket$ in \mathcal{C} ; similarly for the interpretation of a program $x: \tau_1 \vdash_{pl} e: \tau_2$
- the interpretation of an equivalence or existence assertion is a truth value.

Remark 2.6 The let-constructor play a fundamental role: operationally it corresponds to sequential evaluation of programs and categorically it corresponds to composition in the Kleisli category \mathcal{C}_T (while substitution corresponds to composition in \mathcal{C}). In the λ_v -calculus $(\text{let } x \Leftarrow e \text{ in } e')$ is treated as syntactic sugar for $(\lambda x. e')e$. We think that this is not the right way to proceed, because it explains the let-constructor (i.e. sequential evaluation of programs) in terms of constructors available only in functional languages. On the other hand, $(\text{let } x \Leftarrow e \text{ in } e')$ cannot be treated as syntactic sugar for $[e/x]e'$ (involving only the more primitive substitution) without collapsing computations to values.

The existence predicate $e \downarrow$ is inspired by the logic of partial terms/elements (see [Fou77, Sco79, Mog88]); however, there are important differences, e.g.

$$\text{strict } \frac{x: \tau \vdash_{pl} p(e) \downarrow_{\tau_2}}{x: \tau \vdash_{pl} e \downarrow_{\tau_1}} \quad p: \tau_1 \rightarrow \tau_2$$

is admissible for partial computations, but not in general. For certain notions of computation there may be other predicates on computations worth considering, or the existence predicate itself may have a more specialised meaning, for instance:

RULE	SYNTAX	SEMANTICS
A	$\vdash_{pl} A \text{ type}$	$= \llbracket A \rrbracket$
T	$\vdash_{pl} \tau \text{ type}$	$= c$
	$\vdash_{pl} T\tau \text{ type}$	$= Tc$
var	$\vdash_{pl} \tau \text{ type}$	$= c$
	$x: \tau \vdash_{pl} x: \tau$	$= \eta_c$
$p: \tau_1 \multimap \tau_2$	$x: \tau \vdash_{pl} e_1: \tau_1$	$= g$
	$x: \tau \vdash_{pl} p(e_1): \tau_2$	$= g; \llbracket p \rrbracket^*$
[]	$x: \tau \vdash_{pl} e: \tau'$	$= g$
	$x: \tau \vdash_{pl} [e]: T\tau'$	$= g; \eta_T \llbracket \tau' \rrbracket$
μ	$x: \tau \vdash_{pl} e: T\tau'$	$= g$
	$x: \tau \vdash_{pl} \mu(e): \tau'$	$= g; \mu \llbracket \tau' \rrbracket$
let	$x: \tau \vdash_{pl} e_1: \tau_1$	$= g_1$
	$x_1: \tau_1 \vdash_{pl} e_2: \tau_2$	$= g_2$
	$x: \tau \vdash_{pl} (\text{let } x_1 \Leftarrow e_1 \text{ in } e_2): \tau_2$	$= g_1; g_2^*$
eq	$x: \tau_1 \vdash_{pl} e_1: \tau_2$	$= g_1$
	$x: \tau_1 \vdash_{pl} e_2: \tau_2$	$= g_2$
	$x: \tau_1 \vdash_{pl} e_1 \equiv_{\tau_2} e_2$	$\iff g_1 = g_2$
ex	$x: \tau_1 \vdash_{pl} e: \tau_2$	$= g$
	$x: \tau_1 \vdash_{pl} e \downarrow_{\tau_2}$	$\iff \exists! h: \llbracket \tau_1 \rrbracket \rightarrow \llbracket \tau_2 \rrbracket \text{ s.t. } g = h; \eta_{\llbracket \tau_2 \rrbracket}$

Table 5: Interpretation of the Simple Programming Language

refl	$\frac{x: \tau \vdash_{pl} e: \tau_1}{x: \tau \vdash_{pl} e \equiv_{\tau_1} e}$
symm	$\frac{x: \tau \vdash_{pl} e_1 \equiv_{\tau_1} e_2}{x: \tau \vdash_{pl} e_2 \equiv_{\tau_1} e_1}$
trans	$\frac{x: \tau \vdash_{pl} e_1 \equiv_{\tau_1} e_2 \quad x: \tau \vdash_{pl} e_2 \equiv_{\tau_1} e_3}{x: \tau \vdash_{pl} e_2 \equiv_{\tau_1} e_3}$
congr	$\frac{x: \tau \vdash_{pl} e_1 \equiv_{\tau_1} e_2}{x: \tau \vdash_{pl} p(e_1) \equiv_{\tau_2} p(e_2)} \text{ p: } \tau_1 \rightarrow \tau_2$
E.x	$\frac{\vdash_{pl} \tau \text{ type}}{x: \tau \vdash_{pl} x \downarrow_{\tau}}$
E.congr	$\frac{x: \tau \vdash_{pl} e_1 \equiv_{\tau_1} e_2 \quad x: \tau \vdash_{pl} e_1 \downarrow_{\tau_1}}{x: \tau \vdash_{pl} e_2 \downarrow_{\tau_1}}$
subst	$\frac{x: \tau \vdash_{pl} e \downarrow_{\tau_1} \quad x: \tau_1 \vdash_{pl} \phi}{x: \tau \vdash_{pl} [e/x]\phi}$

Table 6: General Inference Rules

- a partial computation exists iff it terminates;
- a non-deterministic computation exists iff it gives exactly one result;
- a computation with side-effects exists iff it does not change the store.

Programs can be translated into terms of the metalanguage via a translation $_^\circ$ s.t. for every well-formed program $x: \tau_1 \vdash_{pl} e: \tau_2$ the term $x: \tau_1 \vdash_{ml} e^\circ: T\tau_2$ is well-formed and $\llbracket x: \tau_1 \vdash_{pl} e: \tau_2 \rrbracket = \llbracket x: \tau_1 \vdash_{ml} e^\circ: T\tau_2 \rrbracket$ (the proof of these properties is left to the reader).

Definition 2.7 Given a signature Σ for the programming language, let Σ° be the signature for the metalanguage with the same base types and a function $p: \tau_1 \rightarrow T\tau_2$ for each command $p: \tau_1 \rightarrow \tau_2$ in Σ . The translation $_^\circ$ from programs over Σ to terms over Σ° is defined by induction on raw programs:

- $x^\circ \stackrel{\Delta}{=} [x]_T$
- $(\text{let } x_1 \Leftarrow e_1 \text{ in } e_2)^\circ \stackrel{\Delta}{=} (\text{let}_T x_1 \Leftarrow e_1^\circ \text{ in } e_2^\circ)$
- $p(e_1)^\circ \stackrel{\Delta}{=} (\text{let}_T x \Leftarrow e_1^\circ \text{ in } p(x))$
- $[e]^\circ \stackrel{\Delta}{=} [e^\circ]_T$
- $\mu(e)^\circ \stackrel{\Delta}{=} (\text{let}_T x \Leftarrow e^\circ \text{ in } x)$

The inference rules for deriving equivalence and existence assertions of the simple programming language can be partitioned as follows:

- general rules (see Table 6) for terms denoting computations, but with variables ranging over values; these rules replace those of Table 2 for many sorted monadic equational logic
- rules capturing the properties of type- and term-constructors (see Table 7) after interpretation of the programming language; these rules replace the additional rules for the metalanguage given in Table 4.

$$\begin{array}{l}
[\cdot].\xi \frac{x:\tau \vdash_{pl} e_1 \equiv_{\tau_1} e_2}{x:\tau \vdash_{pl} [e_1] \equiv_{T\tau_1} [e_2]} \\
E.[\cdot] \frac{x:\tau \vdash_{pl} e_1 : \tau_1}{x:\tau \vdash_{pl} [e_1] \downarrow_{T\tau_1}} \\
\mu.\xi \frac{x:\tau \vdash_{pl} e_1 \equiv_{T\tau_1} e_2}{x:\tau \vdash_{pl} \mu(e_1) \equiv_{\tau_1} \mu(e_2)} \\
\mu.\beta \frac{x:\tau \vdash_{pl} e_1 : \tau_1}{x:\tau \vdash \mu([e_1]) \equiv_{\tau_1} e_1} \\
\mu.\eta \frac{x:\tau \vdash_{pl} e_1 \downarrow_{T\tau_1}}{x:\tau \vdash [\mu(e_1)] \equiv_{T\tau_1} e_1} \\
\text{let.}\xi \frac{x:\tau \vdash_{pl} e_1 \equiv_{\tau_1} e_2 \quad x':\tau_1 \vdash_{pl} e'_1 \equiv_{\tau_2} e'_2}{x:\tau \vdash_{pl} (\text{let } x' \Leftarrow e_1 \text{ in } e'_1) \equiv_{\tau_2} (\text{let } x' \Leftarrow e_2 \text{ in } e'_2)} \\
\text{unit} \frac{x:\tau \vdash_{pl} e_1 : \tau_1}{x:\tau \vdash_{pl} (\text{let } x_1 \Leftarrow e_1 \text{ in } x_1) \equiv_{\tau_1} e_1} \\
\text{ass} \frac{x:\tau \vdash_{pl} e_1 : \tau_1 \quad x_1:\tau_1 \vdash_{pl} e_2 : \tau_2 \quad x_2:\tau_2 \vdash_{pl} e_3 : \tau_3}{x:\tau \vdash_{pl} (\text{let } x_2 \Leftarrow (\text{let } x_1 \Leftarrow e_1 \text{ in } e_2) \text{ in } e_3) \equiv_{\tau_3} (\text{let } x_1 \Leftarrow e_1 \text{ in } (\text{let } x_2 \Leftarrow e_2 \text{ in } e_3))} \\
\text{let.}\beta \frac{x:\tau \vdash_{pl} e_1 \downarrow_{\tau_1} \quad x_1:\tau_1 \vdash_{pl} e_2 : \tau_2}{x:\tau \vdash_{pl} (\text{let } x_1 \Leftarrow e_1 \text{ in } e_2) \equiv_{\tau_2} [e_1/x_1]e_2} \\
\text{let.p} \frac{x:\tau \vdash_{pl} e_1 : \tau_1}{x:\tau \vdash_{pl} p(e_1) \equiv_{\tau_1} (\text{let } x_1 \Leftarrow e_1 \text{ in } p(x_1))} \quad p:\tau_1 \multimap \tau_2
\end{array}$$

Table 7: Inference Rules of the Simple Programming Language

Soundness and completeness of the formal consequence relation w.r.t. interpretation of the simple programming language in categories with a monad satisfying the mono requirement is established in the usual way (see Section 2.1). The only step which differs is how to view a theory \mathcal{T} of the simple programming language (i.e. a set of equivalence and existence assertions closed w.r.t. the inference rules) as a category $\mathcal{F}(\mathcal{T})$ with the required structure.

Definition 2.8 *Given a theory \mathcal{T} of the simple programming language, the category $\mathcal{F}(\mathcal{T})$ is defined as follows:*

- objects are types τ ,
- morphisms from τ_1 to τ_2 are equivalence classes $[x: \tau_1 \vdash_{pl} e: \tau_2]_{\mathcal{T}}$ of existing programs $x: \tau_1 \vdash_{pl} e \downarrow_{\tau_2} \in \mathcal{T}$ w.r.t. the equivalence relation induced by the theory \mathcal{T} , i.e.

$$(x: \tau_1 \vdash_{pl} e_1: \tau_2) \equiv (x: \tau_1 \vdash_{pl} e_2: \tau_2) \iff (x: \tau_1 \vdash_{pl} e_1 \equiv_{\tau_2} e_2) \in \mathcal{T}$$

- composition is substitution, i.e.

$$[x: \tau_1 \vdash_{pl} e_1: \tau_2]_{\mathcal{T}}; [x: \tau_2 \vdash_{pl} e_2: \tau_3]_{\mathcal{T}} = [x: \tau_1 \vdash_{pl} [e_1/x]e_2: \tau_3]_{\mathcal{T}}$$

- identity over τ is $[x: \tau \vdash_{pl} x: \tau]_{\mathcal{T}}$.

In order for composition in $\mathcal{F}(\mathcal{T})$ to be well-defined, it is essential to consider only equivalence classes of existing programs, since the simple programming language satisfies only a restricted form of substitutivity.

Proposition 2.9 *Every theory \mathcal{T} of the simple programming language, viewed as a category $\mathcal{F}(\mathcal{T})$, is equipped with a Kleisli triple $(T, \eta, -^*)$ satisfying the mono requirement:*

- $T(\tau) = T\tau$,
- $\eta_{\tau} = [x: \tau \vdash_{pl} [x]: T\tau]_{\mathcal{T}}$,
- $([x: \tau_1 \vdash_{pl} e: T\tau_2]_{\mathcal{T}})^* = [x': T\tau_1 \vdash_{pl} [(let x \leftarrow \mu(x') \text{ in } \mu(e))]: T\tau_2]_{\mathcal{T}}$.

Proof We have to show that the three axioms for Kleisli triples are valid. The validity of each axiom amounts to the derivability of an existence and equivalence assertion. For instance, $\eta_{\tau}^* = \text{id}_{T\tau}$ is valid provided $x': T\tau \vdash_{pl} x' \downarrow_{T\tau}$ and $x': T\tau \vdash_{pl} [(let x \leftarrow \mu(x') \text{ in } \mu([x]))] \equiv_{T\tau} x'$ are derivable. The existence assertion follows immediately from (E.x), while the equivalence is derived as follows:

- $x': T\tau \vdash_{pl} [(let x \leftarrow \mu(x') \text{ in } \mu([x]))] \equiv_{T\tau} [(let x \leftarrow \mu(x') \text{ in } x)]$ by $(\mu.\beta)$, (refl) and (let. ξ)
- $x': T\tau \vdash_{pl} [(let x \leftarrow \mu(x') \text{ in } x)] \equiv_{T\tau} [\mu(x')]$ by (unit) and (let. ξ)
- $x': T\tau \vdash_{pl} [\mu(x')] \equiv_{T\tau} x'$ by (E.x) and $(\mu.\eta)$
- $x': T\tau \vdash_{pl} [(let x \leftarrow \mu(x') \text{ in } \mu([x]))] \equiv_{T\tau} x'$ by (trans).

We leave to the reader the derivation of the existence and equivalence assertions corresponding to the other axioms for Kleisli triples, and prove instead the mono requirement i.e. that $f_1; \eta_{\tau} = f_2; \eta_{\tau}$ implies $f_1 = f_2$. Let f_i be $[x: \tau' \vdash_{pl} e_i: \tau]_{\mathcal{T}}$, we have to derive $x: \tau' \vdash_{pl} e_1 \equiv_{\tau} e_2$ from $x: \tau' \vdash_{pl} [e_1] \equiv_{T\tau} [e_2]$ (and $x: \tau' \vdash_{pl} e_i \downarrow_{\tau}$):

- $x: \tau' \vdash_{pl} \mu([e_1]) \equiv_{\tau} \mu([e_2])$ by the first assumption and $(\mu.\xi)$
- $x: \tau' \vdash_{pl} \mu([e_i]) \equiv_{\tau} e_i$ by $(\mu.\beta)$
- $x: \tau' \vdash_{pl} e_1 \equiv_{\tau} e_2$ by (trans).

Remark 2.10 One can show that the canonical interpretation of a program $x:\tau_1 \vdash_{pl} e:\tau_2$ in the category $\mathcal{F}(\mathcal{T})$ is the morphism $[x:\tau_1 \vdash_{pl} [e]:T\tau_2]_{\mathcal{T}}$. This interpretation establishes a one-one correspondence between morphisms from τ_1 to $T\tau_2$ in the category $\mathcal{F}(\mathcal{T})$, i.e. morphisms from τ_1 to τ_2 in the Kleisli category, and equivalence classes of programs $x:\tau_1 \vdash_{pl} e:\tau_2$ (not necessarily existing). The inverse correspondence maps a morphism $[x:\tau_1 \vdash_{pl} e':T\tau_2]_{\mathcal{T}}$ to the equivalence class of $x:\tau_1 \vdash_{pl} \mu(e'):\tau_2$. Indeed, $x:\tau_1 \vdash_{pl} e \equiv_{\tau_2} \mu([e])$ and $x:\tau_1 \vdash_{pl} e' \equiv_{\tau_2} [\mu(e')]$ are derivable provided $x:\tau_1 \vdash_{pl} e' \downarrow_{T\tau_2}$.

3 Extending the simple metalanguage

So far we have considered only languages and formal systems for *monadic terms* $x:\tau_1 \vdash e:\tau_2$, having exactly one free variable (occurring once). In this section we want to extend these languages (and formal systems) by allowing *algebraic terms* $x_1:\tau_1, \dots, x_n:\tau_n \vdash e:\tau$, having a finite number of free variables (occurring finitely many times) and investigate how this affects the interpretation and the structure on theories viewed as categories. For convenience in relating theories and categories with additional structure, we also allow types to be closed w.r.t. finite products⁴, in particular a typing context $x_1:\tau_1, \dots, x_n:\tau_n$ can be identified with a type. In general, the interpretation of an algebraic term $x_1:\tau_1, \dots, x_n:\tau_n \vdash e:\tau$ in a category (with finite products) is a morphism from $([\tau_1] \times \dots \times [\tau_n])$ to $[\tau]$.

The extension of monadic equational logic to algebraic terms is equational logic, whose theories correspond to categories with finite products. We will introduce the *metalanguage*, i.e. the extension of the simple metalanguage described in Section 2.2 to algebraic terms, and show that its theories correspond to categories with finite products and a *strong monad*, i.e. a monad and a natural transformation $t_{A,B}: A \times TB \rightarrow T(A \times B)$. Intuitively $t_{A,B}$ transforms a pair value-computation into a computation of a pair of values, as follows

$$a:A, c:TB \quad \stackrel{t_{A,B}}{\longmapsto} \quad (\text{let } y \Leftarrow c \text{ in } [\langle a, y \rangle]):T(A \times B)$$

Remark 3.1 To understand why a category with finite products and a monad is not enough to interpret the metalanguage (and where the natural transformation t is needed), one has to look at the interpretation of a let-expression

$$\text{let } \frac{\Gamma \vdash_{ml} e_1:T\tau_1 \quad \Gamma, x:\tau_1 \vdash_{ml} e_2:T\tau_2}{\Gamma \vdash_{ml} (\text{let}_T x \Leftarrow e_1 \text{ in } e_2):T\tau_2}$$

where Γ is a typing context. Let $g_1:c \rightarrow Tc_1$ and $g_2:c \times c_1 \rightarrow Tc_2$ be the interpretations of $\Gamma \vdash_{ml} e_1:T\tau_1$ and $\Gamma, x:\tau_1 \vdash_{ml} e_2:T\tau_2$ respectively, where c is the interpretation of the typing context Γ and c_i is the interpretation of the type τ_i , then the interpretation of $\Gamma \vdash_{ml} (\text{let}_T x \Leftarrow e_1 \text{ in } e_2):T\tau_2$ ought to be a morphism $g:c \rightarrow Tc_2$. If (T, η, μ) is the *identity monad*, i.e. T is the identity functor over \mathcal{C} and η and μ are the identity natural transformation over T , then computations get identified with values. In this case $(\text{let}_T x \Leftarrow e_1 \text{ in } e_2)$ can be replaced by $[e_1/x]e_2$, so g is simply $\langle \text{id}_c, g_1 \rangle; g_2: c \rightarrow c_2$. In the general case Table 3 suggests that $_ ; _$ above is indeed composition in the Kleisli category, therefore $\langle \text{id}_c, g_1 \rangle; g_2$ should be replaced by $\langle \text{id}_c, g_1 \rangle; g_2^*$. But in $\langle \text{id}_c, g_1 \rangle; g_2^*$ there is a type mismatch, since the codomain of $\langle \text{id}_c, g_1 \rangle$ is $c \times Tc_1$, while the domain of Tg_1 is $T(c \times c_1)$. The natural transformation $t_{A,B}: A \times TB \rightarrow T(A \times B)$ mediates between these two objects, so that g can be defined as $\langle \text{id}_c, g_1 \rangle; t_{c,c_1}; g_2^*$.

⁴If the metalanguage does not have finite products, we conjecture that its theories would no longer correspond to categories with finite products and a strong monad (even by taking as objects contexts and/or the Karoubi envelope, used in [Sco80] to associate a cartesian closed category to an untyped λ -theory), but instead to *multicategories* with a Kleisli triple. We felt the greater generality (of not having products in the metalanguage) was not worth the mathematical complications.

Definition 3.2 A **strong monad** over a category \mathcal{C} with (explicitly given) finite products is a monad (T, η, μ) together with a natural transformation $t_{A,B}$ from $A \times TB$ to $T(A \times B)$ s.t.

$$\begin{array}{ccc}
& TA & \\
r_{TA} \swarrow & \downarrow Tr_A & \searrow t_{1,A} \\
1 \times TA & \xrightarrow{\quad} & T(1 \times A)
\end{array}$$

$$\begin{array}{ccccc}
(A \times B) \times TC & \xrightarrow{\quad t_{A \times B, C} \quad} & T((A \times B) \times C) & & \\
\alpha_{A,B,TC} \downarrow & & & & T\alpha_{A,B,C} \searrow \\
A \times (B \times TC) & \xrightarrow{\quad id_A \times t_{B,C} \quad} & A \times T(B \times C) & \xrightarrow{\quad t_{A,B \times C} \quad} & T(A \times (B \times C))
\end{array}$$

$$\begin{array}{ccccc}
A \times B & & & & \\
id_A \times \eta_B \downarrow & \searrow \eta_{A \times B} & & & \\
A \times TB & \xrightarrow{\quad t_{A,B} \quad} & T(A \times B) & & \\
id_A \times \mu_B \downarrow & & & & \mu_{A \times B} \searrow \\
A \times T^2B & \xrightarrow{\quad t_{A,TB} \quad} & T(A \times TB) & \xrightarrow{\quad Tt_{A,B} \quad} & T^2(A \times B)
\end{array}$$

where r and α are the natural isomorphisms

$$r_A: (1 \times A) \rightarrow A \quad , \quad \alpha_{A,B,C}: (A \times B) \times C \rightarrow A \times (B \times C)$$

Remark 3.3 The diagrams above are taken from [Koc72], where a characterisation of strong monads is given in terms of \mathcal{C} -enriched categories (see [Kel82]). Kock fixes a commutative monoidal closed category \mathcal{C} (in particular a cartesian closed category), and in this setup he establishes a one-one correspondence between *strengths* $st_{A,B}: B^A \rightarrow (TB)^{TA}$ and *tensorial strengths* $t_{A,B}: A \otimes TB \rightarrow T(A \otimes B)$ for a endofunctor T over \mathcal{C} (see Theorem 1.3 in [Koc72]). Intuitively a strength $st_{A,B}$ internalises the action of T on morphisms from A to B , and more precisely it makes (T, st) a \mathcal{C} -enriched endofunctor on \mathcal{C} enriched over itself (i.e. the hom-object $\mathcal{C}(A, B)$ is B^A). In this setting the diagrams of Definition 3.2 have the following meaning:

- the first two diagrams are (1.7) and (1.8) in [Koc72], saying that t is a tensorial strength of T . So T can be made into a \mathcal{C} -enriched endofunctor.
- the last two diagrams say that $\eta: Id_{\mathcal{C}} \rightarrow T$ and $\mu: T^2 \rightarrow T$ are \mathcal{C} -enriched natural transformations, where $Id_{\mathcal{C}}$, T and T^2 are enriched in the obvious way (see Remark 1.4 in [Koc72]).

There is another purely categorical characterisation of strong monads, suggested to us by G. Plotkin, in terms of \mathcal{C} -indexed categories (see [JP78]). Both characterisations are instances of a general methodological principle for studying programming languages (or logics) categorically (see [Mog89b]):

when studying a complex language the 2-category **Cat** of small categories, functors and natural transformations may not be adequate; however, one may replace **Cat** with a different 2-category, whose objects captures better some *fundamental structure* of the language, while *less fundamental structure* can be modelled by *2-categorical concepts*.

Monads are a 2-categorical concept, so we expect notions of computations for a complex language to be modelled by monads in a suitable 2-category.

The first characterisation takes a commutative monoidal closed structure on \mathcal{C} (used in [Laf88, See87] to model a fragment of *linear logic*), so that \mathcal{C} can be enriched over itself. Then a strong monad over a cartesian closed category \mathcal{C} is just a monad over \mathcal{C} in the 2-category of \mathcal{C} -enriched categories.

The second characterisation takes a class \mathcal{D} of display maps over \mathcal{C} (used in [HP87] to model *dependent types*), and defines a \mathcal{C} -indexed category $\mathcal{C}/_{\mathcal{D}\text{-}}$. Then a strong monad over a category \mathcal{C} with finite products amounts to a monad over $\mathcal{C}/_{\mathcal{D}\text{-}}$ in the 2-category of \mathcal{C} -indexed categories, where \mathcal{D} is the class of first projections (corresponding to constant type dependency).

In general the natural transformation t has to be given explicitly as part of the additional structure. However, t is uniquely determined (but it may not exist) by T and the cartesian structure on \mathcal{C} , when \mathcal{C} has enough points.

Proposition 3.4 (Uniqueness) *If (T, η, μ) is a monad over a category \mathcal{C} with finite products and enough points (i.e. $\forall h: 1 \rightarrow A. h; f = h; g \text{ implies } f = g$ for any $f, g: A \rightarrow B$), then (T, η, μ, t) is a strong monad over \mathcal{C} if and only if $t_{A,B}$ is the unique family of morphisms s.t. for all points $a: 1 \rightarrow A$ and $b: 1 \rightarrow TB$*

$$\langle a, b \rangle; t_{A,B} = b; T(\langle !_B; a, id_B \rangle)$$

where $!_B: B \rightarrow 1$ is the unique morphism from B to the terminal object.

Proof Note that there is at most one $t_{A,B}$ s.t. $\langle a, b \rangle; t_{A,B} = b; T(\langle !_B; a, id_B \rangle)$ for all points $a: 1 \rightarrow A$ and $b: 1 \rightarrow TB$, because \mathcal{C} has enough points.

First we show that if (T, η, μ, t) is a strong monad, then $t_{A,B}$ satisfies the equation above. By naturality of t and by the first diagram in Definition 3.2 the following diagram commutes

$$\begin{array}{ccccc}
& \langle a, b \rangle & & & \\
1 & \xrightarrow{\quad} & A \times TB & \xrightarrow{\quad t_{A,B} \quad} & T(A \times B) \\
& \searrow \langle id_1, b \rangle & \uparrow a \times id_{TB} & & \uparrow T(a \times id_B) \\
& & 1 \times TB & \xrightarrow{\quad t_{1,B} \quad} & T(1 \times B) \\
& & & \searrow r_{TB} & \downarrow Tr_B \\
& & & & TB
\end{array}$$

Since r_B is an isomorphism (with inverse $\langle !_B, id_B \rangle$), then the two composite morphisms $\langle a, b \rangle; t_{A,B}$ and $\langle id_1, b \rangle; r_{TB}; T(r_B^{-1}); T(a \times id_B)$ from 1 to $T(A \times B)$ must coincide. But the second composition can be rewritten as $b; T(\langle !_B; a, id_B \rangle)$.

Second we have to show that if t is the unique family of morphisms satisfying the equation above, then (T, η, μ, t) is a strong monad. This amounts to prove that t is a natural transformation and that the three diagrams in Definition 3.2 commute. The proof is a tedious diagram chasing, which relies on \mathcal{C} having enough points. For instance, to prove that $t_{1,A}; Tr_A = r_{TA}$ it is enough to show that $\langle id_1, a \rangle; t_{1,A}; Tr_A = \langle id_1, a \rangle; r_{TA}$ for all points $a: 1 \rightarrow A$. ■

Example 3.5 We go through the monads given in Example 1.4 and show that they have a tensorial strength.

- **partiality** $TA = A_\perp (= A + \{\perp\})$
 $t_{A,B}(a, \perp) = \perp$ and $t_{A,B}(a, b) = \langle a, b \rangle$ (when $b \in B$)
- **nondeterminism** $TA = \mathcal{P}_{fin}(A)$
 $t_{A,B}(a, c) = \{\langle a, b \rangle | b \in c\}$

- **side-effects** $TA = (A \times S)^S$
 $t_{A,B}(a, c) = (\lambda s: S. (\text{let } \langle b, s' \rangle = c(s) \text{ in } \langle \langle a, b \rangle, s' \rangle))$
- **exceptions** $TA = (A + E)$
 $t_{A,B}(a, \text{inr}(e)) = \text{inr}(e)$ (when $e \in E$) and
 $t_{A,B}(a, \text{inl}(b)) = \text{inl}(\langle a, b \rangle)$ (when $b \in B$)
- **continuations** $TA = R^{(R^A)}$
 $t_{A,B}(a, c) = (\lambda k: R^{A \times B}. c(\lambda b: B. k(\langle a, b \rangle)))$
- **interactive input** $TA = (\mu\gamma.A + \gamma^U)$
 $t_{A,B}(a, c)$ is the tree obtained by replacing leaves of c labelled by b with the leaf labelled by $\langle a, b \rangle$
- **interactive output** $TA = (\mu\gamma.A + (U \times \gamma))$
 $t_{A,B}(a, \langle s, b \rangle) = \langle s, \langle a, b \rangle \rangle.$

Remark 3.6 The tensorial strength t induces a natural transformation $\psi_{A,B}$ from $TA \times TB$ to $T(A \times B)$, namely

$$\psi_{A,B} = c_{TA,TB}; t_{TB,A}; (c_{TB,A}; t_{A,B})^*$$

where c is the natural isomorphism $c_{A,B}: A \times B \rightarrow B \times A$.

The morphism $\psi_{A,B}$ has the correct domain and codomain to interpret the pairing of a computation of type A with one of type B , obtained by first evaluating the first argument and then the second, namely

$$c_1: TA, c_2: TB \xrightarrow{\psi_{A,B}} (\text{let } x \Leftarrow c_1 \text{ in } (\text{let } y \Leftarrow c_2 \text{ in } [\langle x, y \rangle])): T(A \times B)$$

There is also a dual notion of pairing, $\tilde{\psi}_{A,B} = c_{TA,TB}; \psi_{B,A}; Tc_{B,A}$ (see [Koc72]), which amounts to first evaluating the second argument and then the first.

3.1 Interpretation and formal system

We are now in a position to give the metalanguage for algebraic terms, its interpretation and inference rules.

Definition 3.7 (metalanguage) An interpretation $\llbracket \cdot \rrbracket$ of the metalanguage in a category \mathcal{C} with terminal object $!_A: A \rightarrow 1$, binary products $\pi_i^{A_1, A_2}: A_1 \times A_2 \rightarrow A_i$ and a strong monad (T, η, μ, t) is parametric in an interpretation of the symbols in the signature and is defined by induction on the derivation of well-formedness for types (see Table 8), terms and equations (see Table 9).

Finite products $\pi_i^{A_1, \dots, A_n}: A_1 \times \dots \times A_n \rightarrow A_i$ used to interpret contexts and variables are defined by induction on n :

$$\begin{aligned} 0 \ A_1 \times \dots \times A_0 &\stackrel{\Delta}{=} 1 \\ n+1 \ A_1 \times \dots \times A_{n+1} &\stackrel{\Delta}{=} (A_1 \times \dots \times A_n) \times A_{n+1} \\ - \ \pi_{n+1}^{A_1, \dots, A_{n+1}} &= \pi_2^{(A_1 \times \dots \times A_n), A_{n+1}} \\ - \ \pi_i^{A_1, \dots, A_{n+1}} &= \pi_1^{(A_1 \times \dots \times A_n), A_{n+1}}; \pi_i^{A_1, \dots, A_n} \end{aligned}$$

The inference rules for the metalanguage (see Table 10) are divided into three groups:

- general rules for many sorted equational logic
- rules for finite products
- rules for T

RULE	SYNTAX	SEMANTICS
A	$\vdash_{ml} A \text{ type}$	$= \llbracket A \rrbracket$
T	$\vdash_{ml} \tau \text{ type}$	$= c$
	$\vdash_{ml} T\tau \text{ type}$	$= Tc$
1	$\vdash_{ml} 1 \text{ type}$	$= 1$
\times	$\vdash_{ml} \tau_1 \text{ type}$ $\vdash_{ml} \tau_2 \text{ type}$	$= c_1$ $= c_2$
	$\vdash_{ml} \tau_1 \times \tau_2 \text{ type}$	$= c_1 \times c_2$
\emptyset	$\vdash_{ml} \tau_i \text{ type } (1 \leq i \leq n)$	$= c_i$
	$x_1 : \tau_1, \dots, x_n : \tau_n \vdash$	$= c_1 \times \dots \times c_n$

Table 8: Interpretation of types in the Metalanguage

Proposition 3.8 *Every theory \mathcal{T} of the metalanguage, viewed as a category $\mathcal{F}(\mathcal{T})$, is equipped with finite products and a strong monad whose tensorial strength is*

$$t_{\tau_1, \tau_2} = [x : \tau_1 \times T\tau_2 \vdash_{ml} (\text{let}_T x_2 \leftarrow \pi_2 x \text{ in } [\langle \pi_1 x, x_2 \rangle]_T) : T(\tau_1 \times \tau_2)]_{\mathcal{T}}$$

Proof Similar to that of Proposition 2.5 ■

Once we have a metalanguage for algebraic terms it is straightforward to add data-types characterised by *universal properties* and extend the categorical semantics accordingly⁵. For instance, if we want to have function spaces, then we simply require the category \mathcal{C} (where the metalanguage is interpreted) to have exponentials B^A and add the inference rules for the simply typed λ -calculus (see Table 11) to those for the metalanguage. From a programming language perspective the situation is more delicate. For instance, the semantics of functional types should reflect the choice of *calling mechanism*⁶:

- in call-by-value a procedure of type $A \rightarrow B$ expects a value of type A and computes a result of type B , so the interpretation of $A \rightarrow B$ is $(TB)^A$;
- in call-by-name a procedure of type $A \rightarrow B$ expects a computation of type A , which is evaluated only when needed, and computes a result of type B , so the interpretation of $A \rightarrow B$ is $(TB)^{TA}$.

In both cases the only exponentials needed to interpret the functional types of a programming language are of the form $(TB)^A$. By analogy with partial cartesian closed categories (pccc), where only *p-exponentials* are required to exist (see [Mog86, Ros86]), we adopt the following definition of λ_c -model:

⁵The next difficult step in extending the metalanguage is the combination of dependent types and computations, which is currently under investigation.

⁶call-by-need does not have a simple categorical semantics, since the environment in which an expression is evaluated may itself undergo evaluation.

RULE	SYNTAX	SEMANTICS
var _i	$\vdash_{ml} \tau_i \text{ type } (1 \leq i \leq n)$ $x_1 : \tau_1, \dots, x_n : \tau_n \vdash x_i : \tau_i$	$= c_i$ $= \pi_i^{c_1, \dots, c_n}$
*	$\Gamma \vdash * : 1$	$= !_{[\Gamma]}$
$\langle \rangle$	$\Gamma \vdash e_1 : \tau_1$ $\Gamma \vdash e_2 : \tau_2$ $\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2$	$= g_1$ $= g_2$ $= \langle g_1, g_2 \rangle$
π_i	$\Gamma \vdash e : \tau_1 \times \tau_2$ $\Gamma \vdash \pi_i(e) : \tau_1$	$= g$ $= g; \pi_i^{[\tau_1], [\tau_2]}$
$f : \tau_1 \rightarrow \tau_2$	$\Gamma \vdash_{ml} e_1 : \tau_1$ $\Gamma \vdash_{ml} f(e_1) : \tau_2$	$= g$ $= g; [f]$
$[.]_T$	$\Gamma \vdash_{ml} e : \tau$ $\Gamma \vdash_{ml} [e]_T : T\tau$	$= g$ $= g; \eta_{[\tau]}$
let	$\Gamma \vdash_{ml} e_1 : T\tau_1$ $\Gamma, x : \tau_1 \vdash_{ml} e_2 : T\tau_2$ $\Gamma \vdash_{ml} (\text{let}_T x \Leftarrow e_1 \text{ in } e_2) : T\tau_2$	$= g_1$ $= g_2$ $= \langle \text{id}_{[\Gamma]}, g_1 \rangle; t_{[\Gamma], [\tau_1]}; g_2^*$
eq	$\Gamma \vdash_{ml} e_1 : \tau$ $\Gamma \vdash_{ml} e_2 : \tau$ $\Gamma \vdash_{ml} e_1 =_\tau e_2$	$= g_1$ $= g_2$ $\iff g_1 = g_2$

Table 9: Interpretation of terms in the Metalanguage

$$\begin{array}{ll}
\text{refl} & \frac{\Gamma \vdash e : \tau}{\Gamma \vdash e =_{\tau} e} \\
\text{symm} & \frac{\Gamma \vdash e_1 =_{\tau} e_2}{\Gamma \vdash e_2 =_{\tau} e_1} \\
\text{trans} & \frac{\Gamma \vdash e_1 =_{\tau} e_2 \quad \Gamma \vdash e_2 =_{\tau} e_3}{\Gamma \vdash e_1 =_{\tau} e_3} \\
\text{congr} & \frac{\Gamma \vdash e_1 =_{\tau_1} e_2 \quad f : \tau_1 \rightarrow \tau_2}{\Gamma \vdash f(e_1) =_{\tau_2} f(e_2)} \\
\text{subst} & \frac{\Gamma \vdash e : \tau \quad \Gamma, x : \tau \vdash \phi}{\Gamma \vdash [e/x]\phi}
\end{array}$$

Inference Rules of Many Sorted Equational Logic

$$\begin{array}{l}
1.\eta \quad \Gamma \vdash * =_1 x \\
\langle . \rangle .\xi \quad \frac{\Gamma \vdash e_1 =_{\tau_1} e'_1 \quad \Gamma \vdash e_2 =_{\tau_2} e'_2}{\Gamma \vdash \langle e_1, e_2 \rangle =_{\tau_1 \times \tau_2} \langle e'_1, e'_2 \rangle} \\
\times.\beta \quad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \pi_i(\langle e_1, e_2 \rangle) =_{\tau_i} e_i} \\
\times.\eta \quad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \langle \pi_1(e), \pi_2(e) \rangle =_{\tau_1 \times \tau_2} e}
\end{array}$$

rules for product types

$$\begin{array}{l}
[\cdot].\xi \quad \frac{\Gamma \vdash_{ml} e_1 =_{\tau} e_2}{\Gamma \vdash_{ml} [e_1]_T =_{T\tau} [e_2]_T} \\
\text{let.}\xi \quad \frac{\Gamma \vdash_{ml} e_1 =_{T\tau_1} e_2 \quad \Gamma, x : \tau_1 \vdash_{ml} e'_1 =_{T\tau_2} e'_2}{\Gamma \vdash_{ml} (\text{let}_T x \leftarrow e_1 \text{ in } e'_1) =_{T\tau_2} (\text{let}_T x \leftarrow e_2 \text{ in } e'_2)} \\
\text{ass} \quad \frac{\Gamma \vdash_{ml} e_1 : T\tau_1 \quad \Gamma, x_1 : \tau_1 \vdash_{ml} e_2 : T\tau_2 \quad \Gamma, x_2 : \tau_2 \vdash_{ml} e_3 : T\tau_3}{\Gamma \vdash_{ml} (\text{let}_T x_2 \leftarrow (\text{let}_T x_1 \leftarrow e_1 \text{ in } e_2) \text{ in } e_3) =_{T\tau_3} (\text{let}_T x_1 \leftarrow e_1 \text{ in } (\text{let}_T x_2 \leftarrow e_2 \text{ in } e_3))} \\
T.\beta \quad \frac{\Gamma \vdash_{ml} e_1 : \tau_1 \quad \Gamma, x_1 : \tau_1 \vdash_{ml} e_2 : T\tau_2}{\Gamma \vdash_{ml} (\text{let}_T x_1 \leftarrow [e_1]_T \text{ in } e_2) =_{T\tau_2} [e_1/x_1]e_2} \\
T.\eta \quad \frac{\Gamma \vdash_{ml} e_1 : T\tau_1}{\Gamma \vdash_{ml} (\text{let}_T x_1 \leftarrow e_1 \text{ in } [x_1]_T) =_{T\tau_1} e_1}
\end{array}$$

Table 10: Inference Rules of the Metalanguage

$$\begin{array}{l}
\text{app.}\xi \quad \frac{\Gamma \vdash e_1 =_{\tau_1} e'_1 \quad \Gamma \vdash e =_{\tau_1 \rightarrow \tau_2} e'}{\Gamma \vdash ee_1 =_{\tau_2} e'e'_1} \\
\lambda.\xi \quad \frac{\Gamma, x : \tau_1 \vdash e_1 =_{\tau_2} e_2}{\Gamma \vdash (\lambda x : \tau_1. e_1) =_{\tau_1 \rightarrow \tau_2} (\lambda x : \tau_1. e_2)} \\
\rightarrow .\beta \quad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash (\lambda x : \tau_1. e_2)e_1 =_{\tau_2} [e_1/x]e_2} \\
\rightarrow .\eta \quad \frac{\Gamma \vdash e : \tau_1 \rightarrow \tau_2}{\Gamma \vdash (\lambda x : \tau_1. ex) =_{\tau_1 \rightarrow \tau_2} e} \quad x \notin \text{DV}(\Gamma)
\end{array}$$

Table 11: rules for function spaces

Definition 3.9 A λ_c -model is a category \mathcal{C} with finite products, a strong monad (T, η, μ, t) satisfying the mono requirement (i.e. η_A mono for every $A \in \mathcal{C}$) and **T-exponential** $(TB)^A$ for every $A, B \in \mathcal{C}$.

Remark 3.10 The definition of λ_c -model generalises that of pccc, in the sense that every pccc can be viewed as a λ_c -model. By analogy with p-exponentials, a T -exponential can be defined by giving an isomorphism $\mathcal{C}_T(C \times A, B) \cong \mathcal{C}(C, (TB)^A)$ natural in $C \in \mathcal{C}$. We refer to [Mog89c] for the interpretation of a call-by-value programming language in a λ_c -model and the corresponding formal system, the λ_c -calculus.

4 Strong monads over a topos

In this section we show that, as far as monads or strong monads are concerned, we can assume w.l.o.g. that they are over a topos (see Theorem 4.9). The proof of Theorem 4.9 involves non-elementary notions from Category Theory, and we postpone it after discussing some applications, with particular emphasis on further extensions of the metalanguage and on conservative extension results.

Let us take as formal system for toposes the type theory described in [LS86], this is a many sorted intuitionistic higher order logic with equality and with a set of types satisfying the following closure properties⁷:

- the terminal object 1 , the natural number object N and the subobject classifier Ω are types
- if A is a type, then the power object PA is a type
- if A and B are types, then the binary product $A \times B$ and the function space $A \rightarrow B$ are types
- if A is a type and $\phi: A \rightarrow \Omega$ is a predicate, then $\{x \in A | \phi(x)\}$ is a type.

Notation 4.1 We introduce some notational conventions for formal systems:

- ML_T is the metalanguage for algebraic terms, whose set of types is closed under terminal object, binary products and TA ;
- λML_T is the extension of ML_T with function spaces $A \rightarrow B$ (interpreted as exponentials);
- HML_T is the type theory described above (see [LS86]) extended with objects of computations TA ;
- PL is the programming language for algebraic terms (see [Mog89c]);
- $\lambda_c\text{PL}$ is the extension of PL with function spaces $A \rightarrow B$ (interpreted as T -exponentials), called λ_c -calculus in [Mog89c].

Definition 4.2 We say that a formal system (L_2, \vdash_2) , where $\vdash_2 \subseteq \mathcal{P}(L_2) \times L_2$ is a formal consequence relation⁸ over L_2 , is a **conservative extension** of (L_1, \vdash_1) provided $L_1 \subseteq L_2$ and \vdash_1 is the restriction of \vdash_2 to $\mathcal{P}(L_1) \times L_1$.

Theorem 4.3 HML_T is a conservative extension of ML_T and λML_T . In particular λML_T is a conservative extension of ML_T .

⁷Lambek and Scott do not require closure under function spaces and subsets $\{x \in A | \phi(x)\}$.

⁸For instance, in the case of ML_T the elements of L are well-formed equality judgements $\Gamma \vdash_{ml} e_1 =_\tau e_2$ and $P \vdash C$ iff there exists a derivation of C , where all assumptions are in P .

Proof The first result follows from Theorem 4.9, which implies that for every model \mathcal{C} of ML_T the Yoneda embedding maps the interpretation of an ML_T -term in \mathcal{C} to its interpretation in $\hat{\mathcal{C}}$, and the faithfulness of the Yoneda embedding, which implies that two ML_T -terms have the same interpretation in \mathcal{C} iff they have the same interpretation in $\hat{\mathcal{C}}$. The second result follows, because the Yoneda embedding preserves function spaces. The third conservative extension result follows immediately from the first two. ■

The above result means that we can think of computations naively in terms of sets and functions, provided we treat them intuitionistically, and can use the full apparatus of higher-order (intuitionistic) logic instead of the less expressive many sorted equational logic.

Before giving a conservative extension result for the programming language, we have to express the mono requirement, equivalence and existence in HML_T . The idea is to extend the translation from PL-terms to ML_T -terms given in Definition 2.7 and exploit the increased expressiveness of HML_T over ML_T to axiomatise the mono requirement and translate existence and equivalence assertions (see Remark 2.1):

- the **mono requirement** for τ , i.e. η_τ is mono, is axiomatised by

$$\text{mono}.\tau \ (\forall x, y: \tau. [x]_T =_{T\tau} [y]_T \rightarrow x =_\tau y)$$

- the **equalising requirement** for τ , i.e. η_τ is the equaliser of $T(\eta_\tau)$ and $\eta_{T\tau}$, is axiomatised by $(\text{mono}.\tau)$ and the axiom

$$\text{eqls}.\tau \ (\forall x: T\tau. [x]_T =_{T^2\tau} (\text{let}_T y \Leftarrow x \text{ in } [[y]_T]_T) \rightarrow (\exists! y: \tau. x =_{T\tau} [y]_T))$$

- the **translation** \circ is extended to assertions and functional types as follows:

$$\begin{aligned} - (e_1 \equiv_\tau e_2)^\circ &\stackrel{\Delta}{=} e_1^\circ =_{T\tau} e_2^\circ \\ - (e_1 \downarrow_\tau)^\circ &\stackrel{\Delta}{=} (\exists! x: \tau. e_1^\circ =_{T\tau} [x]_T) \\ - (\tau_1 \rightarrow \tau_2)^\circ &\stackrel{\Delta}{=} \tau_1^\circ \rightarrow T\tau_2^\circ \end{aligned}$$

Theorem 4.4 $\text{HML}_T + \{(\text{mono}.\tau) \mid \tau \text{ type of PL}\}$ (i.e. τ is built using only base types, 1, TA , and $A \times B$) is a conservative extension of PL (after translation). Similarly, $\text{HML}_T + \{(\text{mono}.\tau) \mid \tau \text{ type of } \lambda_c\text{PL}\}$ (i.e. τ is built using only base types, 1, TA , $A \times B$ and $A \rightarrow B$) is a conservative extension of $\lambda_c\text{PL}$ (after translation).

Proof The proof proceeds as in the previous theorem. The only additional step is to show that for every type τ of PL (or $\lambda_c\text{PL}$) the axiom $(\text{mono}.\tau)$ holds in $\hat{\mathcal{C}}$, under the assumption that \mathcal{C} satisfies the mono requirement. Let c be the interpretation of τ in \mathcal{C} (therefore Yc is the interpretation of τ in $\hat{\mathcal{C}}$), then the axiom $(\text{mono}.\tau)$ holds in $\hat{\mathcal{C}}$ provided $\hat{\eta}_{Yc}$ is a mono. η_c is mono (by the mono requirement), so $\hat{\eta}_{Yc} = Y(\eta_c)$ is mono (as Y preserves monos). ■

In the theorem above only types from the programming language have to satisfy the mono requirement. Indeed, $\text{HML}_T + \{(\text{mono}.\tau) \mid \tau \text{ type of } \text{HML}_T\}$ is not a conservative extension of PL (or $\lambda_c\text{PL}$).

Lemma 4.5 If (T, η, μ) is a monad over a topos \mathcal{C} satisfying the mono requirement, then it satisfies also the equalising requirement.

Proof See Lemma 6 on page 110 of [BW85]. ■

In other words, for any type τ the axiom $(\text{eqls}.\tau)$ is derivable in HML_T from the set of axioms $\{(\text{mono}.\tau) \mid \tau \text{ type of } \text{HML}_T\}$. In general, when \mathcal{C} is not a topos, the mono requirement does not entail the equalising requirement; one can easily define strong monads (over an Heyting algebra) that satisfy the mono but not the equalising requirement (just take $T(A) = A \vee B$, for some element $B \neq \perp$ of the Heyting algebra). In terms of formal consequence relation this means

that in $\text{HML}_T + \text{mono}$ requirement the existence assertion $\Gamma \vdash_{pl} e \downarrow_\tau$ is derivable from $\Gamma \vdash_{pl} [e] \equiv_{T\tau} (x \Leftarrow e)$ (let $x \Leftarrow e$ in $[x]$), while such derivation is not possible in $\lambda_c\text{PL}$. We do not know whether $\text{HML}_T + \text{equalising requirement}$ is a conservative extension of $\text{PL} + \text{equalising requirement}$, or whether $\lambda_c\text{PL}$ is a conservative extension of PL .

A language which combines computations and higher order logic, like HML_T , seems to be the ideal framework for program logics that go beyond proving equivalence of programs, like Hoare's logic for partial correctness of imperative languages. In HML_T (as well as ML_T and PL) one can describe a programming language by introducing additional constant and axioms. In λML_T or $\lambda_c\text{PL}$ such constants correspond to program-constructors, for instance:

- *lookup*: $L \rightarrow TU$, which given a location $l \in L$ produces the value of such location in the current store, and *update*: $L \times U \rightarrow T1$, which changes the current store by assigning to $l \in L$ the value $u \in U$;
- *if*: $\text{Bool} \times TA \times TA \rightarrow TA$ and *while*: $T(\text{Bool}) \times T1 \rightarrow T1$;
- *new*: $1 \rightarrow TL$, which returns a newly created location;
- *read*: $1 \rightarrow TU$, which computes a value by reading it from the input, and *write*: $U \rightarrow T1$, which writes a value $u \in U$ on the output.

In HML_T one can describe also a program logic, by adding constants $p: TA \rightarrow \Omega$ corresponding to properties of computations.

Example 4.6 Let T be the monad for non-deterministic computations (see Example 1.4), then we can define a predicate $may: A \times TA \rightarrow \Omega$ such that $may(a, c)$ is true iff the value a is a possible outcome of the computation c (i.e. $a \in c$). However, there is a more uniform way of defining the *may* predicate for any type. Let $\diamond: T\Omega \rightarrow \Omega$ be the predicate such that $\diamond(X) = \top$ iff $\top \in X$, where Ω is the set $\{\perp, \top\}$ (note that $\diamond(_) = may(\top, _)$). Then, $may(a, c)$ can be defined as $\diamond(\text{let}_T x \Leftarrow c \text{ in } [a =_\tau x]_T)$.

The previous example suggests that predicates defined *uniformly* on computations of any type can be better described in terms of *modal operators* $\gamma: T\Omega \rightarrow \Omega$, relating a computation of truth values to a truth value. This possibility has not been investigated in depth, so we will give only a tentative definition.

Definition 4.7 If (T, η, μ) is a monad over a topos \mathcal{C} , then a **T -modal operator** is a T -algebra $\gamma: T\Omega \rightarrow \Omega$, i.e.

$$\begin{array}{ccccc}
 T^2\Omega & \xrightarrow{\mu_\Omega} & T\Omega & \xleftarrow{\eta_\Omega} & \Omega \\
 T\gamma \downarrow & & \downarrow \gamma & & \swarrow id_{T\Omega} \\
 T\Omega & \xrightarrow{\gamma} & \Omega & &
 \end{array}$$

where Ω is the subobject classifier in \mathcal{C} .

The commutativity of the two diagrams above can be expressed in the metalanguage:

- $x: \Omega \vdash \gamma([x]_T) \longleftrightarrow x$
- $c: T^2\Omega \vdash \gamma(\text{let } x \Leftarrow c \text{ in } x) \longleftrightarrow \gamma(\text{let } x \Leftarrow c \text{ in } [\gamma(x)]_T)$

We consider some examples and *non-examples* of modal operators.

Example 4.8 For the monad T of non-deterministic computations (see Example 1.4) there are only two modal operators \square and \diamond :

- $\square(X) = \perp$ iff $\perp \in X$;

- $\diamond(X) = \top$ iff $\top \in X$.

Given a nondeterministic computation e of type τ and a predicate $A(x)$ over τ , i.e. a term of type Ω , then $\square(\text{let}_T x \Leftarrow e \text{ in } [A(x)]_T)$ is true iff all possible results of e satisfy $A(x)$.

For the monad T of computations with side-effects (see Example 1.4) there is an operator $\square: (\Omega \times S)^S \rightarrow \Omega$ that can be used to express Hoare's triples:

- $\square f = \top$ iff for all $s \in S$ there exists $s' \in S$ s.t. $fs = \langle \top, s' \rangle$

this operator does not satisfy the second equivalence, as only one direction is valid, namely $c: T^2\Omega \vdash \gamma(\text{let } x \Leftarrow c \text{ in } [\gamma(x)]_T) \rightarrow \gamma(\text{let } x \Leftarrow c \text{ in } x)$

Let $P: U \rightarrow \Omega$ and $Q: U \times U \rightarrow \Omega$ be predicates over storable values, $e \in T1$ a computation of type 1 and $x, y \in L$ locations. The intended meaning of the triple $\{P(x)\}e\{Q(x, y)\}$ is “if in the initial state the content u of x satisfies $P(u)$, then in the final state (i.e. after executing e) the content v of y satisfies $Q(u, v)$ ”. This intended meaning can be expressed formally in terms of the modal operator \square and the program-constructors *lookup* and *update* as follows:

$$\forall u: U. P(u) \rightarrow \square(\text{let}_T v \Leftarrow (\text{update}(x, u); e; \text{lookup}(y)) \text{ in } [Q(u, v)]_T)$$

where $_ ; _ : TA \times TB \rightarrow TB$ is the derived operation $e_1; e_2 \stackrel{\Delta}{=} (\text{let}_T x \Leftarrow e_1 \text{ in } e_2)$ with x not free in e_2 .

Finally, we state the main theorem and outline its proof. In doing so we assume that the reader is familiar with non-elementary concepts from Category Theory.

Theorem 4.9 *Let \mathcal{C} be a small category, $\hat{\mathcal{C}}$ the topos of presheaves over \mathcal{C} and Y the Yoneda embedding of \mathcal{C} into $\hat{\mathcal{C}}$. Then for every monad (T, η, μ) over \mathcal{C} , there exists a monad $(\hat{T}, \hat{\eta}, \hat{\mu})$ over $\hat{\mathcal{C}}$ such that the following diagram commutes⁹*

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{T} & \mathcal{C} \\ Y \downarrow & & \downarrow Y \\ \hat{\mathcal{C}} & \xrightarrow{\hat{T}} & \hat{\mathcal{C}} \end{array}$$

and for all $a \in \mathcal{C}$ the following equations hold

$$\hat{\eta}_{Ya} = Y(\eta_a) \quad , \quad \hat{\mu}_{Ya} = Y(\mu_a)$$

Moreover, for every strong monad (T, η, μ, t) over \mathcal{C} , there exists a natural transformation \hat{t} such that $(\hat{T}, \hat{\eta}, \hat{\mu}, \hat{t})$ is a strong monad over $\hat{\mathcal{C}}$ and for all $a, b \in \mathcal{C}$ the following equation holds

$$\hat{t}_{Ya, Yb} = Y(t_{a,b})$$

where we have implicitly assumed that the Yoneda embedding preserves finite products on the nose, i.e. the following diagrams commute

$$\begin{array}{ccccc} 1 & \xrightarrow{1} & \mathcal{C} & \xleftarrow{\times} & \mathcal{C} \times \mathcal{C} \\ & \searrow 1 & \downarrow Y & & \downarrow Y \times Y \\ & & \hat{\mathcal{C}} & \xleftarrow{\times} & \hat{\mathcal{C}} \times \hat{\mathcal{C}} \end{array}$$

⁹This is a simplifying assumption. For our purposes it would be enough to have a natural isomorphism $\sigma: T; Y \rightarrow Y; \hat{T}$, but then the remaining equations have to be patched. For instance, the equation relating η and $\hat{\eta}$ would become $\hat{\eta}_{Ya} = Y(\eta_a); \sigma_a$.

and for all $a, b \in \mathcal{C}$. the following equations hold

$$!_{Y_a} = Y(!_a) \quad , \quad \pi_i^{Y_a, Y_b} = Y(\pi_i^{a,b})$$

Definition 4.10 ([Mac71]) Let $T: \mathcal{C} \rightarrow \mathcal{D}$ be a functor between two small categories and \mathcal{A} a cocomplete category. Then, the **left Kan extension** $L_T^{\mathcal{A}}: \mathcal{A}^{\mathcal{C}} \rightarrow \mathcal{A}^{\mathcal{D}}$ is the left adjoint of \mathcal{A}^T and can be defined as follows:

$$L_T^{\mathcal{A}}(F)(d) = \text{Colim}_{T \downarrow d}^{\mathcal{A}}(\pi; F)$$

where $F: \mathcal{C} \rightarrow \mathcal{A}$, $d \in \mathcal{D}$, $T \downarrow d$ is the comma category whose objects are pairs $\langle c \in \mathcal{C}, f: Tc \rightarrow d \rangle$, $\pi: T \downarrow d \rightarrow \mathcal{C}$ is the projection functor (mapping a pair $\langle c, f \rangle$ to c) and $\text{Colim}_I^{\mathcal{A}}: \mathcal{A}^I \rightarrow \mathcal{A}$ (with I small category) is a functor mapping an I -diagram in \mathcal{A} to its colimit.

The following proposition is a 2-categorical reformulation of Theorem 1.3.10 of [MR77]. For the sake of simplicity, we use the strict notions of 2-functor and 2-natural transformation, although we should have used pseudo-functors and pseudo-natural transformations.

Proposition 4.11 Let \mathbf{Cat} be the 2-category of small categories, \mathbf{CAT} the 2-category of locally small categories and $\underline{\cdot}: \mathbf{Cat} \rightarrow \mathbf{CAT}$ the inclusion 2-functor. Then, the following $\hat{\cdot}: \mathbf{Cat} \rightarrow \mathbf{CAT}$ is a 2-functor

- if \mathcal{C} is a small category, then $\hat{\mathcal{C}}$ is the topos of presheaves $\mathbf{Set}^{\mathcal{C}^{op}}$
- if $T: \mathcal{C} \rightarrow \mathcal{D}$ is a functor, then \hat{T} is the left Kan extension $L_{T^{op}}^{\mathbf{Set}}$
- if $\sigma: S \rightarrow T: \mathcal{C} \rightarrow \mathcal{D}$ is a natural transformation and $F \in \hat{\mathcal{C}}$, then $\hat{\sigma}_F$ is the natural transformation corresponding to $\text{id}_{\hat{T}F}$ via the following sequence of steps

$$\begin{array}{ccc} \hat{\mathcal{C}}(F, T^{op}; \hat{T}F) & \xleftarrow{\sim} & \hat{\mathcal{D}}(\hat{T}F, \hat{T}F) \\ \downarrow & & \downarrow \\ \hat{\mathcal{C}}(F, \sigma^{op}; \hat{T}F) & & \\ \downarrow & & \\ \hat{\mathcal{C}}(F, S^{op}; \hat{T}F) & \xleftarrow{\sim} & \hat{\mathcal{D}}(\hat{S}F, \hat{T}F) \end{array}$$

Moreover, $Y: \underline{\cdot} \rightarrow \hat{\cdot}$ is a 2-natural transformation.

Since monads are a 2-categorical concept (see [Str72]), the 2-functor $\hat{\cdot}$ maps monads in \mathbf{Cat} to monads in \mathbf{CAT} . Then, the statement of Theorem 4.9 about lifting of monads follows immediately from Proposition 4.11. It remains to define the lifting \hat{t} of a tensorial strength t for a monad (T, η, μ) over a small category \mathcal{C} .

Proposition 4.12 If \mathcal{C} is a small category with finite products and T is an endofunctor over \mathcal{C} , then for every natural transformation $t_{a,b}: a \times Tb \rightarrow T(a \times b)$ there exists a unique natural transformation $\hat{t}_{F,G}: F \times \hat{T}G \rightarrow \hat{T}(F \times G)$ s.t. $\hat{t}_{Y_a, Y_b} = Y(t_{a,b})$ for all $a, b \in \mathcal{C}$.

Proof Every $F \in \hat{\mathcal{C}}$ is isomorphic to the colimit $\text{Colim}_{Y \downarrow F}^{\hat{\mathcal{C}}}(\pi; Y)$ (shortly $\text{Colim}_i Y_i$), where Y is the Yoneda embedding of \mathcal{C} into $\hat{\mathcal{C}}$. Similarly G is isomorphic to $\text{Colim}_j Y_j$. Both functors $(\underline{\cdot} \times \hat{T}\underline{\cdot})$ and $\hat{T}(\underline{\cdot} \times \underline{\cdot})$ from $\hat{\mathcal{C}} \times \hat{\mathcal{C}}$ to $\hat{\mathcal{C}}$ preserves colimits (as \hat{T} and $\underline{\cdot} \times F$ are left adjoints) and commute with the Yoneda embedding (as $Y(a \times b) = Ya \times Yb$ and $\hat{T}(Ya) = Y(Ta)$). Therefore, $F \times \hat{T}G$ and $\hat{T}(F \times G)$ are isomorphic to the colimits $\text{Colim}_{i,j} Y_i \times \hat{T}(Y_j)$ and $\text{Colim}_{i,j} \hat{T}(Y_i \times Y_j)$ respectively.

Let \hat{t} be the natural transformation we are looking for, then

$$\begin{array}{ccc} Y_i \times \hat{T}(Y_j) & \xrightarrow{Y(t_{i,j})} & \hat{T}(Y_i \times Y_j) \\ f \times \hat{T}g \downarrow & & \downarrow \hat{T}(f \times g) \\ F \times \hat{T}(G) & \xrightarrow{\hat{t}_{F,G}} & \hat{T}(F \times G) \end{array}$$

for all $f: Yi \rightarrow F$ and $g: Yj \rightarrow g$ (by naturality of \hat{t} and $\hat{t}_{Yi,Yj} = Y(t_{i,j})$). But there exists exactly one morphism $\hat{t}_{F,G}$ making the diagram above commute, as $\langle t_{i,j}|i,j\rangle$ is a morphism between diagrams in $\hat{\mathcal{C}}$ of the same shape, and these diagrams have colimit cones $\langle f \times \hat{T}g|f,g\rangle$ and $\langle \hat{T}(f \times g)|f,g\rangle$ respectively. ■

Remark 4.13 If T is a monad of partial computations, i.e. it is induced by a dominion \mathcal{M} on \mathcal{C} s.t. $P(\mathcal{C}, \mathcal{M})(a, b) \cong \mathcal{C}(a, Tb)$, then the lifting \hat{T} is the monad of partial computations induced by the dominion $\hat{\mathcal{M}}$ on $\hat{\mathcal{C}}$, obtained by lifting \mathcal{M} to the topos of presheaves, as described in [Ros86]. For other monads, however, the lifting is not the *expected* one. For instance, if T is the monad of side-effects $(_- \times S)^S$, then \hat{T} is not (in general) the endofunctor $(_- \times YS)^{YS}$ on the topos of presheaves.

Conclusions and further research

The main contribution of this paper is the category-theoretic semantics of computations and the general principle for extending it to more complex languages (see Remark 3.3 and Section 4), while the formal systems presented are a straightforward fallout, easy to understand and relate to other calculi.

Our work is just an example of what can be achieved in the study of programming languages by using a category-theoretic methodology, which avoids irrelevant syntactic detail and focus instead on the important structures underlying programming languages. We believe that there is a great potential to be exploited here. Indeed, in [Mog89b] we give a categorical account of phase distinction and program modules, that could lead to the introduction of higher order modules in programming languages like ADA or ML (see [HMM90]), while in [Mog89a] we propose a “modular approach” to Denotational Semantics based on the idea of monad-constructor (i.e. an endofunctor on the category of monads over a category \mathcal{C}).

The metalanguage open also the possibility to develop a new Logic of Computable Functions (see [Sco69]), based on an abstract semantic of computations rather than domain theory, for studying axiomatically different notions of computation and their relations. Some recent work by Crole and Pitts (see [CP90]) has consider an extension of the metalanguage equipped with a logic for inductive predicates, which goes beyond equational reasoning. A more ambitious goal would be to try exploiting the capabilities offered by higher-order logic in order to give a uniform account of various program logics, based on the idea of “ T -modal operator” (see Definition 4.7).

The semantics of computations corroborates the view that (constructive) proofs and programs are rather unrelated, although both of them can be understood in terms of functions. Indeed, monads (and comonads) used to model logical modalities, e.g. possibility and necessity in modal logic or why not and of course of linear logic, usually do not have a tensorial strength. In general, one should expect types suggested by logic to provide a more fine-grained type system without changing the *nature* of computations.

We have identified monads as important to model notions of computations, but *computational monads* seem to have additional properties, e.g. they have a tensorial strength and may satisfy the mono requirement. It is likely that there are other properties of computational monads still to be identified, and there is no reason to believe that such properties have to be found in the literature on monads.

Acknowledgements

I have to thank many people for advice, suggestions and criticisms, in particular: R. Amadio, R. Burstall, M. Felleisen, R. Harper, F. Honsell, M. Hyland, B. Jay, A. Kock, Y. Lafont, G. Longo, R. Milner, A. Pitts, G. Plotkin, J. Power and C. Talcott.

References

- [BW85] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer Verlag, 1985.

- [CP90] R.L. Crole and A.M. Pitts. New foundations for fixpoint computations. In *4th LICS Conf.* IEEE, 1990.
- [CS87] R.L. Constable and S.F. Smith. Partial objects in constructive type theory. In *2nd LICS Conf.* IEEE, 1987.
- [CS88] R.L. Constable and S.F. Smith. Computational foundations of basic recursive function theory. In *3rd LICS Conf.* IEEE, 1988.
- [FF89] M. Felleisen and D.P. Friedman. A syntactic theory of sequential state. *Theoretical Computer Science*, 69(3), 1989.
- [FFKD86] M. Felleisen, D.P. Friedman, E. Kohlbecker, and B. Duba. Reasoning with continuations. In *1st LICS Conf.* IEEE, 1986.
- [Fou77] M.P. Fourman. The logic of topoi. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic*. North Holland, 1977.
- [GMW79] M.J.C. Gordon, R. Milner, and C.P. Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation*, volume 78 of *Lecture Notes in Computer Science*. Springer Verlag, 1979.
- [GS89] C. Gunter and S. Scott. Semantic domains. Technical Report MS-CIS-89-16, Dept. of Comp. and Inf. Science, Univ. of Pennsylvania, 1989. to appear in North Holland Handbook of Theoretical Computer Science.
- [HMM90] R. Harper, J. Mitchell, and E. Moggi. Higher-order modules and the phase distinction. In *17th POPL*. ACM, 1990.
- [HP87] J.M.E. Hyland and A.M. Pitts. The theory of constructions: Categorical semantics and topos-theoretic models. In *Proc. AMS Conf. on Categories in Comp. Sci. and Logic (Boulder 1987)*, 1987.
- [JP78] P.T. Johnstone and R. Pare, editors. *Indexed Categories and their Applications*, volume 661 of *Lecture Notes in Mathematics*. Springer Verlag, 1978.
- [Kel82] G.M. Kelly. *Basic Concepts of Enriched Category Theory*. Cambridge University Press, 1982.
- [Koc72] A. Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23, 1972.
- [KR77] A. Kock and G.E. Reyes. Doctrines in categorical logic. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic*. North Holland, 1977.
- [Laf88] Y. Lafont. The linear abstract machine. *Theoretical Computer Science*, 59, 1988.
- [LS86] J. Lambek and P.J. Scott. *Introduction to Higher-Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1986.
- [Mac71] S. MacLane. *Categories for the Working Mathematician*. Springer Verlag, 1971.
- [Man76] E. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. Springer Verlag, 1976.
- [Mas88] I.A. Mason. Verification of programs that destructively manipulate data. *Science of Computer Programming*, 10, 1988.
- [Mog86] E. Moggi. Categories of partial morphisms and the partial lambda-calculus. In *Proceedings Workshop on Category Theory and Computer Programming, Guildford 1985*, volume 240 of *Lecture Notes in Computer Science*. Springer Verlag, 1986.

- [Mog88] E. Moggi. *The Partial Lambda-Calculus*. PhD thesis, University of Edinburgh, 1988.
- [Mog89a] E. Moggi. An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Edinburgh Univ., Dept. of Comp. Sci., 1989. Lecture Notes for course CS 359, Stanford Univ.
- [Mog89b] E. Moggi. A category-theoretic account of program modules. In *Proceedings of the Conference on Category Theory and Computer Science, Manchester, UK, Sept. 1989*, volume 389 of *Lecture Notes in Computer Science*. Springer Verlag, 1989.
- [Mog89c] E. Moggi. Computational lambda-calculus and monads. In *4th LICS Conf.* IEEE, 1989.
- [Mos89] P. Mosses. Denotational semantics. Technical Report MS-CIS-89-16, Dept. of Comp. and Inf. Science, Univ. of Pennsylvania, 1989. to appear in North Holland Handbook of Theoretical Computer Science.
- [MR77] M. Makkai and G. Reyes. *First Order Categorical Logic*. Springer Verlag, 1977.
- [MT89a] I. Mason and C. Talcott. Programming, transforming, and proving with function abstractions and memories. In *16th Colloquium on Automata, Languages and Programming*. EATCS, 1989.
- [MT89b] I. Mason and C. Talcott. A sound and complete axiomatization of operational equivalence of programs with memory. In *POPL 89*. ACM, 1989.
- [Plo75] G.D. Plotkin. Call-by-name, call-by-value and the λ -calculus. *Theoretical Computer Science*, 1, 1975.
- [Plo85] G.D. Plotkin. Denotational semantics with partial functions. Lecture Notes at C.S.L.I. Summer School, 1985.
- [Ros86] G. Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, University of Oxford, 1986.
- [Sch86] D.A. Schmidt. *Denotational Semantics: a Methodology for Language Development*. Allyn & Bacon, 1986.
- [Sco69] D.S. Scott. A type-theoretic alternative to CUCH, ISWIM, OWHY. Oxford notes, 1969.
- [Sco79] D.S. Scott. Identity and existence in intuitionistic logic. In M.P. Fourman, C.J. Mulvey, and D.S. Scott, editors, *Applications of Sheaves*, volume 753 of *Lecture Notes in Mathematics*. Springer Verlag, 1979.
- [Sco80] D.S. Scott. Relating theories of the λ -calculus. In R. Hindley and J. Seldin, editors, *To H.B. Curry: essays in Combinatory Logic, lambda calculus and Formalisms*. Academic Press, 1980.
- [See87] R.A.G. Seely. Linear logic, $*$ -autonomous categories and cofree coalgebras. In *Proc. AMS Conf. on Categories in Comp. Sci. and Logic (Boulder 1987)*, 1987.
- [Sha84] K. Sharma. Syntactic aspects of the non-deterministic lambda calculus. Master's thesis, Washington State University, September 1984. available as internal report CS-84-127 of the comp. sci. dept.
- [SP82] M. Smith and G. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11, 1982.
- [Str72] R. Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2, 1972.

OPERADIC FIBRATIONS AND UNARY OPERADIC 2-CATEGORIES

DOMINIK TRNKA

ABSTRACT. We introduce unary operadic 2-categories as a framework for operadic Grothendieck construction for categorical \mathbb{O} -operads, \mathbb{O} being a unary operadic category. The construction is a fully faithful functor $f_{\mathbb{O}}$ which takes categorical \mathbb{O} -operads to operadic functors over \mathbb{O} , and we characterise its essential image by certain lifting properties. Such operadic functors are called operadic fibration. Our theory is an extension of the discrete (unary) operadic case and, in some sense, of the classical Grothendieck construction of a categorical presheaf. For the terminal unary operadic category \odot , a categorical \odot -operad is a strict monoidal category \mathcal{V} and its Grothendieck construction $f_{\odot}\mathcal{V}$ is connected to the ‘Para’ construction appearing in machine learning. The 2-categorical setting provides a characterisation of \mathbb{O} -operads valued in \mathcal{V} as operadic functors $\mathbb{O} \rightarrow f_{\odot}\mathcal{V}$. Last, we describe a left adjoint to f_{\odot} .

Contents

1	Unary Operadic Categories and Their Operads	5
2	The Nerve and Décalage for 2-categories	9
3	Unary Operadic 2-categories	12
4	Operads of Unary Operadic 2-Categories	18
5	Operadic Grothendieck construction	19
6	Operadic fibrations	22
7	Left adjoint to f_{\odot}	24
8	Closing remarks and further directions	28

Introduction



THE theory of operadic categories was introduced in [BM15] and further developed in [BM23a, BM23b], and [BM24], as a unifying framework for compositional structures and their homotopy theory. The last of the above mentioned works focuses only on ‘unary’ operadic categories which themselves describe a lot of known structures. On one hand, every discrete decomposition space is a unary operadic category [GKW21]. On the other, operads of certain unary operadic categories are monoids, categories, sharades, &c. In [BM24], unary operadic categories were used for a characterisation of the ‘Blob complex’ of [MW12] as a bar construction of a certain

Supported by RVO: 67985840 and Praemium Academiæof M. Markl; Brno Ph.D. Talent Scholarship Holder - Funded by the Brno City Municipality; Supported by MUNI/A/1099/2022: Specific research - support for student projects.

© Dominik Trnka, 2024. Permission to copy for private use granted.

operad. The unary case often serves as a toy model for new directions in the theory of (non-unary) operadic categories, cf. [M24, H23].

It is well known that the Grothendieck construction establishes a correspondence of categorical presheaves and categorical fibrations. In the operadic context, the Grothendieck construction for Set-valued operads, as well as discrete operadic fibrations, were introduced already in the first paper [BM15]. The construction is a part of an equivalence

$$\mathbb{O}\text{-oper}(Set) \simeq \text{DoFib}(\mathbb{O})$$

between Set-valued \mathbb{O} -operads discrete operadic fibrations over an operadic category \mathbb{O} .

Our goal is to establish a correspondence

$$\mathbb{O}\text{-oper}(Cat) \simeq \text{oFib}(\mathbb{O})$$

between categorical \mathbb{O} -operads and operadic functors into \mathbb{O} with certain lifting properties, which we call operadic fibrations. The lifting properties are operadic analogues of the properties of categorical fibrations. Categorical operads appear in the literature quite often, an example is an operad of leveled trees. The operations in arity n are leveled trees with n leaves, and there is a unique isomorphism of trees if they differ by an admissible change of leveling, cf. Figure 1. This operad plays a key rôle in the construction of free operads [BM23b, T].

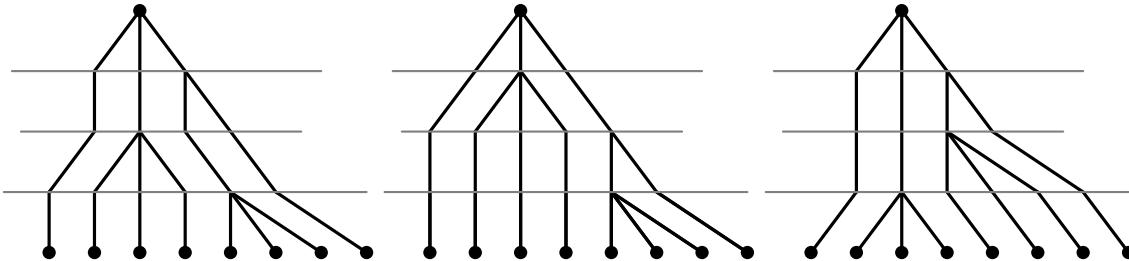
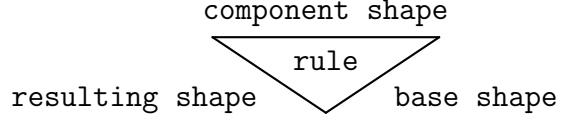


Figure 1: Three isomorphic leveled trees.

It turns out that the framework of operadic (1-)categories is not sufficient for the correspondence, and that a 2-categorical structure is needed. This was the motivation for introducing a theory of operadic 2-categories, but we soon realised that the 2-categorical setting brings other benefits. Namely, one can identify \mathbb{O} -operads with operadic functors out of \mathbb{O} . This interprets \mathbb{O} -operads as generalised presheaves on \mathbb{O} , so operads are no longer isolated objects in the theory of operadic categories. Last, we mention that a categorical operad of the terminal operadic category is a strict monoidal category and its operadic Grothendieck construction is related to the ‘Para’ construction which appears in machine learning, for example in [FST21].

This paper derives the results in the unary case only. An extension of the theory to general operadic categories can be expected in our future work. Before describing the results in more detail, we give an intuitive idea about unary operadic categories, their operads, and the operadic Grothendieck construction.

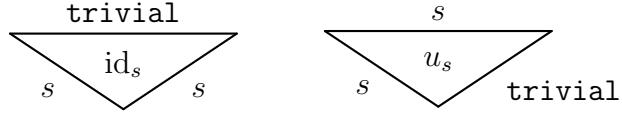
SHAPES AND STRUCTURES. Think of a unary operadic category \mathbb{O} as of a calculus of shapes of a given type. Such calculus is specified by a class of *shapes* (objects of \mathbb{O}) and triangular *composition rules* (morphisms of \mathbb{O}) of the form



A rule consists of three shapes and an information of how to compose the shapes. From the categorical perspective, additionally to source and target, every map

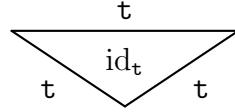
$$\text{resulting shape} \xrightarrow{\text{rule}} \text{base shape}$$

of \mathbb{O} also has a *fiber*, which is the **component shape**. The fiber assignment is assembled into functors $\varphi_s: \mathbb{O}/s \rightarrow \mathbb{O}$ from the slice category, for all shapes s . Lastly, one often needs to work with *trivial shapes* and two trivial composition rules

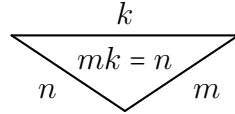


for every shape s , where both have a meaning of ‘do nothing with s ’. Four examples are as follows:

- The trivial calculus \mathbb{O} consist only of one trivial shape t and trivial composition rule

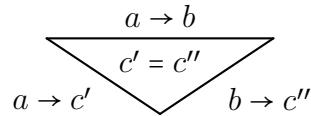


- For a monoid M , the shapes of calculus M^Δ are elements of M . The composition rules are



whenever $mk = n$. The trivial shape is the unit e of M .

- For a fixed set S , the shapes of calculus $Bq^u(A)$ are pairs (a, b) , written as $a \rightarrow b$, of elements of S . The composition rules are



whenever $c' = c''$. The trivial shapes are $a \rightarrow a$, for every $a \in S$.

- For a fixed manifold \mathcal{M} , the shapes of calculus $\text{sub}(\mathcal{M})$ are submanifolds of \mathcal{M} .
The composition rules are

$$\begin{array}{c} B - A \\ \triangle \\ A \subseteq B \\ \backslash \quad / \\ B \qquad A \end{array}$$

whenever $A \subseteq B$. The trivial shape is the empty submanifold.

For a calculus \mathbb{O} , an \mathbb{O} -structure (\mathbb{O} -operad) \mathcal{P} is a compositional structure which consists of sets \mathcal{P}_s for every shape $s \in \mathbb{O}$, and composition maps

$$\mu_{\text{rule}}: \mathcal{P}_{\text{base}} \times \mathcal{P}_{\text{comp}} \rightarrow \mathcal{P}_{\text{result}}$$

indexed by the composition rules. These maps are assumed to be unital and associative, which is expressed in terms of trivial objects and axioms of shape calculus (operadic categories), which we did not state. For the examples above we recover:

- \mathbb{O} -structures are monoids,
- M^Δ -structures are monoidal functors $M \rightarrow (\text{Set}, \times, 1)$, where M is regarded as a discrete monoidal category,
- $\text{Bq}^u(S)$ -structures are categories with set of objects S , and
- $\text{sub}(M)$ -structures are structures describing, for example, gluing of local functions on the manifold M .

The framework of operadic categories offers an interesting interplay between the shape calculi and their structures. Having a \mathbb{O} -structure \mathcal{P} , its Grothendieck construction $\int_{\mathbb{O}} \mathcal{P}$ produces a calculus, whose shapes are all elements of the structure \mathcal{P} and composition rules are given by the composition maps of \mathcal{P} . For example, for a monoid M , regarded as a \mathbb{O} -structure, one has $\int_{\mathbb{O}} M = M^\Delta$. A natural question is what the construction should be for structures valued in categories, and what are its properties. Namely, what is an operadic Grothendieck construction of monoidal category or of a 2-category. This paper answers the question. Let us describe our results in more detail.

MAIN RESULTS. We introduce Definition 3.1 of a unary operadic 2-category, which is motivated by the characterisation of unary operadic categories of [GKW21] via simplicial sets and décalage, recalled in Proposition 2.5. We reformulate the definition in terms of (lax normalised) fiber functors and a choice of local (lali-)terminal objects, which resembles the original definition of [BM15], cf. Definition 1.1. In Definition 4.2 we introduce operads of unary operadic 2-categories. The difference from the 1-categorical case is that the associativity axiom depends on lax triangles of \mathbb{O} instead of strict ones, but more importantly, in the 2-categorical framework a (1-connected) \mathbb{O} -operad can be seen as an operadic functor $\mathcal{P}: \mathbb{O} \rightarrow \mathcal{V}^\Delta$, where the target is an operadic 2-category of lax triangles

in a monoidal category \mathcal{V} . For $\mathcal{V} = \text{Cat}$, we imitate techniques known for presheaves and define the operadic Grothendieck construction of a (1-connected) categorical operad as a certain pullback, cf. Definition 5.2. The extension to non-1-connected operads is immediate and the construction is a fully faithful functor

$$\int_0 : \mathbb{O}\text{-oper}(\text{Cat}) \rightarrow \text{uOpCat}/\mathbb{O}$$

into the category of unary operadic 2-categories over \mathbb{O} . Our main result is Theorem 6.9, which characterises the image of \int_0 as operadic functors into \mathbb{O} with certain lifting properties. The properties are operadic analogues of the lifting properties of classical categorical split fibrations and we call the functors split operadic fibrations, cf. Definition 6.5. Lastly, in the special case of $\mathbb{O} = \odot$, the terminal unary operadic 2-category, Section 7 describes a left adjoint to \int_0 , which gives a comparison of unary operadic 2-categories to strict monoidal categories:

$$\text{StrMonCat} \xrightleftharpoons[\Phi]{\tau} \text{uOpCat}.$$

The adjunction follows from an adjunction

$$\text{StrMonCat} \rightleftarrows \text{sSet}_{\leq 3},$$

which might be of an independent interest.

1. Unary Operadic Categories and Their Operads



HIS introductory section recalls the definition of a (unital) unary operadic category \mathbb{O} , the definition of a (unital) \mathbb{O} -operad, operadic Grothendieck construction for set-values operads, and discrete operadic fibrations. The material is mostly taken from [BM24]. We also recall the result of [GKW21], which characterises unary operadic categories using simplicial sets and décalage construction, and we include some examples.

Let us denote by 1_x the identity morphism on an object x . For a category \mathcal{C} and its object x , let \mathcal{C}/x denote the slice category over x . Observe that any functor $F: \mathcal{C} \rightarrow \mathcal{D}$ induces a functor on slices $F/x: \mathcal{C}/x \rightarrow \mathcal{D}/F(x)$. We denote by Set the category of sets and Cat stands for the category of (small) categories.

1.1. DEFINITION. *A unary operadic category is a category \mathbb{O} together with a family of functors $\text{fib}_x: \mathbb{O}/x \rightarrow \mathbb{O}$, indexed by objects x of \mathbb{O} , such that for any map $f: y \rightarrow x$ of \mathbb{O} , the following diagram commutes.*

$$\begin{array}{ccc} (\mathbb{O}/x)/f & \xrightarrow{\text{fib}_x/f} & \mathbb{O}/\text{fib}_x(f) \\ \text{dom}/f \downarrow & & \downarrow \text{fib}_{\text{fib}_x(f)} \\ \mathbb{O}/y & \xrightarrow{\text{fib}_y} & \mathbb{O}. \end{array} \tag{1}$$

We further require for every connected component c of \mathbb{O} a choice of local terminal object u_c , which belongs to the component c , such that $\text{fib}_{u_c}:\mathbb{O}/u_c \rightarrow \mathbb{O}$ is the domain functor, and for every object x of \mathbb{O} , $\text{fib}_x(\mathbb{1}_x)$ is one of the objects u_c .

The objects u_c are called *trivial* and the functors fib_x are called *fiber* functors. For an object $f:y \rightarrow x$ of \mathbb{O}/x , the value $\text{fib}_x(f)$ is called the *fiber* of f and it will be denoted simply by $\text{fib}(f)$. For a morphism $g:f \circ g \rightarrow f$ in \mathbb{O}/x , the induced map between fibers $\text{fib}_x(g):\text{fib}(f \circ g) \rightarrow \text{fib}(f)$ will be denoted by g_f . A *quasibijection* is a morphism whose fiber is trivial. The fact that an object r is a fiber of a morphism $f:y \rightarrow x$ is usually expressed by

$$r \triangleright y \xrightarrow{f} x.$$

From the simplicial viewpoint on unary operadic categories, recalled in Proposition 1.12 below, one can view the map f as a triangle:

$$\begin{array}{ccc} \cdot & \xrightarrow{r} & \cdot \\ & \searrow^f & \swarrow \\ & \cdot & \end{array}$$

1.2. DEFINITION. An *operadic functor* between unary operadic categories is a functor which respects the fiber functors and preserves trivial objects.

1.3. DEFINITION. Let \mathbb{O} be a unary operadic category and \mathcal{V} a monoidal category. A (*unital*) \mathbb{O} -operad \mathcal{P} with values in \mathcal{V} is a collection of objects $\mathcal{P}_x \in \mathcal{V}$, indexed by objects of \mathbb{O} , together with composition maps

$$\mathcal{P}_x \otimes \mathcal{P}_r \xrightarrow{\mathcal{P}_f} \mathcal{P}_y,$$

for every $r \triangleright y \xrightarrow{f} x$ in \mathbb{O} , which are associative and unital in the following sense. For any composable maps $r \triangleright y \xrightarrow{f} x$ and $s \triangleright z \xrightarrow{g} y$, with $t \triangleright z \xrightarrow{f \circ g} x$ and the induced map $s \triangleright r \xrightarrow{g_f} t$, the diagram

$$\begin{array}{ccc} \mathcal{P}_x \otimes \mathcal{P}_r \otimes \mathcal{P}_s & \xrightarrow{\mathcal{P}_f \otimes \mathcal{P}_s} & \mathcal{P}_y \otimes \mathcal{P}_s \\ \downarrow \mathcal{P}_x \otimes \mathcal{P}_{g_f} & & \downarrow \mathcal{P}_g \\ \mathcal{P}_x \otimes \mathcal{P}_t & \xrightarrow{\mathcal{P}_{f \circ g}} & \mathcal{P}_z \end{array}$$

commutes. For every connected component c of \mathbb{O} , the operad \mathcal{P} is equipped with unit maps

$$\varepsilon_c:I \rightarrow \mathcal{P}_{u_c},$$

from the monoidal unit I of \mathcal{V} , such that for any object x of \mathbb{O} with $\text{fib}(\mathbb{1}_x) = u_c$, the diagram

$$\begin{array}{ccc} \mathcal{P}_x & \xrightarrow{\mathbb{1}} & \mathcal{P}_x \\ \cong \downarrow & & \uparrow \mathcal{P}_{\mathbb{1}_x} \\ \mathcal{P}_x \otimes I & \xrightarrow{\mathcal{P}_x \otimes \varepsilon_c} & \mathcal{P}_x \otimes \mathcal{P}_{u_c} \end{array}$$

commutes, and for any $x \xrightarrow{!_d} u_d$, the unique map to one of the chosen local terminals, the diagram

$$\begin{array}{ccc} \mathcal{P}_x & \xrightarrow{\mathbb{1}} & \mathcal{P}_x \\ \cong \downarrow & & \uparrow \mathcal{P}_{!d} \\ I \otimes \mathcal{P}_x & \xrightarrow{\varepsilon_d \otimes \mathcal{P}_x} & \mathcal{P}_{u_d} \otimes \mathcal{P}_x \end{array}$$

commutes.

When the objects of the category \mathcal{V} are concrete (e.g. when \mathcal{V} is Set or Cat), for elements $a \in \mathcal{P}_x$, $b \in \mathcal{P}_r$ we write $a \cdot_f b$ instead of $\mathcal{P}_f(a, b)$. The above diagrams translate as

$$(a \cdot_f b) \cdot_g c = a \cdot_{f \circ g} (b \cdot_{g_f} c),$$

$$a \cdot_{\mathbb{1}_x} e_c = a,$$

$$e_c \cdot_! b = b,$$

where e_c is the image of ε_c .

1.4. EXAMPLE. Let \odot be the terminal category with one object $*$ and one morphism $\mathbb{1}_*$. It is an operadic category with $* \triangleright * \xrightarrow{\mathbb{1}_*} *$. Analogously to [BM24, Exercise 39], \odot -operads in Cat are strict monoidal categories.

1.5. EXAMPLE. Let A be a set. The operadic category $\text{Bq}^u(A)$ of unary bouquets has as objects pairs (a, b) of elements of A . There is a unique morphism $(a', b') \rightarrow (a'', b'')$ whenever $b' = b''$, with fiber (a', a'') . Consequently, $\text{Bq}^u(A)$ -operads with values in Set are small categories whose set of objects is A , and $\text{Bq}^u(A)$ -operads valued in Cat are (strict) 2-categories with set of objects A .

1.6. EXAMPLE. Let (\mathcal{M}, \cdot, e) be a strict monoidal category. We construct the category \mathcal{M}^Δ as follows. The objects of \mathcal{M}^Δ are the same as objects of \mathcal{M} , but a morphisms $y \rightarrow x$ of \mathcal{M}^Δ is a pair (a, α) , where $a \in M$ and $\alpha: x \cdot a \rightarrow y$ is a map in \mathcal{M} . The identity on an object x is the pair $(e, \mathbb{1}_x)$ and the composite of two morphisms

$$z \xrightarrow{(a, \alpha)} y \xrightarrow{(b, \beta)} x$$

is $(b \cdot a, \alpha \circ (\beta \cdot a))$:

$$x \cdot b \cdot a \xrightarrow{\beta \cdot a} y \cdot a \xrightarrow{\alpha} z.$$

It seems natural to define the fiber of a map (a, α) to be the object a and for

$$z \xrightarrow{(a, \alpha)} y \xrightarrow{(b, \beta)} x,$$

the induced fiber map $(a, \alpha)_{(b, \beta)}$ to be $(a, \mathbb{1}_{b \cdot a})$. This way we *almost* get a unary operadic category. The problem is, that there are in general no local terminal objects. Moreover,

there is a conceptual problem with the fiber functor, since it discards the information of the morphism α . This was the motivating example for our work. We resolve the problems by introducing unary operadic 2-categories. When passing to the 2-categorical framework, instead of being terminal, the monoidal unit $e \in \mathcal{M}^\Delta$ has the property, that for any other object x of \mathcal{M} , the category $\mathcal{M}^\Delta(x, e)$ has a terminal object, namely the map $(x, \mathbb{1}_x)$. More generally, the maps $(a, \mathbb{1}_{b \cdot a})$ will play an important rôle in the theory. In machine learning the category \mathcal{M}^Δ is known as the ‘Para’ construction of a monoidal category [FST21, G19, C&al22].

1.7. DEFINITION. Let $p: \mathbb{P} \rightarrow \mathbb{O}$ be an operadic functor between two unary operadic categories. It is a discrete operadic fibration, if p induces an epimorphism $\pi_0(\mathbb{P}) \rightarrow \pi_0(\mathbb{O})$ on connected components, and for any morphism $f: y \rightarrow x$ in \mathbb{O} with fiber r , and any two objects R, X of \mathbb{P} , such that $p(X) = x$ and $p(R) = r$, there exists a unique object Y and a unique morphism $F: Y \rightarrow X$ in \mathbb{P} with fiber R , such that $p(F) = f$.

1.8. DEFINITION. Let \mathbb{O} be a unary operadic category and \mathcal{P} an \mathbb{O} -operad in Set. Its operadic Grothendieck construction is a unary operadic category $\int_{\mathbb{O}} \mathcal{P}$ whose objects are pairs (x, a) , where x is an object of \mathbb{O} and $a \in \mathcal{P}_x$. A morphisms $(x', a') \rightarrow (x'', a'')$ of $\int_{\mathbb{O}} \mathcal{P}$ is a pair (f, c) of a map $f: x' \rightarrow x''$ and element $c \in \mathcal{P}_{\text{fib}(f)}$, such that $a' = a'' \cdot_f c$.

The operadic structure of $\int_{\mathbb{O}} \mathcal{P}$ is described in more detail in [BM24]. We recall, that the projection $\pi_{\mathcal{P}}: \int_{\mathbb{O}} \mathcal{P} \rightarrow \mathbb{O}$, sending a pair (x, a) to x , is a discrete operadic fibration.

1.9. PROPOSITION. [BM24, Proposition 35] The assignment $\mathcal{P} \mapsto \pi_{\mathcal{P}}$ is part of an equivalence

$$\mathbb{O}\text{-oper}(Set) \simeq \text{DoFib}(\mathbb{O})$$

between the category of set-valued \mathbb{O} -operads and discrete operadic fibrations over \mathbb{O} .

1.10. EXAMPLE. For a discrete monoidal category \mathcal{M} (equivalently \mathcal{M} is a monoid in Set), the only maps of the category \mathcal{M}^Δ of Example 1.6 are $b \cdot a \xrightarrow{(a, \mathbb{1}_{b \cdot a})} b$, and there is an isomorphism $\mathcal{M}^\Delta \cong \int_{\mathbb{O}} \mathcal{M}$, where \mathcal{M} is regarded as an \mathbb{O} -operad in Set. Hence the unique map $\mathcal{M}^\Delta \rightarrow \mathbb{O}$ is a discrete operadic fibration.

We close this section by recalling a characterisation of unary operadic categories via simplicial sets and décalage construction.

1.11. DEFINITION. An upper décalage $\text{dec}_{\mathbb{T}} X$ of simplicial set X is a simplicial set

$$(\text{dec}_{\mathbb{T}} X)_n = X_{n+1},$$

without the top face and degeneracy maps of X , i.e. $d_i^{\text{dec}} = d_i$ and $s_j^{\text{dec}} = s_j$.

Last, recall that for every category \mathcal{C} , its nerve \mathcal{NC} is a simplicial set of sequences of composable morphisms. The assignment \mathcal{N} is a fully faithful functor $\text{Cat} \rightarrow \text{sSet}$ and the simplicial set \mathcal{NC} is 2-coskeletal. This means that any element of $\mathcal{NC}_{k>2}$ can be

expressed as a tuple of compatible elements $t_1, \dots, t_k \in \mathcal{NC}_2$, and thus the simplicial set \mathcal{NC} is determined by the data

$$\begin{array}{ccccc} & & \xleftarrow{\quad d_2 \quad} & & \\ & \mathcal{NC}_0 & \xleftarrow{\quad s_0 \rightarrow \quad} & \mathcal{NC}_1 & \xleftarrow{\quad d_1 \quad} \xleftarrow{\quad s_1 \rightarrow \quad} \mathcal{NC}_2 \\ & \xleftarrow{\quad d_0 \quad} & & \xleftarrow{\quad s_0 \rightarrow \quad} & \xleftarrow{\quad d_0 \quad} \end{array}$$

1.12. PROPOSITION. [GKW21, Section 4] *A unary operadic category is equivalently a pair $\mathbb{O} = (X, \mathcal{C})$ of a simplicial set X and a category \mathcal{C} , such that*

$$\text{dec}_{\mathbb{T}} X \cong \mathcal{NC}.$$

2. The Nerve and Décalage for 2-categories

 E recall some standard 2-categorical constructions. By a 2-category we always mean a strict 2-category, i.e. a structure with objects (0-cells), morphisms (1-cells), and morphisms between morphisms (2-cells), with all possible compositions being strictly associative and unital. From now on Cat denotes the 1-category of 2-categories. It is a monoidal category with the cartesian product. When needed, we regard $(\text{Cat}, \times, 1)$ as an equivalent *strict* monoidal category. In general, we wish to ignore the coherence isomorphisms, as well as all size issues.

The 2-categorical nerve was explicitly introduced by J. Duskin in [D01]. It agrees with the classical nerve on 1-categories, regarded as 2-categories with only identity 2-cells, hence we keep the notation \mathcal{N} from the previous section.

2.1. DEFINITION. *For a 2-category \mathcal{C} , its nerve \mathcal{NC} is a simplicial set defined as*

$$\mathcal{NC}_n = \text{Cat}_{\text{NLax}}(n, \mathcal{C}),$$

where n is the ordinal $n = [0 < 1 < \dots < n]$, viewed as a 2-category, and Cat_{NLax} is the category of 2-categories and lax functors, which strictly preserve identities.

This gives a fully faithful functor $\mathcal{N}: \text{Cat} \rightarrow \text{sSet}$, whose image consists of 3-coskeletal simplicial sets. A general n -simplex σ of \mathcal{C} is a collection of objects x_0, \dots, x_n of \mathcal{C} , maps $f_{ij}: x_i \rightarrow x_j$, for $0 \leq i < j \leq n$, and lax triangles

$$f_{jk} \circ f_{ij} \xrightarrow{\alpha_{ijk}} f_{ik}, \text{ for } 0 \leq i < j < k \leq n,$$

such that the following diagram commutes for all $0 \leq i < j < k < l \leq n$.

$$\begin{array}{ccc} f_{kl} \circ f_{jk} \circ f_{ij} & \xrightarrow{\alpha_{jkl} \circ f_{ij}} & f_{jl} \circ f_{ij} \\ \Downarrow f_{kl} \circ \alpha_{ijk} & & \Downarrow \alpha_{ijl} \\ f_{kl} \circ f_{ik} & \xrightarrow{\alpha_{ikl}} & f_{il} \end{array}$$

Alternatively the nerve can be characterised by strict 2-functors from path 2-categories (cf. [Ker, Tag 00B9]). An n -simplex σ in \mathcal{C} is thus given by a collection of objects x_0, \dots, x_n of \mathcal{C} , maps $f_S: x_i \rightarrow x_j$ for every path $S = [i = p_0 < \dots < p_k = j]$, and 2-cells $\alpha_{ST}: f_S \Rightarrow f_T$ for $T \subseteq S$, an endpoint preserving (strict) inclusion. It is required that $f_{S''} \circ f_{S'} = f_{S' \cup S''}$, whenever the paths S' and S'' can be concatenated, and that $\alpha_{ST} \circ \alpha_{RS} = \alpha_{TR}$ for $T \subseteq S \subseteq R$. The first few terms of \mathcal{NC} are the following.

- \mathcal{NC}_0 is the set of objects of \mathcal{C} ,
- \mathcal{NC}_1 is the set of maps $x_0 \xrightarrow{f_{01}} x_1$ of \mathcal{C} ,
- \mathcal{NC}_2 is the set of lax triangles

$$\begin{array}{ccc} x_0 & \xrightarrow{f_{01}} & x_1 \\ & \searrow \alpha_{012} & \swarrow \\ & x_2 & \end{array}$$

in \mathcal{C} , where $f_{12} \circ f_{01} \xrightarrow{\alpha_{012}} f_{02}$ is a 2-cell of \mathcal{C} ,

- \mathcal{NC}_3 is a set of 3-simplices in \mathcal{C} , that is, quadruples $\sigma = (\alpha_{123}, \alpha_{023}, \alpha_{013}, \alpha_{012})$ of 2-cells $f_{jk} \circ f_{ij} \xrightarrow{\alpha_{ijk}} f_{ik}$, such that $\alpha_{013} \circ (\alpha_{123} \square f_{01}) = \alpha_{023} \circ (f_{23} \square \alpha_{012})$ where ‘ \square ’ is the horizontal composition of \mathcal{C} .

One can interpret σ as a filled 3-simplex obtained by gluing two squares along their boundaries, whose interiors equal:

$$\begin{array}{ccc} x_0 & \xrightarrow{f_{01}} & x_1 \\ f_{03} \downarrow & \nearrow \alpha_{013} & \downarrow f_{12} \\ x_3 & \xleftarrow{f_{23}} & x_2 \end{array} = \begin{array}{ccc} x_0 & \xrightarrow{f_{01}} & x_1 \\ f_{03} \downarrow & \searrow \alpha_{012} & \downarrow f_{12} \\ x_3 & \xleftarrow{f_{23}} & x_2 \end{array}$$

The face maps are clear from the graphical description, d_i deletes the vertex x_i . For a 3-simplex this means $\sigma = (d_0\sigma, d_1\sigma, d_2\sigma, d_3\sigma)$. The degeneracy maps are:

- for an object x in \mathcal{C} , $s_0x = \mathbb{1}_x$,
- for a map $f: x \rightarrow y$ in \mathcal{C} , $s_0f = (f \circ \mathbb{1}_x \xrightarrow{\mathbb{1}_f} f)$, and $s_1f = (\mathbb{1}_y \circ f \xrightarrow{\mathbb{1}_f} f)$,
- for a lax triangle $f_{12} \circ f_{01} \xrightarrow{\alpha_{012}} f_{02}$, of \mathcal{C} ,

$$s_0\alpha = (\alpha, \alpha, \mathbb{1}, \mathbb{1}), \quad s_1\alpha = (\mathbb{1}, \alpha, \alpha, \mathbb{1}), \quad s_2\alpha = (\mathbb{1}, \mathbb{1}, \alpha, \alpha).$$

Next, we study how the upper décalage (Definition 1.11) interacts with the 2-categorical nerve and we introduce a corresponding construction ‘ \mathcal{D} ’ for 2-categories.

2.2. DEFINITION. Let \mathcal{C} be a 2-category and x an object of \mathcal{C} . The lax slice \mathcal{C}/x is the 2-category with:

- Objects are maps $y \xrightarrow{f} x$ of \mathcal{C} with codomain x .
- For objects $z \xrightarrow{h} x$ and $y \xrightarrow{g} x$, a map $h \rightarrow g$ of \mathcal{C}/x is a pair (f, α) , where $z \xrightarrow{f} y$ is a map of \mathcal{C} and $g \circ f \xrightarrow{\alpha} h$ is a 2-cell of \mathcal{C} . The pair (f, α) is drawn as:

$$\begin{array}{ccc} z & \xrightarrow{f} & y \\ & \searrow h \quad \alpha \quad \swarrow g & \\ & x & \end{array}$$

- A 2-cell γ of \mathcal{C}/x ,

$$\begin{array}{ccc} h & \xrightarrow{(f', \alpha')} & g \\ & \Downarrow \gamma & \\ & \xleftarrow{(f'', \alpha'')} & \end{array}$$

is a 2-cell $f' \xrightarrow{\gamma} f''$ of \mathcal{C} , such that $\beta \circ (\mathbb{1}_g \square \gamma) = \alpha$.

More details can be found for instance in [JY20, Definition 7.1], but note the opposite orientation of the triangle interior.

2.3. DEFINITION. For a 2-category \mathcal{C} we write

$$\mathcal{DC} := \coprod_{x \in \mathcal{C}_0} \mathcal{C}/x.$$

2.4. EXAMPLE. Let \mathcal{M} be a strict monoidal category, \mathcal{BM} its associated 1-object 2-category and let $U\mathcal{C}$ denote the 1-category obtained from a 2-category \mathcal{C} by forgetting the 2-cells. Recall the category \mathcal{M}^Δ of Example 1.6. We have

$$\mathcal{M}^\Delta \cong U\mathcal{DBM}.$$

2.5. PROPOSITION. For a 2-category \mathcal{C} , $\text{dec}_\top \mathcal{NC} \cong \mathcal{ND}\mathcal{C}$.

PROOF. Since the simplicial set $\mathcal{ND}\mathcal{C}$ is 3-coskeletal, it is enough to verify bijections $\mathcal{ND}\mathcal{C}_k \cong \mathcal{NC}_{k+1}$ for $k \leq 3$. We comment only on the case $k = 3$.

First, a 3-simplex σ of \mathcal{DC} is determined by objects f_0, \dots, f_3 of \mathcal{DC} , that is, by maps $x_i \xrightarrow{f_i} y_i$ of \mathcal{C} . Further, for every path $S = [i = p_0 < \dots < p_k = j]$, there is a map $f_S = (t_S, \alpha_S) : f_i \rightarrow f_j$ in \mathcal{DC} , that is a lax triangle

$$\begin{array}{ccc} x_i & \xrightarrow{t_S} & x_j, \\ & \searrow f_i \quad \alpha_S \quad \swarrow f_j & \\ & y_i = y_j & \end{array}$$

and for every subsequence $T \subseteq S$ there is a 2-cell $\alpha_{ST}: f_S \Rightarrow f_T$ in \mathcal{DC} , that is, a 2-cell $\alpha_{ST}: t_S \Rightarrow t_T$ in \mathcal{C} , such that $\alpha_T \circ (f_j \cdot \alpha_{ST}) = \alpha_S$. On the other hand, a 4-simplex δ of \mathcal{C} is given by objects z_0, \dots, z_4 , maps $g_S: z_i \rightarrow z_j$, and 2-cells $\beta_{ST}: g_S \rightarrow g_T$ of \mathcal{C} .

Given δ we construct a 3-simplex σ of \mathcal{DC} as follows. Define $f_i := g_{[i < 4]}: z_i \rightarrow z_4$. Let $S = [i = p_0 < \dots < p_k = j]$ and denote $\bar{S} := S \cup [j < 4]$. Since $g_{[j < 4]} \circ g_S = g_{\bar{S}}$, we define $f_S := \beta_{[i < 4]\bar{S}}: g_{[j < 4]} \circ g_S \rightarrow g_{[i < 4]}$ and $\alpha_{ST} := \beta_{ST}$.

Conversely, given a 3-simplex σ of \mathcal{DC} , we construct δ as follows. For $i \leq 3$ define $z_i := x_i$ and $z_4 := y_0 = \dots = y_4$. Further, $g_S := t_S$ and $\beta_{ST} := \alpha_{ST}$. These two constructions are inverse and the bijection preserves the simplicial structure. ■

3. Unary Operadic 2-categories



THE following definition was motivated by the characterisation of unary operadic categories of [GKW21], which we recalled in Proposition 1.12. We adapt the characterisation to the 2-categorical setting and we use it as a starting point for our theory.

3.1. DEFINITION. A *unary operadic 2-category* is a pair $\mathbb{O} = (X, \mathcal{C})$ of a 2-category \mathcal{C} and a simplicial set X , such that the upper décalage of X is the nerve of \mathcal{C} :

$$\text{dec}_\top X \cong \mathcal{NC}.$$

An operadic functor between two unary operadic 2-categories is a morphism of their underlying simplicial sets. We denote the category of unary operadic 2-categories by uOpCat .

We unpack the definition in terms of the underlying 2-category \mathcal{C} , the set X_0 , and additional top face and degeneracy maps on \mathcal{NC} , which we denote by φ and u , respectively.

3.2. PROPOSITION. A *unary operadic 2-category* is a 2-category \mathcal{C} together with the following data:

1. for every object a of \mathcal{C} , there is $\varphi(a) \in \pi_0(\mathcal{C})$,
2. for every arrow $x \xrightarrow{q} y$ in \mathcal{C} , $\varphi(q)$ is an object of \mathcal{C} ,
3. for every lax triangle $g \circ f \xrightarrow{\alpha} h$ in \mathcal{C} , $\varphi(h) \xrightarrow{\varphi(\alpha)} \varphi(g)$ is a map in \mathcal{C} ,
4. for a 3-simplex σ_{0123} of \mathcal{C} , $\varphi(\alpha_{123}) \circ \varphi(\alpha_{013}) \xrightarrow{\varphi(\sigma_{0123})} \varphi(\alpha_{023})$ is a lax triangle in \mathcal{C} ,
5. for every $c \in \pi_0(\mathcal{C})$, u_c is an object of \mathcal{C} ,
6. for every object $a \in \mathcal{C}$, $a \xrightarrow{u_a} u_{\pi(a)}$ is a map in \mathcal{C} ,
7. for every map $x \xrightarrow{q} y$ of \mathcal{C} , there is a 2-cell $u_y \circ q \xrightarrow{u_q} u_x$, and

8. for every lax triangle $g \circ f \xrightarrow{\alpha} h$, there is a 3-simplex $u_\alpha = (u_g, u_h, u_f, \alpha)$ in \mathcal{C} .

This data are subject to the following equations. For every c, a, q, α , and σ_{0123} as above,

9. $\pi(\varphi(q)) = \varphi(x)$,

10. $\pi(u_c) = c$,

11. $\varphi(\mathbb{1}_a) = u_{\varphi(a)}$,

12. $\varphi(\mathbb{1}_y \circ q \xrightarrow{\mathbb{1}_q} q) = u_{\varphi(q)}$ and $\varphi(q \circ \mathbb{1}_x \xrightarrow{\mathbb{1}_q} q) = \mathbb{1}_{\varphi(q)}$,

13. $\varphi(\mathbb{1}, \mathbb{1}, \alpha, \alpha) = u_{\varphi(\alpha)}$, $\varphi(\mathbb{1}, \alpha, \alpha, \mathbb{1}) = \mathbb{1}_{\varphi(g)} \circ \varphi(\alpha) \xrightarrow{\mathbb{1}_{\varphi(\alpha)}} \varphi(\alpha)$, and $\varphi(\alpha, \alpha, \mathbb{1}, \mathbb{1}) = \varphi(\alpha) \circ \mathbb{1}_{\varphi(h)} \xrightarrow{\mathbb{1}_{\varphi(\alpha)}} \varphi(\alpha)$,

14. $u_{u_c} = \mathbb{1}_{u_c}$, $u_{u_a} = \mathbb{1}_{u_a}$, and $u_{u_q} = \mathbb{1}_{u_q}$,

15. $\varphi \circ u = \mathbb{1}$, more precisely $\varphi(u_c) = c$, $\varphi(u_a) = a$, $\varphi(u_q) = q$, and $\varphi(u_\alpha) = \alpha$,

16. $\varphi(\varphi(\sigma_{0123})) = \varphi(\alpha_{012})$, $\varphi(\varphi(\alpha)) = \varphi(f)$, and $\varphi(\varphi(q)) = \varphi(x)$,

17. $u_{\mathbb{1}_x} = \mathbb{1}_{u_x}$, $u_{s_0 q} = s_0 u_q$, and $u_{s_1 q} = s_1 u_q$.

3.3. REMARK. These conditions are not minimal. For example the condition

$$\varphi(\varphi(q: x \rightarrow y)) = \varphi(x)$$

can be derived from $\varphi(\varphi(\alpha)) = \varphi(f)$ with $\alpha = s_0 q$ as follows:

$$\varphi \varphi q = \pi u \varphi \varphi q = \pi \varphi s_0 \varphi q = \pi \varphi \varphi s_0 q = \pi \varphi d_2 s_0 q = \pi \varphi s_0 d_1 q = \pi u \varphi d_1 q = \varphi d_1 q = \varphi x.$$

Before we prove the proposition, we make two observations. Let $\mathbb{O} = (\mathcal{C}, X)$ be a unary operadic 2-category.

3.4. LEMMA. *There is a bijection $X_0 \cong \pi_0(\mathcal{C})$, such that the following diagram commutes.*

$$\begin{array}{ccc} X_0 & \xrightarrow{\cong} & \pi_0(\mathcal{C}) \\ d_0 \uparrow & & \uparrow \pi \\ X_1 & \xrightarrow{\cong} & \mathcal{C}_0 \end{array}$$

PROOF. This follows from the fact that for any simplicial set X , the set X_0 is a split coequalizer of $d_0, d_1: X_2 \rightrightarrows X_1$ and $\pi_0(\mathcal{C})$ is a coequalizer of $X_2 = \mathcal{C}_1 \rightrightarrows \mathcal{C}_0 = X_1$. ■

3.5. LEMMA. *The simplicial set X is 4-coskeletal.*

PROOF. This follows from the defining isomorphism $\text{dec}_T X \cong \mathcal{NC}$ and the fact that the nerve \mathcal{NC} is 3-coskeletal. ■

PROOF OF PROPOSITION 3.2. It follows from the two lemmas above that a unary operadic 2-category is determined by a diagram

$$\begin{array}{ccccccc} \pi_0\mathcal{C} & \xleftarrow{\varphi} & \mathcal{C}_0 & \xleftarrow{\varphi} & \mathcal{C}_1 & \xleftarrow{\varphi} & \mathcal{NC}_2 \\ \xleftarrow{\pi} & \xleftarrow{u} & \xleftarrow{u} & \xleftarrow{u} & \xleftarrow{u} & \xleftarrow{u} & \xleftarrow{u} \\ & \xleftarrow{d_0} & & \xleftarrow{d_0} & & \xleftarrow{d_0} & \end{array}$$

and simplicial identities up to $X_4 = (\mathcal{NC})_3$. Most of the identities are satisfied since \mathcal{C} is a 2-category. We translate those which are not automatic:

$\mathcal{C}_1 \rightarrow \pi_0\mathcal{C}$	$\mathcal{NC}_2 \rightarrow \mathcal{C}_0$	$\mathcal{NC}_3 \rightarrow \mathcal{C}_1$	$\mathcal{C}_1 \rightarrow \mathcal{NC}_3$	$\mathcal{C}_0 \rightarrow \mathcal{NC}_2$
$\pi\varphi = \varphi d_0 \quad (9)$	$d_0\varphi = \varphi d_0 \quad (4)$ $d_1\varphi = \varphi d_1 \quad (3)$ $\varphi\varphi = \varphi d_2 \quad (16)$	$d_1\varphi = \varphi d_1 \quad (4)$ $d_2\varphi = \varphi d_2 \quad (4)$ $\varphi\varphi = \varphi d_3 \quad (16)$	$us_0 = s_0u \quad (17)$ $us_1 = s_1u \quad (17)$ $uu = s_2u \quad (14)$	$us_0 = s_0u \quad (17)$ $uu = s_1u \quad (14)$
$\pi_0\mathcal{C} \rightarrow \mathcal{C}_1$	$\pi_0\mathcal{C} \rightarrow \pi_0\mathcal{C}$	$\mathcal{C}_0 \rightarrow \mathcal{C}_0$	$\mathcal{C}_1 \rightarrow \mathcal{C}_1$	$\mathcal{NC}_2 \rightarrow \mathcal{NC}_2$
$uu = s_0u \quad (14)$	$\pi u = 1 \quad (10)$ $\varphi u = 1 \quad (15)$	$\varphi s_0 = u\varphi \quad (11)$ $d_0u = u\pi \quad (6)$ $d_1u = 1 \quad (6)$ $\varphi u = 1 \quad (15)$	$\varphi s_0 = s_0\varphi \quad (12)$ $\varphi s_1 = s_1\varphi \quad (12)$ $\varphi s_2 = u\varphi \quad (13)$ $d_0u = ud_0 \quad (7)$ $d_1u = ud_1 \quad (7)$ $d_2u = 1 \quad (7)$ $\varphi u = 1 \quad (15)$	$\varphi s_0 = s_0\varphi \quad (13)$ $\varphi s_1 = s_1\varphi \quad (13)$ $\varphi s_2 = u\varphi \quad (13)$ $d_0u = ud_0 \quad (8)$ $d_1u = ud_1 \quad (8)$ $d_2u = ud_2 \quad (8)$ $d_3u = 1 \quad (8)$ $\varphi u = 1 \quad (15)$

These equations compare directly (cf. numbers in brackets) to the data and conditions of Proposition 3.2, which finishes the proof. ■

Our next goal is to reformulate Definition 3.1, so it resembles the original definition of unary operadic categories of [BM24], see Definition 1.1. We find analogues to fiber functors and local terminal objects.

3.6. DEFINITION. An object v in a 2-category \mathcal{C} is lali-terminal, if for every object x of \mathcal{C} , the category $\mathcal{C}(x, v)$ has a terminal object. We require that the terminal object of $\mathcal{C}(v, v)$ is the identity on v . An object v is local lali-terminal if it is lali-terminal in its component.

The prefix *lali* stands for ‘left adjoint left inverse’. This terminology appears in [Š24], however the condition on the terminal object of $\mathcal{C}(v, v)$ is not required there.

3.7. LEMMA. An object v in a 2-category \mathcal{C} is lali-terminal if and only if for every object $x \in \mathcal{C}$, there is a chosen map $x \xrightarrow{v_x} v$, for every map $x \xrightarrow{f} y$ of \mathcal{C} , there is a chosen 2-cell $v_y \circ f \xrightarrow{v_f} v_x$, and for every lax triangle $g \circ f \xrightarrow{\alpha} h$, there is a 3-simplex $v_\alpha = (v_g, v_h, v_f, \alpha)$ in \mathcal{C} , such that $v_v = \mathbb{1}_v$ and $v_{v_x} = \mathbb{1}_{v_x}$.

PROOF. Let v be lali-terminal in \mathcal{C} . For an object x , the map v_x is the terminal object of $\mathcal{C}(x, v)$ and for $f: x \rightarrow y$, the 2-cell v_f is the terminal map $v_y \circ f \Rightarrow v_x$ of $\mathcal{C}(x, v)$. For a 2-cell $g \circ f \xrightarrow{\alpha} h$, there are two 2-cells $v_z \circ g \circ f \Rightarrow v_x$, namely $v_h \circ (v_x \cdot \alpha)$ and $v_f \circ (v_g \cdot f)$. Since v_x is terminal, the two composites equal, hence they form a 3-simplex $v_\alpha = (v_g, v_h, v_f, \alpha)$. By definition, $v_v = \mathbb{1}_v$ and it follows that $v_{v_x} = \mathbb{1}_{v_x}$.

Conversely, we will show that the choice of v_x, v_f , and v_α , with $v_v = \mathbb{1}_v$ and $v_{v_x} = \mathbb{1}_{v_x}$, makes v lali-terminal. Let there be two 2-cells $\alpha, \beta: f \Rightarrow v_x$ in $\mathcal{C}(x, v)$. We interpret them as lax triangles $f \circ \mathbb{1}_x \Rightarrow v_x$. We have their 3-simplices v_α and v_β , which give

$$\alpha = v_{v_x} \circ (v_v \cdot \alpha) = v_{\mathbb{1}_x} \circ (v_f \cdot \mathbb{1}_x) = v_{v_x} \circ (v_v \cdot \beta) = \beta.$$

■

3.8. PROPOSITION. *A unary operadic 2-category is a 2-category \mathcal{C} together with a normalised lax functor $\mathcal{DC} \xrightarrow{\varphi} \mathcal{C}$ and a set \mathcal{U} of chosen local lali-terminal objects u_c in each connected component c of \mathcal{C} , such that the following conditions hold.*

- For any object x of \mathcal{C} , $\varphi(\mathbb{1}_x) \in \mathcal{U}$,
- for a map $f: x \rightarrow y$ of \mathcal{C} , $\varphi(f, \mathbb{1}_f) = u_{\varphi(f)}$, where $(f, \mathbb{1}_f)$ is the 2-cell $\mathbb{1}_y \circ f \xrightarrow{\mathbb{1}_f} f$,
- for $u_c \in \mathcal{U}$, $\varphi_{u_c}: \mathcal{C}/u_c \rightarrow \mathcal{C}$ is the domain functor,
- for a 3-simplex σ_{0123} in \mathcal{C} , $\varphi(\varphi(\sigma_{0123})) = \varphi(\alpha_{012})$.

PROOF. Elementary, using the characterisation of Proposition 3.2 and Lemma 3.7. ■

3.9. EXAMPLE. Every unital unary operadic category of [BM24] is a unary operadic 2-category. We shall call them unary operadic 1-categories.

3.10. EXAMPLE. By Proposition 2.5, for any 2-category \mathcal{C} , the pair $\mathbb{C} = (\mathcal{NC}, \mathcal{DC})$ is a unary operadic 2-category. In particular, any strict monoidal category \mathcal{V} can be viewed as a one-object 2-category \mathcal{BV} . Hence, $(\mathcal{NBV}, \mathcal{DBV})$ is a unary operadic 2-category which we denote by \mathcal{V}^Δ . This is in a slight conflict with the notation of Example 1.6, however we stick to this new meaning. Let us make the structure explicit. The 2-category \mathcal{DBV} consists of the following data.

- Objects are the objects of \mathcal{V} .
- Morphisms $x \rightarrow y$ are pairs (a, f) such that $y \otimes a \xrightarrow{f} x$ is a map in \mathcal{V} .
- 2-cells $(a, f) \xrightarrow{\delta} (b, g)$ are maps $\delta: a \rightarrow b$ of \mathcal{V} , such that the triangle

$$\begin{array}{ccc} y \otimes a & \xrightarrow{y \otimes \delta} & y \otimes b \\ & \searrow f & \swarrow g \\ & x & \end{array}$$

commutes.

The additional structure maps u and φ on $\mathcal{ND}\mathcal{BV}$ are obtained from the simplicial set \mathcal{NBV} and the isomorphism $\text{dec}_{\top} \mathcal{NBV} \cong \mathcal{ND}\mathcal{BV}$. Namely, \mathcal{NBV} consists of:

- $(\mathcal{NBV})_0 = \{\cdot\}$,
- $(\mathcal{NBV})_1$ is the set of segments

$$\cdot \xrightarrow{x} \cdot, x \in \mathcal{V},$$

- $(\mathcal{NBV})_2$ is the set of triangles

$$\begin{array}{ccc} \cdot & \xrightarrow{a_{01}} & \cdot \\ & \searrow f_{01} \quad \swarrow & \\ x_0 & & x_1 \\ & \cdot & \end{array}$$

with a map $x_1 \otimes a_{01} \xrightarrow{f_{01}} x_0$ in \mathcal{V} ,

- $(\mathcal{NBV})_3$ is the set of 3-simplices

$$\begin{array}{ccc} \cdot & \xrightarrow{a_{01}} & \cdot \\ x_0 \downarrow & \nearrow f_{01} \quad \downarrow a_{12} & \downarrow \\ \cdot & \xrightarrow{a_{02}} & \cdot \\ x_0 \downarrow & \nearrow f_{02} \quad \downarrow a_{12} & \downarrow \\ & \cdot & \end{array} = \begin{array}{ccc} \cdot & \xrightarrow{a_{01}} & \cdot \\ x_0 \downarrow & \nearrow a_{012} \quad \downarrow a_{12} & \downarrow \\ \cdot & \xrightarrow{a_{02}} & \cdot \\ x_0 \downarrow & \nearrow f_{02} \quad \downarrow a_{12} & \downarrow \\ & \cdot & \end{array},$$

- (we do not specify $(\mathcal{NBV})_4$, instead we describe $(\mathcal{ND}\mathcal{BV})_3$ directly below.)

The simplicial set $\mathcal{ND}\mathcal{BV}$ consists of:

- $(\mathcal{ND}\mathcal{BV})_0 = \mathcal{V}_0$,
- $(\mathcal{ND}\mathcal{BV})_1$ is the set of segments

$$x_0 \xrightarrow{(a_{01}, f_{01})} x_1,$$

where $x_1 \otimes a_{01} \xrightarrow{f_{01}} x_0$ is a map in \mathcal{V} ,

- $(\mathcal{ND}\mathcal{BV})_2$ is the set of triangles

$$\begin{array}{ccc} x_0 & \xrightarrow{(a_{01}, f_{01})} & x_1 \\ & \searrow \alpha_{012} \quad \swarrow & \\ (a_{02}, f_{02}) & \searrow & \swarrow (a_{12}, f_{12}) \\ x_2 & & x_2 \end{array}$$

with a map $a_{12} \otimes a_{01} \xrightarrow{\alpha_{012}} a_{02}$ in \mathcal{V} , such that the diagram

$$\begin{array}{ccc} x_2 \otimes a_{12} \otimes a_{01} & \xrightarrow{x_2 \otimes \alpha_{012}} & x_2 \otimes a_{02} \\ f_{12} \otimes a_{01} \downarrow & & \downarrow f_{02} \\ x_1 \otimes a_{01} & \xrightarrow{f_{01}} & x_0 \end{array}$$

commutes. We denote the triangle by $\Delta_{012} = [f_{12}, f_{02}, f_{01}, \alpha_{012}]$, the square brackets indicate the commuting square.

- $(\mathcal{ND}\mathcal{BV})_3$ is the set of quadruples $\sigma_{0123} = (\Delta_{123}, \Delta_{023}, \Delta_{013}, \Delta_{012})$ of triangles, that is, a collection of maps

$$x_q \otimes a_{pq} \xrightarrow{f_{pq}} x_p \text{ and } a_{jk} \otimes a_{ij} \xrightarrow{\alpha_{ijk}} a_{ik}$$

in \mathcal{V} , such that the following cube commutes. The labeling of each edge is determined by its source and target.

$$\begin{array}{ccccc} x_3 \otimes a_{23} \otimes a_{12} \otimes a_{01} & \longrightarrow & x_2 \otimes a_{12} \otimes a_{01} & & \\ \downarrow & \searrow & \downarrow & \searrow & \\ & x_3 \otimes a_{13} \otimes a_{01} & \longrightarrow & x_1 \otimes a_{01} & \\ x_3 \otimes a_{23} \otimes a_{02} & \dashrightarrow & x_2 \otimes a_{02} & \dashrightarrow & x_0 \\ \downarrow & \searrow & \downarrow & \searrow & \downarrow \\ & x_3 \otimes a_{03} & \longrightarrow & x_0 & \end{array}$$

Note that \mathcal{DBV} has only one connected component c_I , since for any $x \in \mathcal{V}$ there is a map $x \xrightarrow{(x, \mathbb{1}_x)} I$ into the monoidal unit I of \mathcal{V} . Hence, the nerve $\mathcal{ND}\mathcal{BV}$ is indeed extended by the set $\pi_0(\mathcal{ND}\mathcal{BV}) = \{c_I\} \cong (\mathcal{NBV})_0$. The maps u and φ on $\mathcal{ND}\mathcal{BV}$ are:

- on connected components: $u_{c_I} = I$,
- on 0-simplices: $u_x = (x, \mathbb{1}_x)$ and $\varphi(x) = c_I$,
- on 1-simplices: $u_{(a_{01}, f_{01})} = [f_{01}, f_{01}, \mathbb{1}_{x_0}, \mathbb{1}_{x_1}]$ and $\varphi(a_{01}, f_{01}) = a_{01}$,
- on 2-simplices: $u_{\Delta_{012}} = (\Delta_{012}, u_{(a_{01}, f_{01})}, u_{(a_{02}, f_{02})}, u_{(a_{12}, f_{12})})$ and $\varphi(\Delta_{012}) = (a_{01}, \alpha_{012})$,
- on 3-simplices: $\varphi(\sigma_{0123}) = \Delta_{012}$.

The 2-category \mathcal{DBV} is a 2-categorical version of the Para-construction, mentioned in Example 1.6, which might be of interest. For an ordinary monoid M in Set, viewed as a discrete monoidal category, $\mathcal{ND}\mathcal{BM}$ recovers the classical bar complex $(\mathcal{ND}\mathcal{BM})_n = M^{\times n}$.

4. Operads of Unary Operadic 2-Categories

 BEFORE we introduce \mathbb{O} -operads of an operadic 2-category \mathbb{O} , we define 1-connected \mathbb{O} -operads. That is, operads whose part \mathcal{P}_{u_c} over chosen local (lali-)terminal object u_c contains only the identity operation e_c . The reason is, that 1-connected operads can be viewed as generalised presheaves and this conceptual approach leads naturally to operadic Grothendieck construction for categorical \mathbb{O} -operads. We then distil an explicit definition which we use to describe operadic Grothendieck construction of a non-1-connected categorical \mathbb{O} -operad.

4.1. DEFINITION. *Let \mathcal{V} be a strict monoidal category and \mathcal{V}^Δ the unary operadic 2-category of Example 3.10. A 1-connected \mathbb{O} -operad \mathcal{P} with values in \mathcal{V} is an operadic functor $\mathcal{P}: \mathbb{O} \rightarrow \mathcal{V}^\Delta$.*

For simplicity let $\mathcal{V} = \text{Cat}$, the category of small categories (considered as an equivalent strict monoidal category) and let $\mathbb{O} = (X, \mathcal{C})$. A categorical \mathbb{O} -operad \mathcal{P} is thus a collection of categories \mathcal{P}_x , indexed by objects $x \in \mathcal{C}$, such that $\mathcal{P}_{u_c} = 1$, the terminal category, for every $c \in \pi_0(\mathcal{C})$. We denote the unique object of \mathcal{P}_{u_c} by e_c . The collection is further equipped with functors

$$\mathcal{P}_y \times \mathcal{P}_{\varphi(f)} \xrightarrow{\mathcal{P}_f} \mathcal{P}_x,$$

for every map $f: x \rightarrow y$ in C which satisfy the following associativity and unit laws. We use the notation $a \cdot_f b$ for $\mathcal{P}_f(a, b)$.

- For any lax triangle α in C ,

$$\begin{array}{ccc} x & \xrightarrow{f} & y, \\ & \searrow h \quad \swarrow \alpha & \\ & z & \end{array}$$

and objects $a \in \mathcal{P}_z, b \in \mathcal{P}_{\varphi(g)}, c \in \mathcal{P}_{\varphi(f)}$,

$$(a \cdot_g b) \cdot_f c = a \cdot_h (b \cdot_{\varphi(\alpha)} c).$$

It follows from the axioms of an operadic 2-category that $\varphi(f) = \varphi(\varphi(\alpha))$, hence the expression above is well defined.

- For any object $x \in C$ and $a \in \mathcal{P}_x$,

$$a \cdot_{\mathbb{1}_x} e_{\varphi(x)} = a,$$

$$e_{\pi(x)} \cdot_{u_x} a = a.$$

From the simplicial viewpoint, the operad multiplication has a more pleasing form:

$$\mathcal{P}_{d_0 f} \times \mathcal{P}_{d_2 f} \xrightarrow{\mathcal{P}_f} \mathcal{P}_{d_1 f},$$

and the laws can be written as

$$\mathcal{P}_{d_1\alpha}(\mathcal{P}_{d_3\alpha} \times \mathbb{1}) = \mathcal{P}_{d_2\alpha}(\mathbb{1} \times \mathcal{P}_{d_0\alpha})$$

$$\mathcal{P}_{s_0x}(-, e_{d_1x}) = \mathbb{1}_{\mathcal{P}_x} = \mathcal{P}_{s_1x}(e_{d_0x}, -).$$

The higher information of Definition 4.1 is redundant. This motivates the following definition where we weaken the 1-connected condition.

4.2. DEFINITION. A categorical \mathbb{O} -operad \mathcal{P} is a structure described under Definition 4.1 with the condition $\mathcal{P}_{u_c} = \{e_c\}$ replaced by a choice of maps $1 \xrightarrow{e_c} \mathcal{P}_{u_c}$ for every $c \in \pi_0(\mathcal{C})$, which pick the units.

4.3. EXAMPLE. For unary operadic 1-categories we recover the original definition of a unital \mathbb{O} -operad of [BM24].

4.4. EXAMPLE. For a 2-category \mathcal{C} , the \mathbb{C} -operads in \mathcal{V} of $\mathbb{C} = (\mathcal{NC}, \mathcal{DC})$ are lax 2-functors from \mathcal{C} to \mathcal{BV} . In particular, for a strict monoidal category \mathcal{M} , the \mathcal{M}^Δ -operads in \mathcal{V} are lax monoidal functors $\mathcal{M} \rightarrow \mathcal{V}$.

5. Operadic Grothendieck construction



OR a unary operadic 2-category \mathbb{O} and a 1-connected categorical \mathbb{O} -operad \mathcal{P} , we introduce its operadic Grothendieck construction, i.e. a unary operadic 2-category $\int_{\mathbb{O}} \mathcal{P}$ with a projection $\pi_{\mathcal{P}}: \int_{\mathbb{O}} \mathcal{P} \rightarrow \mathbb{O}$. We make the construction explicit. For non-1-connected operads the construction is obtained simply by weakening the 1-connected condition.

Observe that for any $\mathcal{C} \in \text{Cat}$, the lax coslice category \mathcal{C}/Cat is again monoidal. We keep our convention that lax means ‘from many to one’, hence morphisms $F \rightarrow G$ of \mathcal{C}/Cat are pairs (H, α) :

$$\begin{array}{ccc} & \mathcal{C} & \\ F \swarrow & \downarrow \alpha & \searrow G \\ \mathcal{D} & \xrightarrow{H} & \mathcal{E}, \end{array}$$

with $H \circ F \xrightarrow{\alpha} G$. The codomain projection $\pi: X/\text{Cat} \rightarrow \text{Cat}$ is a monoidal functor which induces an operadic functor $\pi^\Delta: (X/\text{Cat})^\Delta \rightarrow \text{Cat}^\Delta$ (for the notation cf. Example 3.10).

5.1. PROPOSITION. *The category uOpCat has pullbacks.*

PROOF. Let $p: \mathbb{O} \rightarrow \mathbb{P}$ and $q: \mathbb{Q} \rightarrow \mathbb{P}$ be two operadic functors, with $\mathbb{O} = (X_{\mathbb{O}}, \mathcal{C}_{\mathbb{O}})$, $\mathbb{P} = (X_{\mathbb{P}}, \mathcal{C}_{\mathbb{P}})$, and $\mathbb{Q} = (X_{\mathbb{Q}}, \mathcal{C}_{\mathbb{Q}})$. It follows from the properties of the nerve and décalage that the pullback of p and q is $\mathbb{O} \times_{\mathbb{P}} \mathbb{Q} = (X_{\mathbb{O}} \times_{X_{\mathbb{P}}} X_{\mathbb{Q}}, \mathcal{C}_{\mathbb{O}} \times_{\mathcal{C}_{\mathbb{P}}} \mathcal{C}_{\mathbb{Q}})$. ■

5.2. DEFINITION. Let $\mathbb{O} = (X, \mathcal{C})$ be a unary operadic 2-category and \mathcal{P} a categorical 1-connected \mathbb{O} -operad. The operadic Grothendieck construction $\int_{\mathbb{O}} \mathcal{P} = (X_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}})$ is a pullback of \mathcal{P} along $\pi^{\Delta}: (1/\text{Cat})^{\Delta} \rightarrow \text{Cat}^{\Delta}$, where 1 denotes the terminal category:

$$\begin{array}{ccc} \int_{\mathbb{O}} \mathcal{P} & \longrightarrow & (1/\text{Cat})^{\Delta} \\ \pi_{\mathcal{P}} \downarrow & \lrcorner & \downarrow \pi^{\Delta} \\ \mathbb{O} & \xrightarrow{\mathcal{P}} & \text{Cat}^{\Delta}. \end{array}$$

The assignment $\int_{\mathbb{O}}$ induces a fully faithful functor $\mathbb{O}\text{-oper}(\text{Cat}) \rightarrow \text{uOpCat}/\mathbb{O}$. Its codomain is a category of operadic functors into \mathbb{O} and morphisms are commutative triangles of operadic functors. We make the definition explicit in terms of the 2-category $\mathcal{C}_{\mathcal{P}}$ and additional maps φ and u on $\mathcal{NC}_{\mathcal{P}}$. Denote by ψ the fiber assignment of \mathbb{O} , by v the unit maps of \mathbb{O} , and by $e_c \in \mathcal{P}_{v_c}$ the units of the operad \mathcal{P} .

- The objects of $\mathcal{C}_{\mathcal{P}}$ are pairs $(A \in \mathcal{C}_0, a \in \mathcal{P}_A)$,
- 1-cells $(A, a) \rightarrow (B, b)$ are triples (f, p, α) , where $f: A \rightarrow B$ is a map in \mathcal{C} , $p \in \mathcal{P}_{\psi(f)}$, and $b \cdot_f p \xrightarrow{\alpha} a$ is a map in \mathcal{P}_A . The identity 1-cell of an object (A, a) is the triple $(\mathbb{1}_A, e_{\varphi(A)}, \mathbb{1}_a)$. The composite of 1-cells

$$(A, a) \xrightarrow{(f', p', \alpha')} (B, b) \xrightarrow{(f'', p'', \alpha'')} (C, c)$$

is defined by

$$(f'', p'', \alpha'') \circ (f', p', \alpha') = (f'' \circ f', p'' \cdot_{\psi(\mathbb{1}_{f'' \circ f'})} p', \alpha' \circ (\alpha'' \cdot_{f'} p')).$$

- a 2-cell

$$(A, a) \begin{array}{c} \xrightarrow{(f, p, \alpha)} \\ \Downarrow (\rho, \gamma) \\ \xrightarrow{(g, q, \beta)} \end{array} (B, b)$$

is a pair (ρ, γ) consisting of a 2-cell $\rho: f \Rightarrow g$ of \mathcal{C} and a map γ in $\mathcal{P}_{\psi(g)}$ which we further specify. Let us first define a map $\mathcal{P}_{\rho}: \mathcal{P}_{\psi(f)} \rightarrow \mathcal{P}_{\psi(g)}$. We consider ρ as a triangle $\bar{\rho}: f \circ \mathbb{1} \Rightarrow g$, whose fiber is $\psi(\bar{\rho}): \psi(g) \rightarrow \psi(f)$ and we have $\psi(\psi(\bar{\rho})) = \psi(\mathbb{1}) = v_c$ for some $c \in \pi_0(\mathcal{C})$. We define $\mathcal{P}_{\rho}(p) := p \cdot_{\psi(\bar{\rho})} e_c$ and require $\gamma: \mathcal{P}_{\rho}(p) \rightarrow q$ in $\mathcal{P}_{\psi(g)}$, such that following diagram commutes in \mathcal{P}_A .

$$\begin{array}{ccc} b \cdot_g \mathcal{P}_{\rho}(p) & \xrightarrow{=} & b \cdot_f p \\ b \cdot_g \gamma \downarrow & & \downarrow \alpha \\ b \cdot_g q & \xrightarrow{\beta} & a \end{array}$$

The augmented nerve $\mathcal{NC}_{\mathcal{P}}$ consists of:

- $(\mathcal{NC}_{\mathcal{P}})_{-1} = \pi_0(\mathcal{C}_{\mathcal{P}})$, the set connected components of $\mathcal{C}_{\mathcal{P}}$. This is in bijection with connected components of \mathcal{C} via the maps

$$(A, a) \xrightarrow{(u_A, a, \mathbb{1}_a)} (v_{\pi(A)}, e_{\pi(A)})$$

in $\mathcal{C}_{\mathcal{P}}$, and also in bijection with the set of units e_c of the operad \mathcal{P} . We shall use

$$(\mathcal{NC}_{\mathcal{P}})_{-1} = \pi_0(\mathcal{C}).$$

- $(\mathcal{NC}_{\mathcal{P}})_0 = (\mathcal{C}_{\mathcal{P}})_0$ consists of pairs $(A \in \mathcal{C}_0, a \in \mathcal{P}_A)$, interpreted as segments

$$\psi(A) \xrightarrow{(A, a)} \pi(A).$$

This gives $\varphi(A, a) := \psi(A)$.

- $(\mathcal{NC}_{\mathcal{P}})_1 = (\mathcal{C}_{\mathcal{P}})_1$ consists of maps (f, p, α) , with $A \xrightarrow{f} B$ and $b \cdot_f p \xrightarrow{\alpha} a$, interpreted as triangles

$$\begin{array}{ccc} \psi(A) & \xrightarrow{(\psi(f), p)} & \psi(B) \\ & \searrow^{(A, a)} & \swarrow^{(B, b)} \\ & (f, p, \alpha) & \\ \pi(A) = \pi(B) & & \end{array}$$

This gives $\varphi(f, p, \alpha) := (\psi(f), p)$.

- $(\mathcal{NC}_{\mathcal{P}})_2$ consists of lax triangles in $\mathcal{C}_{\mathcal{P}}$, that is 2-cells

$$(\rho, \gamma): (g, q, \beta) \circ (f, p, \alpha) \Rightarrow (h, r, \delta)$$

which means

- $\rho: (g \circ f) \Rightarrow h$ is a 2-cell in \mathcal{C} ,
- $\gamma: \mathcal{P}_\rho(q \cdot_{\psi(\mathbb{1}_{g \circ f})} p) \rightarrow r$ is a map in $\mathcal{P}_{\psi(h)}$, and
- $\delta \circ (c \cdot_h \gamma) = \alpha \circ (\beta \cdot_f p)$.

5.3. LEMMA. *Every element (ρ, γ) of $\mathcal{NC}_{\mathcal{P}}$ can be presented as a 3-simplex*

$$((g, q, \beta), (h, r, \delta), (f, p, \alpha), (\psi(\rho), p, \gamma)).$$

PROOF. One has to check that

$$q \cdot_{\psi(\rho)} p = q \cdot_{\psi(\rho)} (p \cdot_{\mathbb{1}_f} e_{\pi A}) = (q \cdot_{\psi(\mathbb{1}_{g \circ f})} p) \cdot_{\psi(\bar{\rho})} e_{\pi A} = \mathcal{P}_{\bar{\rho}}(q \cdot_{\psi(\mathbb{1}_{g \circ f})} p),$$

which follows from the associativity of \mathcal{P} and the following 3-simplex in \mathcal{C} .

$$\begin{array}{ccc} A & \xrightarrow{=} & A \\ h \downarrow & \bar{\rho} \nearrow g \circ f & \downarrow f \\ C & \xleftarrow{g} & B \end{array} = \begin{array}{ccc} A & \xrightarrow{=} & A \\ h \downarrow & \rho \nearrow f & \downarrow f \\ C & \xleftarrow{g} & B \end{array}$$

■

The lemma gives $\varphi(\rho, \gamma) := (\psi(\rho), p, \gamma)$.

– The maps u are given by

$$\begin{aligned} u_c &= (v_c, e_c), \\ u_{(A,a)} &= (v_A, a, \mathbb{1}_a), \\ u_{(f,p,\alpha)} &= (v_f, \alpha). \end{aligned}$$

We leave to the reader the specification of $(\mathcal{NC}_{\mathcal{P}})_3$ and the maps $\varphi: (\mathcal{NC}_{\mathcal{P}})_3 \rightarrow (\mathcal{NC}_{\mathcal{P}})_2$ and $u: (\mathcal{NC}_{\mathcal{P}})_2 \rightarrow (\mathcal{NC}_{\mathcal{P}})_3$, together with checking the conditions of Proposition 3.2.

5.4. DEFINITION. *For any (non-1-connected) categorical \mathbb{O} -operad we define its operadic Grothendieck construction explicitly in the very same way as for 1-connected operads above. The only difference from the construction for 1-connected operads is that there can be more objects and morphisms above the objects $v_c \in \mathcal{C}$, that is of form (v_c, a) .*

5.5. EXAMPLE. A strict monoidal category \mathcal{V} is a (non-1-connected) categorical \odot -operad, for \odot the terminal unary operadic 2-category. Its operadic Grothendieck construction $\int_{\odot} \mathcal{V}$ is isomorphic to \mathcal{V}^{Δ} of Example 3.10. The assignment $\mathcal{V} \mapsto \int_{\odot} \mathcal{V}$ is a fully faithful functor $\int_{\odot}: \text{StrMonCat} \rightarrow \text{uOpCat}$. In Section 7 we describe its left adjoint.

5.6. EXAMPLE. For the operadic 1-category $\text{Bq}^u(A)$ of unary bouquets on a set A , a $\text{Bq}^u(A)$ -operad \mathcal{P} is a 2-category with the set of objects A and $\int_{\text{Bq}^u(A)} \mathcal{P} \cong (\mathcal{NP}, \mathcal{DP})$.

5.7. EXAMPLE. If \mathbb{O} is a unary operadic 1-category, and \mathcal{P} a categorical \mathbb{O} -operad which is discrete in each component, we recover the discrete operadic Grothendieck construction of [BM15].

6. Operadic fibrations



OPERADIC fibrations (with splitting) are operadic functors lying in the essential image of the functor \int of the preceding section. We characterise them by certain lifting properties. A prototypical operadic fibration is thus the projection $\pi_{\mathcal{P}}: \int_{\mathbb{O}} \mathcal{P} \rightarrow \mathbb{O}$ from the previous section.

6.1. DEFINITION. *Let $\mathbb{O} = (X, \mathcal{C})$ be a unary operadic 2-category. The objects $u_c \in \mathcal{U}$, i.e. objects lying in the image of $u: \pi_0 \mathcal{C} \rightarrow \mathcal{C}_0$, are called trivial. A morphism $g: y \rightarrow x$ of a unary operadic 2-category is a quasibijection if for any map $f: z \rightarrow y$,*

$$\varphi(g \circ f \xrightarrow{\mathbb{1}_{g \circ f}} (g \circ f)) = u_{\varphi(f)}.$$

It follows that g has a trivial fiber $u_{\varphi(x)} = u_{\varphi(y)}$, which recovers 1-categorical quasibijections [BM23b, Section 1.1].

6.2. PROPOSITION. Let $\mathbb{O} = (X, \mathcal{C})$ be a unary operadic 2-category and \mathcal{P} a categorical \mathbb{O} -operad. A morphism $G = (g, q, \beta): (B, b) \rightarrow (C, c)$ of $\int_{\mathbb{O}} \mathcal{P}$ is a quasibijection if and only if g is a quasibijection in \mathbb{O} and $q = e_{\varphi(B)}$, the unit of the operad \mathcal{P} .

PROOF. Let $(f, p, \alpha): (A, a) \rightarrow (B, b)$ be a morphism of $\int_{\mathbb{O}} \mathcal{P}$ and $g: B \rightarrow C$ a quasibijection in \mathbb{O} . We check that

$$\varphi(\mathbf{1}_{(g, e_{\varphi(B)}, \beta) \circ (f, p, \alpha)}) = \varphi(\mathbf{1}_{g \circ f}, \mathbf{1}_p) = (\varphi(\mathbf{1}_{g \circ f}), p, \mathbf{1}_p) = (u_{\varphi(f)}, p, \mathbf{1}_p) = u_{\varphi(f, p)}.$$

Conversely, let (g, q, β) be a quasibijection. We get the equation

$$(\varphi(\mathbf{1}_{(g \circ f)}), q \cdot p, \mathbf{1}_{q \cdot p}) = \varphi(\mathbf{1}_{g \circ f}, \mathbf{1}_{q \cdot p}) = \varphi(\mathbf{1}_{(g, q, \beta) \circ (f, p, \alpha)}) = u_{\varphi(f, p)} = (u_{\varphi(f)}, p, \mathbf{1}_p),$$

which holds for any $f: A \rightarrow B$ and $p \in \mathcal{P}_{\varphi(f)}$. From the first components we see immediately that g is a quasibijection in \mathbb{O} . By evaluating $f = \mathbf{1}_B$ and $p = e_{\varphi(B)}$ we get $q = e_{\varphi(B)}$. ■

6.3. DEFINITION. For $k \geq 0$, Δ^k denotes the simplicial set $\text{Set}(-, [k])$. A $(3, 2)$ -horn is the sub-simplicial set $\iota: \Lambda_2^3 \hookrightarrow \Delta^3$, which is the union of all faces except the face (013) . A $(3, 2)$ -horn $\lambda = (d_0\lambda, d_1\lambda, d_3\lambda)$ in a simplicial set X is a morphism $\lambda: \Lambda_2^3 \rightarrow X$. A filler of a $(3, 2)$ -horn $\lambda: \Lambda_2^3 \rightarrow X$ is a 3-simplex $\sigma: \Delta^3 \rightarrow X$, whose restriction along the inclusion ι is λ .

6.4. DEFINITION. Let $p: \mathbb{P} \rightarrow \mathbb{O}$, be a morphism of operadic 2-categories with $\mathbb{P} = (X, \mathcal{C})$, $\mathbb{O} = (Y, \mathcal{D})$, and let $f \in X_2$. We say that f is operadic p -cartesian if for any $(3, 2)$ -horn λ in X with $d_0\lambda = f$ and any filler σ of $p\lambda$ in Y_2 , there exists a unique filler $\tilde{\sigma}$ of λ , such that $p\tilde{\sigma} = \sigma$.

6.5. DEFINITION. A morphism $p: \mathbb{P} \rightarrow \mathbb{O}$ is an operadic fibration if it induces an epimorphism $\pi_0 \mathbb{P} \twoheadrightarrow \pi_0 \mathbb{O}$ on the sets connected components, and for any $a, b \in X_1$ and $g \in Y_2$ with $d_0 g = p(a)$ and $d_2 g = p(b)$, there exists an operadic p -cartesian lift \tilde{g} of g in \mathcal{C} . An operadic fibration is split if there is a fixed choice of \tilde{g} for every a, b, g , such that the following two conditions hold. The choice of \tilde{g} will be denoted by $\ell(a, b, g)$.

- For every $x \in X_1$, $\ell(u_{\pi(x)}, x, u_{p(x)}) = u_x$ and $\ell(x, u_{\varphi(x)}, \mathbf{1}_{p(x)}) = \mathbf{1}_x$.
- For every 3-simplex $\sigma_{0123} \in Y_3$ with $f_{01} = p(x)$, $f_{12} = p(y)$, and $f_{23} = p(z)$ (cf. Section 2), for some $x, y, z \in X_1$,

$$\ell(d_1 \ell(z, y, \alpha_{123}), x, \alpha_{013}) = \ell(z, d_1 \ell(y, x, \alpha_{012}), \alpha_{023}).$$

6.6. PROPOSITION. A split operadic fibration $p: \mathbb{P} \rightarrow \mathbb{O}$ induces an isomorphism on connected components.

PROOF. Let $c, d \in \pi_0(\mathcal{D})$ with $p(c) = p(d)$. Then $p(u_d) = p(u_c) = u_{p(c)}$. By Proposition 3.2, (15), $u_{\varphi(u_c)} = u_c$. The splitting conditions give $\ell(u_d, u_c, \mathbf{1}_{u_{p(c)}}) = \mathbf{1}_{u_c}$. Comparing fibers of these two maps we see $u_c = u_d$, and hence $c = d$. ■

6.7. COROLLARY. Let $p: \mathbb{P} \rightarrow \mathbb{O}$ be a split operadic fibration. For any trivial object v of \mathbb{O} there is a unique trivial object u in \mathbb{P} above v .

6.8. PROPOSITION. For a unary operadic 2-category \mathbb{O} and a categorical \mathbb{O} -operad \mathcal{P} , the operadic functor $\pi_{\mathcal{P}}: \int_{\mathbb{O}} \mathcal{P} \rightarrow \mathbb{O}$ is a split operadic fibration.

PROOF. For objects (A, a) and (B, b) of $\int_{\mathbb{O}} \mathcal{P}$, and morphism $f: A \rightarrow B$ in \mathbb{O} , the operadic $\pi_{\mathcal{P}}$ -cartesian lift of f is $(f, b, \mathbb{1}_{a \cdot f b})$. It is straightforward to check the conditions. ■

6.9. THEOREM. There is a one-to-one correspondence of categorical \mathbb{O} -operads and split operadic fibrations over a unary operadic 2-category \mathbb{O} .

For a split operadic fibration $p: \mathbb{P} \rightarrow \mathbb{O}$ we define a categorical \mathbb{O} -operad \mathcal{P} . Let $\mathbb{P} = (Y, \mathcal{D})$ and $\mathbb{O} = (X, \mathcal{C})$. For $x \in \mathcal{C}$, let $\mathcal{P}_x := p_{qb}^{-1}(\mathbb{1}_x)$ be the subcategory of \mathcal{D} of all objects which are mapped to x and quasibijections which are mapped to $\mathbb{1}_x$. For a morphism g of \mathcal{C} , that is $g \in X_2$, the operad multiplication $\cdot_g: \mathcal{P}_{d_0 g} \times \mathcal{P}_{d_2 g} \rightarrow \mathcal{P}_{d_1 g}$ is defined using the p -cartesian lifts: for $a \in \mathcal{P}_{d_0 g}$ and $b \in \mathcal{P}_{d_2 g}$, $a \cdot_g b := d_1 \ell(a, b, g)$. Let $\alpha: a' \rightarrow a''$ in $\mathcal{P}_{d_0 g}$ and $\beta: b' \rightarrow b''$ in $\mathcal{P}_{d_2 g}$. Since α is a quasibijection, $\varphi(\mathbb{1}_{\alpha \circ \ell(a', b', g)}) = u_{b'}$. There is a $(3, 2)$ -horn $\lambda = (\ell(a'', b'', g), \alpha \circ \ell(a', b', g), \beta)$, and $p\lambda$ has a filler $\sigma = (g, g, \mathbb{1}, \mathbb{1})$ in \mathcal{C} . Since $\ell(a'', b'', g)$ is p -cartesian, there is a unique filler $\tilde{\sigma}$ of λ in \mathcal{D} . This defines $\alpha \cdot_g \beta := d_2 \tilde{\sigma}$. For $c \in \pi_0(\mathcal{C})$, by Corollary 6.7, there is a unique trivial object $e_c \in \mathcal{P}_{u_c}$, which provides the unit of \mathcal{P} . The functoriality, unitality and associativity of the operad \mathcal{P} is ensured by the splitting conditions of Definition 6.5 and the uniqueness of horn fillers. Using Proposition 6.2, it is straightforward to check that this construction is inverse to $\int_{\mathbb{O}}$. ■

6.10. EXAMPLE. Let $\mathbb{O} = \odot$, the terminal unary operadic 2-category. Theorem 6.9 provides a (possibly new) characterisation of strict monoidal categories. Namely, a strict monoidal category is a unary operadic 2-category $\mathbb{M} = (X_{\mathbb{M}}, \mathcal{C}_{\mathbb{M}})$ with the following lifting property. For any two objects $a, b \in \mathcal{C}_{\mathbb{M}}$, there exists a universal object $a \cdot b$ and a universal morphisms $\ell(a, b): a \cdot b \rightarrow a$ in $\mathcal{C}_{\mathbb{M}}$ with $\varphi(\ell(a, b)) = b$, and such that for any pair of maps $g: c \rightarrow a \cdot b$ and $h: \varphi(g) \rightarrow b$, there is a unique morphism $k: c \rightarrow a$ with $\varphi(k) = \varphi(h)$ and $(f, g, k, h) \in (\mathcal{NC}_{\mathbb{M}})_3$. The splitting provides a unique trivial object e of \mathbb{M} and conditions $\ell(e, x) = u_x$, $\ell(x, e) = \mathbb{1}_x$, and $\ell(d_1 \ell(a, b), c) = \ell(a, d_1 \ell(b, c))$. By looking at domains of these maps we get $e \cdot x = x = x \cdot e$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

7. Left adjoint to \int_{\odot}



ECALL the terminal unary operadic 2-category \odot , whose categorical operads are strict monoidal categories. In this section we construct a left adjoint to the functor $\int_{\odot}: \text{StrMonCat} \rightarrow \text{uOpCat}$. First we recall a few familiar adjunctions.

Let $\Delta_{\leq n}$ denote the full subcategory of Δ , which consists only of objects $[0], \dots, [n]$. An n -truncated simplicial set is a presheaf $\Delta_{\leq n}^{\text{op}} \rightarrow \text{Set}$. The category of n -truncated

simplicial sets is denoted by $\text{sSet}_{\leq n}$. The inclusion $\Delta_{\leq n} \subseteq \Delta$ induces an adjunction

$$\text{sSet}_{\leq n} \begin{array}{c} \xrightarrow{\text{cosk}_n} \\ \xleftarrow[\text{tr}_n]{\tau} \end{array} \text{sSet},$$

where tr_n is the restriction along the inclusion, and cosk_n is given by right Kan extension.

For $S \in \text{Set}$, $S\text{-coll}(\text{Set})$ denotes the category of functors

$$\coprod_{n \geq 0} (S^{\times n} \times S) \xrightarrow{Q} \text{Set},$$

and Multicat_S denotes the category of multicategories with the set of objects S . An S -collection Q is just a collection of sets $Q(a_1 \cdots a_k; a_0)$, $a_i \in S$, $k \geq 0$. There is the free-forgetful adjunction

$$\text{Multicat}_S \begin{array}{c} \xrightarrow{U} \\ \xleftarrow[F]{\tau} \end{array} S\text{-coll}(\text{Set}).$$

The multimorphisms of FQ are planar rooted trees with inner vertices labeled by elements of the collection Q .

Last, we recall the adjunction

$$\text{StrMonCat} \begin{array}{c} \xrightarrow{R} \\ \xleftarrow[L]{\tau} \end{array} \text{Multicat}.$$

For $\mathcal{M} \in \text{StrMonCat}$, the multicategory $R\mathcal{M}$ has $R\mathcal{M}(a_1 \cdots a_k; a_0) = \mathcal{M}(a_1 \otimes \cdots \otimes a_k, a_0)$. For $\mathcal{M} \in \text{Multicat}$, the objects of LM are lists of objects of \mathcal{M} . A morphism of lists $(a_1, \dots, a_k) \rightarrow (b_1, \dots, b_l)$ is an l -tuple tuple of multimorphisms of \mathcal{M} . The monoidal product is concatenation of lists and the monoidal unit is the empty list.

We first establish an adjunction

$$\text{StrMonCat} \begin{array}{c} \xrightarrow{\Psi} \\ \xleftarrow[\Phi]{\tau} \end{array} \text{sSet}_{\leq 3}.$$

For $\mathcal{M} \in \text{StrMonCat}$ we define $\Psi\mathcal{M} = \text{tr}_3 \mathcal{N}\mathcal{B}\mathcal{M}$, the 3-truncated nerve of the one-object 2-category $\mathcal{B}\mathcal{M}$.

7.1. LEMMA. *Let \mathcal{C} be a small category, \mathcal{PC} the category of presheaves on \mathcal{C} , and $G: \mathcal{D} \rightarrow \mathcal{PC}$ any functor from a cocomplete category. Let F_0 be a functor $F_0: \mathcal{C} \rightarrow \mathcal{D}$, such that for any object $s \in \mathcal{C}$ and $a \in \mathcal{D}$, there is a natural isomorphism*

$$\mathcal{D}(F_0 b, a) \cong \mathcal{PC}(yb, Ga),$$

where $y: \mathcal{C} \rightarrow \mathcal{PC}$ is the Yoneda embedding. Then $F = \text{Lan}_y F_0$, the left Kan extension of F_0 along y , is left adjoint to G .

PROOF. Any $X \in \mathcal{PC}$ is isomorphic to the colimit of representables $X \cong \text{colim } yb$, where the colimit is taken over the category of elements of X . There are natural isomorphisms

$$\begin{aligned} \mathcal{D}(FX, a) &\cong \mathcal{D}(F \text{colim } yb, a) \cong \mathcal{D}(\text{colim } Fyb, a) \cong \mathcal{D}(\text{colim } F_0 b, a) \cong \lim \mathcal{D}(F_0 b, a) \cong \\ &\cong \lim \mathcal{PC}(yb, Ga) \cong \mathcal{PC}(\text{colim } yb, Ga) \cong \mathcal{PC}(X, Ga). \end{aligned}$$

■

By the preceding lemma it suffices to define a functor $\Phi_0: \Delta_{\leq 3} \rightarrow \text{StrMonCat}$ with

$$\text{StrMonCat}(\Phi_0[k], M) \cong \text{sSet}_{\leq 3}(\Delta^k, \Psi M),$$

for $k = 0, 1, 2, 3$, where $\Delta^k := y[k] = \text{Set}(-, [k])$. By the Yoneda lemma,

$$\text{sSet}_{\leq 3}(\Delta^k, \Psi M) \cong (\Psi M)_k.$$

- For $k = 0$, we define $\Phi_0[0] = 1$, the terminal category, which is also the initial monoidal category. We see that

$$1 \cong (\Psi M)_0 \cong \text{StrMonCat}(1, M).$$

- For $k = 1$, we define $\Phi_0[1] = (\mathbb{N}, +, 0)$, the free strict monoidal category on one object. A monoidal functor $H: \mathbb{N} \rightarrow M$ is determined by an object $H(1) \in M$, hence

$$M_0 = (\Psi M)_1 \cong \text{StrMonCat}(\mathbb{N}, M).$$

- For $k = 2$, we consider the simplicial set Δ^2 , whose non-degenerate simplices are:

$$\begin{array}{ccc} x_0 & \xrightarrow{f_{01}} & x_1 \\ & \searrow \alpha_{012} & \swarrow \\ & x_2 & \end{array}$$

$$\begin{array}{ccc} & f_{02} & \\ \swarrow & & \searrow \\ x_0 & & x_1 \\ & f_{12} & \end{array}$$

Let $S = \{f_{01}, f_{12}, f_{02}\}$. We define an S -collection Q by $Q(f_{12}, f_{01}; f_{02}) = \{\alpha_{012}\}$ and empty otherwise. We put $\Phi_0[2] = LFQ$. Objects of the monoidal category $\Phi_0[2]$ are lists (a_1, \dots, a_k) , $a_i \in \{f_{01}, f_{12}, f_{02}\}$, $k \geq 0$. Morphisms $(a_1, \dots, a_k) \rightarrow (b_1, \dots, b_l)$ are tuples (g_1, \dots, g_l) , where each g_i is either α_{012} or the identity. In other words, $\Phi_0[2]$ is the universal strict monoidal category containing a lax triangle. A monoidal functor $H: \Phi_0[2] \rightarrow M$ is determined by three objects $H(f_{01}), H(f_{12}), H(f_{02})$ of M , and a morphism $H(f_{12}) \otimes H(f_{01}) \xrightarrow{H(\alpha_{012})} H(f_{02})$, which gives

$$(\Psi M)_2 \cong \text{StrMonCat}(\Phi_0[2], M).$$

- For $k = 3$, we consider a filled 3-simplex σ_{0123} with boundary

$$\begin{array}{ccc} x_0 & \xrightarrow{f_{01}} & x_1 & & x_0 & \xrightarrow{f_{01}} & x_1 \\ f_{03} \downarrow & \nearrow \alpha_{013} & \downarrow f_{12} & & f_{03} \downarrow & \nearrow \alpha_{012} & \downarrow f_{12} \\ x_3 & \xleftarrow{f_{23}} & x_2 & & x_3 & \xleftarrow{f_{23}} & x_2 \end{array}$$

Let B be the set of its 1-simplices,

$$B = \{f_{ij} \mid 0 \leq i < j \leq 3\}.$$

We define a B -collection R by

- $R(f_{jk} f_{ij}; f_{ijk}) = \{\alpha_{ijk} \mid 0 \leq i < j < k \leq 3\}$.
- $R(f_{23} f_{12}, f_{01}; f_{03}) = \{\sigma_{0123}\}$, and empty otherwise.

Consider FR/\sim , the quotient of the free multicategory, where \sim is generated by:

$$\begin{array}{ccc} \begin{array}{c} (f_{23} \quad f_{12} \quad f_{01}) \\ \diagup \quad \diagdown \\ \alpha_{013} \bullet \\ \diagdown \quad \diagup \\ f_{13} \\ \alpha_{123} \bullet \\ \diagup \quad \diagdown \\ (f_{03}) \end{array} & \sim & \begin{array}{c} (f_{23} \quad f_{12} \quad f_{01}) \\ \diagup \quad \diagdown \\ \sigma_{0123} \bullet \\ \diagup \quad \diagdown \\ (f_{03}) \end{array} & \sim & \begin{array}{c} (f_{23} \quad f_{12} \quad f_{01}) \\ \diagup \quad \diagdown \\ \alpha_{012} \bullet \\ \diagdown \quad \diagup \\ f_{02} \\ \alpha_{023} \bullet \\ \diagup \quad \diagdown \\ (f_{03}) \end{array} \end{array}$$

We put $\Phi[3] = L(FR/\sim)$. Equivalently, the monoidal category $\Phi[3]$ consists of objects (a_1, \dots, a_k) , $a_i \in B$, $k \geq 0$, and morphisms $(a_1, \dots, a_k) \rightarrow (b_1, \dots, b_l)$ are tuples (g_1, \dots, g_l) , where each g_i is either an identity on $f_{ij} \in B$, a non-degenerate 2-simplex α_{ijk} , or the 3-simplex σ_{0123} . An example of composition of morphisms is:

$$\begin{array}{ccc} \begin{array}{c} (f_{23} \quad f_{12} \quad f_{01} \quad f_{02} \quad f_{23} \quad f_{02} \quad f_{23} \quad f_{12} \quad f_{01}) \\ \diagup \quad \diagdown \quad | \quad | \quad \diagup \quad \diagdown \\ \alpha_{123} \bullet \\ | \quad | \quad | \quad | \quad | \quad | \\ (f_{13} \quad f_{01} \quad f_{02} \quad f_{23} \quad f_{02} \quad f_{03}) \\ \alpha_{013} \bullet \\ | \quad | \quad | \quad | \quad | \quad | \\ (f_{13} \quad f_{02} \quad f_{03}) \end{array} & = & \begin{array}{c} (f_{23} \quad f_{12} \quad f_{01} \quad f_{02} \quad f_{23} \quad f_{02} \quad f_{23} \quad f_{12} \quad f_{01}) \\ \diagup \quad \diagdown \quad | \quad | \quad \diagup \quad \diagdown \\ \sigma_{0123} \bullet \\ | \quad | \quad | \quad | \quad | \quad | \\ (f_{13} \quad f_{02} \quad f_{03}) \end{array} \end{array}$$

A monoidal functor $H: \Phi[3] \rightarrow M$ is determined by six objects $H(f_{ij}) \in M$, four morphisms

$$H(f_{jk}) \otimes H(f_{ij}) \xrightarrow{H(\alpha_{ijk})} H(f_{ik}),$$

and a morphism

$$H(f_{23}) \otimes H(f_{12}) \otimes H(f_{01}) \xrightarrow{H(\sigma_{0123})} H(f_{03}),$$

which satisfy

$$H(\alpha_{123}) \circ (H(\alpha_{013}) \otimes \mathbb{1}_{H(f_{01})}) = H(\sigma_{0123}) = H(\alpha_{023}) \circ (\mathbb{1}_{H(f_{23})} \otimes H(\alpha_{012})).$$

This gives

$$(\Psi M)_3 \cong \text{StrMonCat}(\Phi[3], M).$$

By a k -simplex in a monoidal category \mathcal{M} we mean a k -simplex in the 2-category \mathcal{BM} (cf. Definition 2.1). Denote by σ_l the generating l -simplex of $\Phi_0[l]$. Let $k, l \leq 3$ and $f: [k] \rightarrow [l]$. By the above

$$\text{StrMonCat}(\Phi_0[k], \Phi_0[l]) \cong \{k\text{-simplices of } \Phi_0[l]\},$$

i.e. any monoidal functor $\Phi_0[k] \rightarrow \Phi_0[l]$ is determined by a k -simplex in $\Phi_0[l]$. In particular, $\Phi_0 f$ is determined by the k -simplex $\sigma_l \circ f$. This completes the functor

$$\Phi_0: \Delta_{\leq 3} \rightarrow \text{StrMonCat},$$

such that the above isomorphisms are natural, which finishes the construction of the adjunction

$$\text{StrMonCat} \begin{array}{c} \xrightarrow{\Psi} \\[-1ex] \xleftarrow[\Phi]{\tau} \end{array} \text{sSet}_{\leq 3}.$$

Since $\mathcal{NB}\mathcal{M}$ is 3-coskeletal, $\mathcal{NB}\mathcal{M} \cong \text{cosk}_3(\text{tr}_3(\mathcal{NB}\mathcal{M}))$, and from the adjunction $\text{tr}_3 \dashv \text{cosk}_3$ we have

$$\text{sSet}(X, \mathcal{NB}\mathcal{M}) \cong \text{sSet}_{\leq 3}(\text{tr}_3 X, \text{tr}_3 \mathcal{NB}\mathcal{M}) \cong \text{StrMonCat}(\Phi \text{tr}_3 X, M).$$

Thus, for a unary operadic 2-category $\mathbb{O} = (X, \mathcal{C})$, we have the adjunction $\Phi \text{tr}_3 \dashv \int_{\odot}$,

$$\text{uOpCat}(\mathbb{O}, \int_{\odot} \mathcal{M}) := \text{sSet}(X, \mathcal{NB}\mathcal{M}) \cong \text{StrMonCat}(\Phi \text{tr}_3 X, \mathcal{M}).$$

7.2. EXAMPLE. For a 1-category \mathcal{C} (viewed as a 2-category with only trivial 2-cells), and its associated unary operadic 2-category $\mathbb{C} = (\mathcal{NC}, \mathcal{DC})$. The monoidal category $\mathcal{M}_{\mathcal{C}} := \Phi \text{tr}_3 \mathcal{NC}$ has objects lists (f_1, \dots, f_k) , $f_i \in \mathcal{C}_1, k \geq 0$, and morphisms are generated by

$$(f_1, \dots, g, h, \dots, f_k) \xrightarrow{\text{compose}} (f_1, \dots, g \circ h, \dots, f_k).$$

The monoidal product is concatenation and the monoidal unit is the empty list.

8. Closing remarks and further directions

 HERE is a straightforward generalisation of our theory to unary operadic bi-categories and categorical pseudo-operads (operads with relaxed associativity and unitality conditions). We expect that these correspond to non-splitting fibrations. Working in this generalised framework seems more natural, in particular one could work directly with monoidal categories and not with their strictification. The proofs might become more complicated however.

A direct continuation of our work is to establish the 2-categorical (or bicategorical) framework for operadic categories of general arities. This is a work in progress. Unfortunately, there is no such clear characterisation using simplicial sets as in the unary case, which complicates the situation. We believe that a good path is to follow Proposition 3.8. The proposed definition is:

8.1. DEFINITION. An operadic 2-category is a 2-category \mathbb{O} equipped with

- a cardinality functor $|-|: \mathbb{O} \rightarrow \mathcal{S}$ to the skeletal category of finite sets,
- normalised lax functors $\varphi_x: \mathbb{O}/x \rightarrow \mathbb{O}^{|x|}$, for every object x of \mathbb{O} , and
- chosen local lali-terminal objects u_c , in each connected component c of \mathbb{O} ,
- such that analogues of the axioms of operadic 1-categories hold.

When the cardinality functor is constantly zero, the theory should recover ordinary categories, such that their operads are presheaves and operadic fibrations are categorical fibrations.

Let $F: \mathbb{O} \rightarrow \mathbb{P}$ be an operadic functor. It induces a restriction functor on the categories of operads

$$F^*: \mathbb{P}\text{-oper(Cat)} \rightarrow \mathbb{O}\text{-oper(Cat)}.$$

An interesting problem is to find a formula for its left adjoint. Evaluating this on the terminal \mathbb{P} -operad we possibly get a left adjoint to the operadic Grothendieck construction for general \mathbb{O} .

References

- [BM15] M.A. Batanin and M. Markl. Operadic categories and duoidal Deligne’s conjecture. *Advances in Mathematics*, 285:1630–1687, 2015.
- [BM23a] M.A. Batanin and M. Markl. Operadic categories as a natural environment for Koszul duality, *Compositionality*, 5(3), 2023.
- [BM23b] M.A. Batanin and M. Markl. Koszul duality for operadic categories, *Compositionality*, 5(4), 2023.
- [BM24] M.A. Batanin and M. Markl. Operads, operadic categories and the blob complex, *Applied Categorical Structures*, 32(1), 2024.
- [C&al22] G.S.H. Cruttwell, B. Gavranović, N. Ghani, P. Wilson, and F. Zanasi. Categorical foundations of gradient-based learning, *European Symposium on Programming*, 2022.
- [D01] J. Duskin. Simplicial matrices and the nerves of weak n-categories I: nerves of bicategories, *Theory and Applications of Categories*, 9(10), 2001.
- [FST21] B. Fong, D. Spivak, and R. Tuyéras. Backprop as Functor: A compositional perspective on supervised learning, *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science*, 11, 2021.

- [GKW21] R. Garner, J. Kock, and M. Weber. Operadic categories and décalage, *Advances in Mathematics*, 377:107440, 2021.
- [G19] B. Gavranović. Compositional Deep Learning, <https://arxiv.org/abs/1907.08292>, 2019.
- [H23] P. Hackney. Operadic categories and 2-Segal sets, <https://arxiv.org/abs/2312.00756>, 2023.
- [JY20] N. Johnson and D. Yau. 2-Dimensional Categories, <https://arxiv.org/abs/2002.06055>, 2020.
- [Ker] J. Lurie. Kerodon, <https://kerodon.net>, 2024.
- [M24] M. Markl. Bivariant operadic categories, <https://arxiv.org/abs/2402.12963>, 2024.
- [MW12] S. Morrison, and K. Walker. Blob homology, *Geometry & Topology*, 16:1481–1607, 2012.
- [Š24] M. Štěpán. Colax adjunctions and lax-idempotent pseudomonads, <https://arxiv.org/abs/2405.00488>, 2024.
- [T] D. Trnka. Category-colored operads, internal operads, and Markl O-operads, *Theory and Applications of Categories*, 39(30):874–915, 2023.

Department of Mathematics and Statistics, Masaryk University, Kotlářská 2, 61137 Brno, and Institute of Mathematics, Czech Academy of Sciences, Žitná 25, 115 67 Prague 1, Czech Republic

Fixpoint operators for 2-categorical structures

Zeinab Galal
LIP6, Sorbonne University

Abstract—Fixpoint operators are tools to reason on recursive programs and data types obtained by induction (e.g. lists, trees) or coinduction (e.g. streams). They were given a categorical treatment with the notion of categories with fixpoints. A theorem by Plotkin and Simpson characterizes existence and uniqueness of fixpoint operators for categories satisfying some conditions on bifree algebras and recovers the standard examples of the category Cppo (ω -complete pointed partial orders and continuous functions) in domain theory and the relational model in linear logic.

We present a categorification of this result and develop the theory of 2-categorical fixpoint operators where the 2-dimensional framework allows to model the execution steps for languages with (co)inductive principles. We recover the standard categorical constructions of initial algebras and final coalgebras for endofunctors as well as fixpoints of generalized species and polynomial functors.

INTRODUCTION

Fixpoints operators play an important role to model infinite computation in a wide range of computer science disciplines: design of programming languages, verification, model checking, databases, concurrency theory, type theory, etc. A fixpoint for a program t is an input x such that there is a calculation sequence between x and $t(x)$. For example, a fundamental property of untyped λ -calculus is the existence of fixpoint combinators *i.e.* terms \mathbf{Y} such that for any λ -term t , there is a reduction path connecting the terms $\mathbf{Y}t$ and $t(\mathbf{Y}t)$.

From a set-theoretic viewpoint, the standard notion of fixpoint for an endomap $f : A \rightarrow A$ is an element $x \in A$ such that $f(x) = x$ and it was axiomatized with the notion of categories with fixpoint operators [1]. There is however no notion of categorical fixpoint operator taking into account the computational reduction steps and not collapsing them into strict equalities.

The objective of this paper is to work in a 2-dimensional framework to model explicitly the reductions of languages with fixpoints and study their coherences *i.e.* the equations satisfied by the program computations steps. It fits into the line of research of *categorifying* models of computation by replacing semantics where types are sets or preorders with richer categorical structures to establish stronger mathematical invariants.

Categories in dimension one have objects and morphisms that can be composed. We can consider an additional dimension with the notion of 2-categories or bicategories which have objects, 1-morphisms that can be composed and 2-morphisms that can be composed in two different ways that verify compatibility conditions. These 2-morphisms are thought of

as morphisms between 1-morphisms. When using bidimensional categorical structures to model computations, we can study program execution steps as primitive objects as they become explicit 2-morphisms carrying information on program reductions. It has seen many applications in concurrency, game semantics, type theory, higher dimensional rewriting [2], [3], [4], [5], [6], [7], [8], [9], [10].

When generalizing preorder semantics to richer categorical semantics, least fixpoints become initial algebras and greatest fixpoints become final coalgebras. In both cases, strict equalities for fixpoints $t(x) = x$ are now represented by explicit isomorphisms

$$t(x) \xrightarrow{\text{algebra}} x \quad x \xrightarrow{\text{coalgebra}} t(x)$$

capturing the dynamic aspect of fixpoint reductions. Initial algebras correspond to recursive definition with induction as a logical reasoning principle. They are typically used to model finite data types (such as finite lists and trees) with a constructor operation.

Dually, final coalgebras are the counterpart of corecursion and coinduction where structures are described with a destrutor or observer operation. They are now widely used in computer science to model state-based systems with circular or non-terminating behavior (automata, transitions systems, network dynamics etc.). The categorical coalgebraic framework is also a standard tool to study notions such as bisimilarity and trace equivalence. It has been used in functional programming to model lazy datatypes in languages such as Haskell or reactive programs in proof assistants such as Coq and Agda.

Settings where initial algebra and final coalgebras coincide because of limit-colimit coincidence arguments have also been studied to solve equations with mixed variance variables (such as reflexive objects $D \cong D \Rightarrow D$ for λ -calculus models [11]). These algebras are called *bifree* and they provide a framework where inductive and coinductive arguments are equivalent. An important result by Plotkin and Simpson in this area states that provided some conditions on bifree algebras are satisfied, we obtain the existence of a unique uniform fixpoint operator for 1-categories [1].

While using initial algebra or final coalgebra semantics to model infinite computations are now well-established traditions in computer science, there is no counterpart axiomatization of fixpoint operators for 2-categorical structures and the goal of this paper is develop their theory. In order to axiomatize the notion of 2-dimensional fixpoint operator, the main difficulty is to understand what axioms and coherences the rewriting 2-morphisms should satisfy. We both need to

ensure that the equations are correct, *i.e.* that they hold in concrete models, and we also need completeness properties, *i.e.* we want to ensure that we have stated them all.

We proceed in two steps: we first categorify the restricted case of the Plotkin-Simpson theorem characterizing the existence of a unique uniform fixpoint operator for 1-categories to extract an axiomatization of pseudo-fixpoint operator for 2-categories. These operators form a category and uniqueness of the fixpoint operator is replaced by a contractibility property: there is a unique isomorphism between any two fixpoint operators. The second step is to generalize to models where fixpoints are not unique to verify that our axiomatization holds. Our motivation for choosing this approach is that the Plotkin-Simpson proof constructs explicitly a canonical fixpoint operator using the bifree algebras and then uses the fixpoint axioms to show that it is in fact unique. In the 2-dimensional case, the pseudo-bifree algebras allow us to construct a canonical pseudo-fixpoint operator and in order to obtain a contractibility property, we need to impose certain coherence equations on the structural reduction 2-morphisms which provides a guideline for the axiomatization of general pseudo-fixpoint operators.

Related works

Categorification of recursive domain theory has an established history with the work of Lambek, Freyd, Lehmann, Adamek, Taylor, Fiore, Winskel and Cattani [12], [13], [14], [15], [16], [17], [18]. In this paper, we mostly use results by Cattani, Fiore and Winskel generalizing the notion of algebraically compact categories for enriched categories to enriched bicategories and proving limit-colimit coincidence theorems in this setting with applications to presheaf models of concurrency [17], [18]. From a syntactic viewpoint, Pitts presented a candidate 2-dimensional type theory for fixpoints which can serve to prove coherence theorems for our notion of pseudo-fixpoint operators [19].

Ponto and Shulman have also studied a categorification of the notion of fixpoint and trace for bicategories in a different direction [20], [21]. They consider the trace of endo-2-cells $\alpha : f \Rightarrow f$ for a 1-cell $f : A \rightarrow B$; whereas in our case we still want to compute the trace or fixpoint of endo-1-cells but up to explicit rewriting 2-cells. We aim to investigate whether the two approaches can be compared for the cartesian closed setting in future work.

Plan of the paper

- We start in Section I by recalling the standard theory of 1-categorical fixpoint operators and give examples from domain theory and linear logic that can be recovered by the Plotkin-Simpson theorem.
- In Section II, we state the definitions of pseudo-fixpoint operators for 2-categories with uniformity and dinaturality axioms.
- We prove in Section III a generalization of the Plotkin-Simpson theorem for 2-categories where pseudo-fixpoint

operators now form a category and uniqueness of the fixpoint operator is replaced by a contractibility property.

- We show in Section V how the notion of 2-categorical fixpoint we developed in Section II is verified in well-known 2-categorical models.

I. FIXPOINT OPERATORS FOR 1-CATEGORIES

Definition I.1. Let \mathbb{D} be a category with a terminal object 1 , a *fixpoint operator* on \mathbb{D} is a family of functions $(-)^* : \mathbb{D}(A, A) \rightarrow \mathbb{D}(1, A)$ indexed by the objects A of \mathbb{D} verifying that for all morphisms $f : A \rightarrow A$,

$$f^* = f f^*. \quad (\text{fix})$$

We can further require additional axioms such as uniformity and dinaturality which are usually satisfied in concrete models and can serve to characterize uniqueness properties of fixpoint operators [22]. We can also consider parametrized axioms giving the connection with traced monoidal categories [23] but we leave the parametrized case for an accompanying paper.

For a category with fixpoints, the uniformity principle (also called Plotkin's axiom) is relative to a subcategory of “strict maps” and is used to characterize the fixpoint operator uniquely without relying on order-theoretic arguments [24], [25]. For domain-like structures, strict maps are usually the ones which preserve bottom elements \perp whereas general maps are just assumed to be Scott-continuous. Another possibility is to consider linear maps instead of strict maps, *i.e.* maps which commute with all suprema not just directed ones and these are the typical examples in linear logic models with fixpoints. Freyd has also considered the case of a reflective subcategory for the subcategory of strict maps [26].

Definition I.2. Let \mathbb{C}, \mathbb{D} be categories with terminal objects and $J : \mathbb{C} \rightarrow \mathbb{D}$ be an identity-on-objects functor preserving terminal objects.

- A fixpoint operator $(-)^*$ on \mathbb{D} is said to be *uniform with respect to J* if for every $s : A \rightarrow B$ in \mathbb{C} and $f : A \rightarrow A, g : B \rightarrow B$ in \mathbb{D} , we have:

$$J(s)f = gJ(s) \quad \text{implies} \quad J(s)f^* = g^*. \quad (\text{unif})$$

- A fixpoint operator $(-)^*$ on \mathbb{D} is *dinatural* if for every $f : A \rightarrow B$ and $g : B \rightarrow A$ in \mathbb{D} ,

$$(fg)^* = f(gf)^*. \quad (\text{dinat})$$

Remark 1. We can in fact consider bijective-on-objects functors for $J : \mathbb{C} \rightarrow \mathbb{D}$ but we restrict to identity-on-objects functors to make the notation less cumbersome.

As mentioned in the introduction, our approach is to use a categorification of the Plotkin-Simpson theorem which characterizes existence and uniqueness of fixpoint operators for 1-categories that are obtained as Kleisli categories of comonads satisfying some conditions on bifree algebras [1].

Recall that for an endofunctor $T : \mathbb{D} \rightarrow \mathbb{D}$, a *bifree T -algebra* (also called dialgebra, compact algebra or free bialgebra) is an initial T -algebra $(A, a : TA \rightarrow A)$ such that the inverse of a is a final T -coalgebra $(A, a^{-1} : A \rightarrow TA)$.

Theorem I.3 (Plotkin-Simpson [1]). *Let \mathbb{C} be a category equipped with a comonad (T, δ, ε) and a terminal object. We denote by \mathbb{D} the co-Kleisli category \mathbb{C}_T and by $J : \mathbb{C} \rightarrow \mathbb{D}$ the free functor induced by the comonadic adjunction.*

- 1) *If the endofunctor T has a bifree algebra, then \mathbb{D} has a unique uniform (with respect to J) fixpoint operator.*
- 2) *If \mathbb{C} is cartesian and the endofunctor TT has a bifree algebra, then \mathbb{D} has a unique uniform (with respect to J) dinatural fixpoint operator.*

We proceed to give the general recipe to obtain bifree algebras for endofunctors in a suitable preorder-enriched setting in order to motivate the generalizations to dimension 2 in Section V.

We denote by \mathbf{Cpo} the category whose objects are ω -complete partial orders and morphisms are Scott-continuous functions (monotone maps preserving colimits of ω -chains). If we restrict the objects to *pointed cpo's* (with a bottom element \perp), we denote this subcategory \mathbf{Cppo} . We can further restrict the morphisms to Scott-continuous functions that are *strict* (preserving bottom elements) and we denote by \mathbf{Cppo}_\perp the category of pointed cpo's and strict Scott-continuous functions. We are interested in \mathbf{Cppo}_\perp -enriched categories since they provide a well-behaved setting to compute bifree algebras as we will see below. Explicitly, a category \mathbb{C} is \mathbf{Cppo}_\perp -enriched if each homset $\mathbb{C}(A, B)$ is a cpo with a bottom element \perp and composition $\mathbb{C}(A, B) \times \mathbb{C}(B, C) \rightarrow \mathbb{C}(A, C)$ is Scott-continuous and strict in both components. Examples of \mathbf{Cppo}_\perp -enriched categories include the category \mathbf{Cppo}_\perp which is enriched over itself and the category \mathbf{Rel} whose objects are sets and morphisms are binary relations. Another example is the category \mathbf{Lin} of preorders and ideal relations (binary relations $R \subseteq A \times B$ between preorders that are up-closed in A and down-closed in B). The category \mathbf{Rel} can be viewed as the subcategory of \mathbf{Lin} containing discrete preorders.

The following theorem is a consequence of the limit-colimit coincidence theorem by Smyth and Plotkin [27] which was generalized by Fiore [16]:

Theorem I.4. *Let \mathbb{C} be a \mathbf{Cppo}_\perp -enriched category. If \mathbb{C} has an initial object and ω -colimits of chains of embeddings, then \mathbb{C} is \mathbf{Cpo} -algebraically compact. It means that for every endofunctor $T : \mathbb{C} \rightarrow \mathbb{C}$, if T is \mathbf{Cpo} -enriched, i.e. for all A, B , the induced map*

$$\mathbb{C}(A, B) \longrightarrow \mathbb{C}(TA, TB)$$

preserves colimits of ω -chains, then T has a bifree algebra.

We proceed to give examples of well-known fixpoint operators that can be recovered from the Plotkin-Simpson theorem. Consider the lifting comonad $(-)_\perp$ on \mathbf{Cppo}_\perp : this comonad freely adjoins a bottom element and its co-Kleisli category is isomorphic to \mathbf{Cppo} . Since the endofunctors $(-)_\perp$ and $(-)_\perp(-)_\perp$ are \mathbf{Cppo} -enriched, the category \mathbf{Cppo} has a unique dinatural fixpoint operator uniform with respect to the

free functor $\mathbf{Cppo}_\perp \rightarrow \mathbf{Cppo}$ and it is given by the standard formula

$$f^* = \bigvee_{n \in \omega} f^n(\perp)$$

for $f : A \rightarrow A$ in \mathbf{Cppo} . We can also consider the finite multiset comonad \mathcal{M}_{fin} on the category \mathbf{Rel} . It leads to a *quantitative* model of linear logic where multisets allow to count multiplicities of the inputs for a term [28]. As \mathcal{M}_{fin} and $\mathcal{M}_{\text{fin}}\mathcal{M}_{\text{fin}}$ have bifree algebras, we recover the fixpoint operator in the relational model [29] and obtain that it is unique.

Lastly, the category \mathbf{Lin} (also called the linear Scott category \mathbf{ScottL}) can be equipped with the \vee -semi-lattice comonad yielding a *qualitative* model of linear logic where substitution allows for duplication and erasure. This comonad also verifies the necessary enrichment conditions to obtain a unique uniform dinatural fixpoint operator.

II. BIDIMENSIONAL FIXPOINTS OPERATORS

We state in this section the definition of fixpoint operators for 2-categories. We will show in the remaining sections how the axioms we present arise from the generalization of the Plotkin-Simpson construction in dimension 2 and how they are verified in concrete examples. When moving to a higher dimension, we have several possibilities depending on the degrees of strictness and direction of the 2-morphisms (strict, pseudo, (op)lax).

In this paper, for space considerations, we will focus on one case: *pseudo-fixpoint operators for 2-categories* and leave the remaining cases and the bicategorical weakening for the long version. Even if we only consider the pseudo case where the 2-cells are invertible, we will still provide a direction for the arrows to give a better understanding of where they come from and provide a guideline for the directed lax and oplax cases.

Definition II.1. Let \mathcal{D} be a 2-category with a terminal object $\mathbf{1}$. A *pseudo-fixpoint operator* on \mathcal{D} consists of a family of functors indexed by the objects A of \mathcal{D} :

$$(-)_A^* : \mathcal{D}(A, A) \rightarrow \mathcal{D}(\mathbf{1}, A)$$

together with a family of natural isomorphisms fix_A (we will omit the subscripts for objects to simplify the notation) with components:

$$\begin{array}{ccc} & f^* & \\ \mathbf{1} & \nearrow \text{fix}_f & \downarrow f \\ & f^* & \end{array}$$

for a 1-cell $f : A \rightarrow A$ in \mathcal{D} . Naturality means that for an invertible 2-cell $\alpha : f \Rightarrow g$ in \mathcal{D} , we have:

$$\begin{array}{c}
 \text{Diagram showing two 2-cell configurations:} \\
 \text{Left: } 1 \xrightarrow{\text{fix}_g} A \quad \text{Right: } 1 \xrightarrow{\text{fix}_f} A \\
 \text{Both have a blue oval labeled } g^* \text{ and a red oval labeled } f^*. \\
 \text{Left: } \text{fix}_g \text{ is a 2-cell from } 1 \text{ to } A \text{ through } g^*, f^*, \text{ and } \alpha^*. \\
 \text{Right: } \text{fix}_f \text{ is a 2-cell from } 1 \text{ to } A \text{ through } f^*, g^*, \text{ and } \alpha^*.
 \end{array}$$

Remark 2. Strictly speaking, our pseudo-fixpoint operators act on the sub-2-category of \mathcal{D} where we only consider invertible 2-cells. Indeed, for a general 2-cell $\alpha : f \Rightarrow g$, α^* is not defined unless α is invertible and we need to consider (op)lax-fixpoint operators to act on non-invertible 2-cells. In order to lighten the notation, for a 2-category \mathcal{D} , we implicitly use the same notation for \mathcal{D} and its sub-2-category containing only invertible 2-cells for the rest of the paper.

In the preordered case, we can compare fixpoint operators pointwise and we are usually interested in the least and greatest fixpoints. In the categorified setting, fixpoint operators form a category so there can be more than one way of comparing two fixpoint operators. Morphisms of pseudo-fixpoint operators are transformations that commute with the structural 2-cells fix . Initial objects in this category correspond to least fixpoints while terminal objects correspond to greatest fixpoints. The uniqueness property for fixpoint operators in the preorder setting now becomes a contractibility property for the category of fixpoint operators. A category is *contractible* when it is not empty and for any two objects, there is a unique isomorphism between them (in particular, it is a groupoid).

Definition II.2. Let $((-)^*, \text{fix}^*)$ and $((-)^{\dagger}, \text{fix}^{\dagger})$ be two pseudo-fixpoint operators on a 2-category \mathcal{D} . A pseudomorphism of pseudo-fixpoint operators $((-)^*, \text{fix}^*) \rightarrow ((-)^{\dagger}, \text{fix}^{\dagger})$ consists of a family of natural isomorphisms

$$\delta_A : (-)_A^* \Rightarrow (-)_A^{\dagger} : \mathcal{D}(A, A) \rightarrow \mathcal{D}(1, A)$$

indexed by the objects A of \mathcal{D} that commutes with the structural 2-cells fix , i.e. it satisfies the following coherence for every $f : A \rightarrow A$:

$$\begin{array}{c}
 \text{Diagram showing two 2-cell configurations:} \\
 \text{Left: } 1 \xrightarrow{\text{fix}_f^{\dagger}} A \quad \text{Right: } 1 \xrightarrow{\text{fix}_f^*} A \\
 \text{Both have a blue oval labeled } f^{\dagger} \text{ and a red oval labeled } f^*. \\
 \text{Left: } \text{fix}_f^{\dagger} \text{ is a 2-cell from } 1 \text{ to } A \text{ through } f^{\dagger}, f, \text{ and } \delta_f. \\
 \text{Right: } \text{fix}_f^* \text{ is a 2-cell from } 1 \text{ to } A \text{ through } f^*, f, \text{ and } \delta_f.
 \end{array}$$

We denote by $\text{Fix}(\mathcal{D})$ the category of pseudo-fixpoint operators on \mathcal{D} .

Dinatural transformations are used to model mixed variance operators using their argument both covariantly and contravariantly. The typical examples arising from semantics occur with (cartesian or monoidal) closed structure where the evaluation map is dinatural and with fixpoint operators. In our setting, we use the notion of pseudo-dinatural transformation for 2-functors.

Definition II.3. A pseudo-dinatural fixpoint operator on \mathcal{D} consists a pseudo-fixpoint operator $((-)^*, \text{fix})$ on \mathcal{D} together with a family of invertible 2-cells

$$\begin{array}{c}
 \text{Diagram showing a 2-cell } (gf)^* \text{ from } 1 \text{ to } A \text{ through } f \text{ and } g. \\
 \text{Below it is } (fg)^* \text{ from } 1 \text{ to } B \text{ through } f \text{ and } g.
 \end{array}$$

for 1-cells $f : A \rightarrow B$ and $g : B \rightarrow A$ in \mathcal{D} satisfying the following axioms:

1) **pseudo-dinaturality axioms:**

a) Unity axiom: for a 1-cell $f : A \rightarrow A$, we have

$$\begin{array}{c}
 \text{Diagram showing a 2-cell } f^* \text{ from } 1 \text{ to } A \text{ through } f \text{ and } 1_A. \\
 \text{Below it is } f^* \text{ from } 1 \text{ to } A \text{ through } f^*.
 \end{array}$$

b) 1-naturality axiom: for all 1-cells $f : A \rightarrow B$, $g : B \rightarrow C$ and $h : C \rightarrow A$ in \mathcal{D} :

$$\begin{array}{c}
 \text{Diagram showing two 2-cell configurations:} \\
 \text{Left: } 1 \xrightarrow{\text{dinat}_{hg}^f} B \quad \text{Right: } 1 \xrightarrow{\text{dinat}_{h}^{gf}} B \\
 \text{Both have a blue oval labeled } (hgf)^* \text{ and a red oval labeled } (gfh)^*. \\
 \text{Left: } \text{dinat}_{hg}^f \text{ is a 2-cell from } 1 \text{ to } B \text{ through } (fhg)^*, f, \text{ and } g. \\
 \text{Right: } \text{dinat}_{h}^{gf} \text{ is a 2-cell from } 1 \text{ to } B \text{ through } (gfh)^*, h, \text{ and } g.
 \end{array}$$

c) 2-naturality axiom: for an invertible 2-cell $\alpha : f \Rightarrow f' : A \rightarrow B$ in \mathcal{D} and a 1-cell $g : B \rightarrow A$ in \mathcal{D} , we have

$$\begin{array}{c}
 \text{Diagram showing two 2-cell configurations:} \\
 \text{Left: } 1 \xrightarrow{\text{dinat}_g^{f'}} B \quad \text{Right: } 1 \xrightarrow{\text{dinat}_g^f} B \\
 \text{Both have a blue oval labeled } (gf')^* \text{ and a red oval labeled } (fg)^*. \\
 \text{Left: } \text{dinat}_g^{f'} \text{ is a 2-cell from } 1 \text{ to } B \text{ through } (f'g)^*, f', \text{ and } \alpha. \\
 \text{Right: } \text{dinat}_g^f \text{ is a 2-cell from } 1 \text{ to } B \text{ through } (fg)^*, f, \text{ and } \alpha.
 \end{array}$$

these three axioms induce a pseudo-dinatural transformation:

$$\text{dinat} : \mathcal{D}(-, -) \rightrightarrows \mathcal{D}(1, -) : \mathcal{D}^{\text{op}} \times \mathcal{D} \rightarrow \text{CAT}$$

2) **coherence between dinat and fix:** for all 1-cells $f : A \rightarrow B$ and $g : B \rightarrow A$ in \mathcal{D} , we have:

$$\begin{array}{ccc}
\text{Diagram showing coherence between dinat and fix:} & & \\
\text{Left: } \begin{array}{c} (fg)^* \rightarrow B \\ \nearrow \text{dinat}_f^g \downarrow g \\ 1 \xrightarrow{(gf)^*} A \\ \nearrow \text{dinat}_g^f \downarrow f \\ (fg)^* \rightarrow B \end{array} & = & \text{Right: } \begin{array}{c} (fg)^* \rightarrow B \\ \downarrow g \\ 1 \xrightarrow{\text{fix}_{fg}} A \\ \downarrow f \\ (fg)^* \rightarrow B \end{array}
\end{array}$$

Remark 3. Note that the coherence axiom between **dinat** and **fix** above implies that

$$\begin{array}{ccc}
\text{Diagram showing coherence between dinat and fix:} & & \\
\text{Left: } \begin{array}{c} f^* \rightarrow A \\ \nearrow \text{dinat}_{1_A}^f \downarrow f \\ 1 \xrightarrow{f^*} A \\ \nearrow f^* \downarrow f \\ f^* \rightarrow A \end{array} & = & \text{Right: } \begin{array}{c} f^* \rightarrow A \\ \downarrow f \\ 1 \xrightarrow{\text{fix}_f} A \\ \downarrow f^* \\ f^* \rightarrow A \end{array}
\end{array}$$

since $\text{dinat}_{1_A}^f = \text{id}_{f^*}$. In the 1-categorical setting, if the axiom for dinaturality $f(gf)^* = (fg)^*$ holds, the fixpoint axiom $ff^* = f^*$ becomes redundant (it suffices to take $g = \text{id}_A$). In the 2-dimensional case, the 2-cells **fix** are entirely determined by the 2-cells **dinat** and we therefore just write $((-)^*, \text{dinat})$ for a pseudo-dinatural fixpoint operator instead of $((-)^*, \text{fix}, \text{dinat})$.

Definition II.4. Let $((-)^*, \text{dinat}^*)$ and $((-)^{\dagger}, \text{dinat}^{\dagger})$ be two pseudo-dinatural fixpoint operators on a 2-category \mathcal{D} . A pseudo-morphism of pseudo-dinatural fixpoint operators $((-)^*, \text{dinat}^*) \rightarrow ((-)^{\dagger}, \text{dinat}^{\dagger})$ consists of a family of natural isomorphisms

$$\delta_A : (-)_A^* \Rightarrow (-)_A^{\dagger} : \mathcal{D}(A, A) \rightarrow \mathcal{D}(1, A)$$

indexed by the objects A of \mathcal{D} that commutes with the structural 2-cells **dinat**, i.e. it satisfies the following coherence for every $f : A \rightarrow B$ and $g : B \rightarrow A$:

$$\begin{array}{ccc}
\text{Diagram showing coherence for dinat and fix:} & & \\
\text{Left: } \begin{array}{c} (gf)^{\dagger} \rightarrow A \\ \nearrow \text{dinat}_g^{f^{\dagger}} \downarrow f \\ 1 \xrightarrow{(fg)^{\dagger}} A \\ \nearrow \delta_{fg} \downarrow f \\ (fg)^* \rightarrow B \end{array} & = & \text{Right: } \begin{array}{c} (gf)^{\dagger} \rightarrow A \\ \nearrow \delta_{gf} \downarrow f \\ 1 \xrightarrow{(fg)^*} A \\ \nearrow \text{dinat}_g^{f^*} \downarrow f \\ (fg)^* \rightarrow B \end{array}
\end{array}$$

We denote by **DinFix**(\mathcal{D}) the category of pseudo-dinatural fixpoint operators on \mathcal{D} .

Dinatural transformations do not compose in general and therefore they do not form a category. In order to make the notion compositional, *strong dinatural transformations* were introduced. It was noted by Mulry that the uniformity axiom for fixpoints (Definition I.2) can be reformulated by requiring $(-)^*$ to be part of a strong dinatural transformation with respect to strict maps [30] and we use this characterization when moving to dimension 2. While generalizations of dinatural

and extranatural transformations for 2-categorical structures have been considered before [31], [32], [33], [34], [35], to our knowledge, there is no existing notion of strong dinatural transformations in dimension 2.

Definition II.5. Let $J : \mathcal{C} \rightarrow \mathcal{D}$ be an identity-on-objects 2-functor (strictly) preserving terminal objects. A pseudo-fixpoint operator on \mathcal{D} that is *uniform with respect to J* consists of a pseudo-fixpoint operator $((-)^*, \text{fix})$ as in Definition II.1 together with a family of 2-cells

$$\begin{array}{c} f^* \rightarrow A \\ \swarrow \text{unif}_{\gamma} \downarrow J_s \\ 1 \xrightarrow{J_s} B \\ \searrow g^* \end{array}$$

for every 1-cells $s : A \rightarrow B$ in \mathcal{C} , $f : A \rightarrow A$ and $g : B \rightarrow B$ in \mathcal{D} and invertible 2-cell γ as below:

$$\begin{array}{c} A \xrightarrow{f} A \\ \swarrow J_s \quad \swarrow \gamma \quad \searrow J_s \\ B \xrightarrow{g} B \end{array}$$

satisfying the following axioms:

1) **strong pseudo-dinaturality:**

a) Unity axiom: we have

$$\begin{array}{c} f^* \rightarrow A \\ \swarrow \text{unif}_{\text{id}_f} \downarrow J_{1_A} \\ 1 \xrightarrow{J_{1_A}} B \\ \searrow f^* \end{array} = \text{id}_{f^*}$$

b) 1-naturality: for two squares in \mathcal{D}

$$\begin{array}{cc} \begin{array}{c} A \xrightarrow{f} A \\ \swarrow J_s \quad \swarrow \gamma \quad \searrow J_s \\ B \xrightarrow{g} B \end{array} & \begin{array}{c} B \xrightarrow{g} B \\ \swarrow J_r \quad \swarrow \rho \quad \searrow J_r \\ C \xrightarrow{h} C \end{array} \end{array}$$

we have:

$$\begin{array}{c} f^* \rightarrow A \\ \swarrow \text{unif}_{\gamma * v \rho} \downarrow J_s \\ 1 \xrightarrow{J_s} B \\ \searrow h^* \\ & = & \begin{array}{c} f^* \rightarrow A \\ \swarrow \text{unif}_{\gamma} \downarrow J_s \\ 1 \xrightarrow{g^*} B \\ \searrow \text{unif}_{\rho} \downarrow J_r \\ h^* \rightarrow C \end{array} \end{array}$$

where $\gamma * v \rho$ denotes the 2-cell corresponding to stacking the two squares vertically as follows:

$$\begin{array}{ccc}
A \xrightarrow{f} A & & \\
Jrs \downarrow \swarrow \gamma *_v \rho \downarrow Jrs & := & \\
C \xrightarrow{h} C & & \\
& & \\
A \xrightarrow{f} A & & \\
Js \downarrow \swarrow \gamma & & Js \downarrow \\
B \xrightarrow{g} B & & \\
Jr \downarrow \swarrow \rho & & Jr \downarrow \\
C \xrightarrow{h} C & &
\end{array}$$

$$\begin{array}{ccc}
g^* & & g^* \\
\text{fix}_g & & \text{unif}_\gamma \\
1 & & 1 \\
f^* & & f^* \\
\text{fix}_f & & \text{unif}_\gamma \\
f & & f \\
g & & g \\
\uparrow \text{unif}_\gamma & & \uparrow \text{unif}_\gamma \\
A \circ J_s & & A \circ J_s \\
\downarrow & & \downarrow \\
A \circ J_s & & A \circ J_s
\end{array}$$

c) 2-naturality: for every invertible 2-cell $\theta : s \Rightarrow r$ in \mathcal{C} such that

$$\begin{array}{ccc}
A \xrightarrow{f} A & & \\
Jr \left(\begin{array}{c} \swarrow J\theta \\ \circ \end{array} \right) Js \swarrow \gamma & = & \\
B \xrightarrow{g} B & & \\
& &
\end{array}$$

we have

$$\begin{array}{ccc}
f^* & & f^* \\
\text{unif}_\gamma & & \text{unif}_\rho \\
1 & & 1 \\
\downarrow & & \downarrow \\
g^* & & g^* \\
\downarrow & & \downarrow \\
A & & A \\
\downarrow & & \downarrow \\
B & & B
\end{array}$$

d) If the following equality holds,

$$\begin{array}{ccc}
A \xrightarrow{f} A & & \\
J_s \left(\begin{array}{c} \swarrow \alpha \\ h \\ \swarrow \rho \end{array} \right) J_s & = & \\
B \xrightarrow{k} B & & \\
& &
\end{array}$$

Definition II.6. Let $((-)^*, \text{fix}^*, \text{unif}^*)$ and $((-)^{\dagger}, \text{fix}^{\dagger}, \text{unif}^{\dagger})$ be two pseudo-fixpoint operators uniform with respect to $J : \mathcal{C} \rightarrow \mathcal{D}$. A pseudo-morphism of uniform fixpoint operators from $((-)^*, \text{fix}^*, \text{unif}^*)$ to $((-)^{\dagger}, \text{fix}^{\dagger}, \text{unif}^{\dagger})$ is a pseudo-morphism of fixpoint operators $\delta : ((-)^*, \text{fix}^*) \rightarrow ((-)^{\dagger}, \text{fix}^{\dagger})$ as in Definition II.2 satisfying the additional coherence for every square in \mathcal{D} :

$$\begin{array}{ccc}
A \xrightarrow{f} A & & \\
J(s) \downarrow \swarrow \gamma & & J(s) \downarrow \\
B \xrightarrow{g} B & &
\end{array}$$

we have:

$$\begin{array}{ccc}
f^* & & f^* \\
\text{unif}_\gamma & & \text{unif}_\rho \\
1 & & 1 \\
\downarrow & & \downarrow \\
g^* & & g^* \\
\downarrow & & \downarrow \\
A & & A \\
\downarrow & & \downarrow \\
B & & B
\end{array}$$

We denote by $\text{Fix}(\mathcal{D}, J)$ the category of pseudo-fixpoint operators on \mathcal{D} uniform with respect to J .

Definition II.7. Let $J : \mathcal{C} \rightarrow \mathcal{D}$ be a identity-on-objects 2-functor (strictly) preserving terminal objects. A *pseudo-dinatural fixpoint operator uniform with respect to J* consists of a pseudo-dinatural fixpoint operator $((-)^*, \text{dinat})$ on \mathcal{D} together with a strong dinatural transformation

$$\text{unif} : \mathcal{D}(J(-), J(-)) \rightrightarrows \mathcal{D}(1, J(-),) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \text{CAT}$$

satisfying the following additional coherence between **dinat** and **unif**: for two squares in \mathcal{D} of the form

$$\begin{array}{ccccc}
A & \xrightarrow{f} & B & \xrightarrow{g} & A \\
Js \downarrow \swarrow \gamma & & Jr \downarrow \swarrow \rho & & Js \downarrow \\
C & \xrightarrow{h} & D & \xrightarrow{k} & C
\end{array}$$

we have:

These four axioms induce a strong pseudo-dinatural transformation:

$$\text{unif} : \mathcal{D}(J(-), J(-)) \rightrightarrows \mathcal{D}(1, J(-),) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \text{CAT}$$

2) **coherence between fix and unif:** for an invertible 2-cell as below

$$\begin{array}{ccc}
A & \xrightarrow{f} & A \\
J(s) \downarrow \swarrow \gamma & & J(s) \downarrow \\
B & \xrightarrow{g} & B
\end{array}$$

we have:

where $\gamma * \rho$ corresponds to the following 2-cell:

$$\gamma * \rho = Jr \begin{array}{c} \xrightarrow{g} A \xrightarrow{f} B \\ \circ \quad \circ \\ \Downarrow \rho \quad Js \\ \downarrow \quad \downarrow \gamma \\ D \xrightarrow{k} C \xrightarrow{h} D \end{array}$$

Note that if we restrict to the case where $A = B, C = D, g = 1_A, k = 1_C, r = s$ and $\rho = \text{id}$, we obtain the coherence axiom between **fix** and **unif** in Definition II.5. Pseudo-fixpoint operators uniform with respect to J are therefore a special case of pseudo-dinatural fixpoint operators uniform with respect to J as expected.

Definition II.8. Let $((-)^*, \text{dinat}^*, \text{unif}^*)$ and $((-)^{\dagger}, \text{dinat}^{\dagger}, \text{unif}^{\dagger})$ be two pseudo-dinatural fixpoint operators uniform with respect to $J : \mathcal{C} \rightarrow \mathcal{D}$. A pseudo-morphism of uniform dinatural fixpoint operators from $((-)^*, \text{dinat}^*, \text{unif}^*)$ to $((-)^{\dagger}, \text{dinat}^{\dagger}, \text{unif}^{\dagger})$ is a pseudo-morphism of dinatural fixpoint operators $\delta : ((-)^*, \text{dinat}^*) \rightarrow ((-)^{\dagger}, \text{dinat}^{\dagger})$ as in Definition II.4 that is also a pseudo-morphism of uniform fixpoint operators as in Definition II.6, i.e. δ commutes with both the dinaturality and uniformity structural 2-cells **dinat** and **unif**.

We denote by **DinFix**(\mathcal{D}, J) the category of pseudo-dinatural fixpoint operators on \mathcal{D} uniform with respect to J .

Before stating the main theorem of the paper, we recall the notion of pseudo-bifree algebras for 2-functors.

Definition II.9. For a 2-functor $! : \mathcal{C} \rightarrow \mathcal{C}$, a *pseudo-initial algebra* is a 1-cell $R : !\Phi \rightarrow \Phi$ such that for every 1-cell $f : !A \rightarrow A$, there exists a pseudo-morphism of algebras $(u_f, \mu_f) : R \rightarrow f$, i.e. a 1-cell $u_f : \Phi \rightarrow A$ and a 2-cell

$$\begin{array}{ccc} !\Phi & \xrightarrow{R} & \Phi \\ !u_f \downarrow \Downarrow \mu_f & & \downarrow u_f \\ !A & \xrightarrow{f} & A \end{array}$$

verifying the following universal property: for any pseudo-algebra 1-cells $(v, \nu), (w, \omega) : R \rightarrow f$, there is a unique invertible algebra 2-cell $\phi : (v, \nu) \Rightarrow (w, \omega)$, i.e. a unique invertible 2-cell $\phi : v \Rightarrow w$ in \mathcal{C} such that:

We can similarly define a dual notion of pseudo-final $!$ -coalgebra. Lambek's theorem stating that an initial algebra or a final coalgebra is an invertible morphism is generalized to an adjoint equivalence:

Lemma II.10 ([18]). *If $R : !\Phi \rightarrow \Phi$ is a pseudo-initial $!$ -algebra, then it is part of an adjoint equivalence $(R : !\Phi \rightarrow \Phi, L : \Phi \rightarrow !\Phi, \eta : \text{id} \xrightarrow{\cong} RL, \varepsilon : LR \xrightarrow{\cong} \text{id})$.*

Definition II.11. We say that $R : !\Phi \rightarrow \Phi$ is a *pseudo-bifree algebra* if R is a pseudo-initial algebra and its (uniquely determined) left adjoint L is a pseudo-final coalgebra.

We can now state the main theorem of this paper which is categorification of Theorem I.3:

Theorem II.12. *Let \mathcal{C} be a 2-category equipped with a (strict) 2-comonad $(!, \delta, \varepsilon)$ and a (strict) terminal object **1**. We denote by \mathcal{D} the co-Kleisli 2-category $\mathcal{C}_!$ and by $J : \mathcal{C} \rightarrow \mathcal{D}$ the free functor induced by the comonadic adjunction.*

- 1) *If the endofunctor $!$ has a pseudo-bifree algebra, then the category **Fix**(\mathcal{D}, J) of pseudo-fixpoint operators on \mathcal{D} uniform with respect to J is contractible.*
- 2) *If \mathcal{C} is cartesian and the endofunctor $!!$ has a pseudo-bifree algebra, then the category **DinFix**(\mathcal{D}, J) of pseudo-dinatural fixpoint operators on \mathcal{D} uniform with respect to J is contractible.*

We proceed to prove this theorem in the next section by first constructing an explicit pseudo-fixpoint operator from the bifree algebras and showing that it verifies the required axioms and then proving the contractibility property i.e. for any other pseudo-fixpoint operator, there is a unique isomorphism between them.

III. THE PLOTKIN-SIMPSON THEOREM FOR 2-CATEGORIES

In this section, we fix a 2-category \mathcal{C} equipped with a (strict) 2-comonad $(!, \delta, \varepsilon)$ and a (strict) terminal object **1**. We denote by \mathcal{D} the co-Kleisli 2-category $\mathcal{C}_!$ and by $J : \mathcal{C} \rightarrow \mathcal{D}$ the free functor induced by the comonadic adjunction. We assume further that the endofunctor $!$ has a pseudo-bifree algebra $R : !\Phi \rightarrow \Phi$.

The following lemma is simply a reformulation of Definition II.9 from \mathcal{C} to the co-Kleisli \mathcal{D} :

Lemma III.1. *For any 1-cell $f : A \rightarrow A$ in \mathcal{D} , there exists a 1-cell $u_f : \Phi \rightarrow A$ in \mathcal{C} and a 2-cell μ_f*

$$\begin{array}{ccc}
\Phi & \xrightarrow{R} & \Phi \\
J(u_f) \downarrow & \swarrow \mu_f & \downarrow J(u_f) \\
A & \xrightarrow{f} & A
\end{array}$$

in \mathcal{D} verifying the following property: for any 1-cells $v, w : \Phi \rightarrow A$ in \mathcal{C} and 2-cells $\nu : J(v)R \Rightarrow fJ(v)$ and $\omega : J(w)R \Rightarrow fJ(w)$ in \mathcal{D} , there exists a unique invertible $\phi : v \Rightarrow w$ in \mathcal{C} such that

$$\begin{array}{ccc}
Jw \left(\begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \swarrow J\phi \\ A \xrightarrow{f} A \end{array} \right) Jv \not\llcorner \nu & = & Jw \left(\begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \swarrow \omega Jw \quad \swarrow J\phi \\ A \xrightarrow{f} A \end{array} \right) Jv
\end{array}$$

The next lemma uses the fact that R is pseudo-bifree and therefore part of an adjoint equivalence where the left adjoint L is pseudo-final.

Lemma III.2. There exists a 1-cell $t : 1 \rightarrow \Phi$ and an invertible 2-cell $\tau : t \Rightarrow Rt$ in \mathcal{D}

$$\begin{array}{ccc}
t & \rightarrow & \Phi \\
1 \not\llcorner \tau & & R \\
& & \downarrow \\
t & \rightarrow & \Phi
\end{array}$$

satisfying the following property: for any 1-cells $v, w : 1 \rightarrow \Phi$ in \mathcal{D} and invertible 2-cells $\nu : v \Rightarrow Rv$ and $\omega : w \Rightarrow Rw$, there exists a unique invertible 2-cell $\psi : v \Rightarrow w$ in \mathcal{D} such that

$$\begin{array}{ccc}
1 \left(\begin{array}{c} w \rightarrow \Phi \\ \swarrow \omega \\ v \rightarrow \Phi \end{array} \right) R & = & 1 \left(\begin{array}{c} w \rightarrow \Phi \\ \swarrow \psi \\ v \rightarrow \Phi \\ \swarrow \nu \end{array} \right) R
\end{array}$$

Using these two lemmas, we can now construct the pseudo-fixpoint operator on \mathcal{D} . We define a family of functors indexed by the objects $A \in \mathcal{D}$

$$(-)^*_A : \mathcal{D}(A, A) \rightarrow \mathcal{D}(1, A)$$

mapping a 1-cell $f : A \rightarrow A$ to

$$f^* := 1 \xrightarrow{t} \Phi \circ \xrightarrow{J(u_f)} A$$

where u_f and t are obtained from Lemmas III.1 and III.2. For a 2-cell $\alpha : f \Rightarrow g$ in $\mathcal{D}(A, A)$, define $\alpha^* : f^* \Rightarrow g^*$ as

$$1 \xrightarrow{t} \Phi \left(\begin{array}{c} \Downarrow J\phi \\ J(u_g) \end{array} \right) A$$

where ϕ is the unique 2-cell $u_f \Rightarrow u_g$ in \mathcal{C} such that

$$\begin{array}{ccc}
J(u_g) \left(\begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \swarrow J\phi \\ A \xrightarrow{f} A \end{array} \right) J(u_f) & = & J(u_g) \not\llcorner \mu_g J(u_g) \left(\begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \swarrow J\phi \\ A \xrightarrow{g} A \end{array} \right) J(u_f)
\end{array}$$

which exists by Lemma III.1.

We can now define the fixpoint 2-cell $\text{fix}_f : ff^* \Rightarrow f^*$ for a 1-cell $f : A \rightarrow A$ in \mathcal{D} as:

$$1 \left(\begin{array}{c} f^* \rightarrow A \\ \not\llcorner \text{fix}_f \\ f \end{array} \right) := 1 \left(\begin{array}{c} t \rightarrow \Phi \circ \xrightarrow{J(u_f)} A \\ \not\llcorner \tau \\ R \not\llcorner \mu_f \\ f \end{array} \right) t \rightarrow \Phi \circ \xrightarrow{J(u_f)} A$$

where μ_f and τ are obtained from Lemmas III.1 and III.2. To obtain that $((-)^*, \text{fix})$ is a pseudo-fixpoint operator on \mathcal{D} , it only remains to show that fix is natural:

Lemma III.3 (Naturality of fix). For a 2-cell $\alpha : f \Rightarrow g$ in \mathcal{D} , we have $\text{fix}_g \alpha^* = (\alpha \cdot \alpha^*) \text{fix}_f$.

Now that we have constructed a pseudo-fixpoint operator on \mathcal{D} , we want to show that it is uniform with respect to the free functor $J : \mathcal{C} \rightarrow \mathcal{D}$ from the base 2-category \mathcal{C} to the co-Kleisli $\mathcal{D} = \mathcal{C}$.

Assume that there exist a 1-cell $s : A \rightarrow B$ in \mathcal{C} and $f : A \rightarrow A$, $g : B \rightarrow B$ in \mathcal{D} and a 2-cell

$$\begin{array}{ccc}
A & \xrightarrow{f} & A \\
J(s) \downarrow & \swarrow \gamma & \downarrow J(s) \\
B & \xrightarrow{g} & B
\end{array}$$

in \mathcal{D} . By Lemma III.1, there exists a unique 2-cell $\phi : su_f \Rightarrow u_g$ in \mathcal{C} such that:

$$\begin{array}{ccc}
J(u_g) \left(\begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \swarrow J\phi \\ A \xrightarrow{f} A \end{array} \right) J(u_f) & = & J(u_g) \not\llcorner \mu_g J(u_g) \left(\begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \swarrow J\phi \\ A \xrightarrow{g} A \end{array} \right) J(u_f)
\end{array}$$

Define unif_γ as:

$$1 \left(\begin{array}{c} f^* \rightarrow A \\ \not\llcorner \text{unif}_\gamma \\ J_s \end{array} \right) := 1 \xrightarrow{t} \Phi \left(\begin{array}{c} \Downarrow J\phi \\ J(u_g) \end{array} \right) J_s \rightarrow B$$

Proposition III.4. *The 2-cells unif_γ we constructed yield a strong pseudo-dinatural transformation and they verify the coherence axiom between fix and unif .*

We have constructed a uniform pseudo-fixpoint operator showing that the category $\text{Fix}(\mathcal{D}, J)$ is inhabited, it remains to show that it is contractible.

Assume that there exists another pseudo-fixpoint operator $((-)^{\dagger}, \text{fix}^{\dagger}, \text{unif}^{\dagger})$ on \mathcal{D} that is uniform with respect to $J : \mathcal{C} \rightarrow \mathcal{D}$. We want to show that there is a unique isomorphism of uniform pseudo-fixpoint operators $\delta : ((-)^* \text{fix}^*, \text{unif}^*) \rightarrow ((-)^{\dagger}, \text{fix}^{\dagger}, \text{unif}^{\dagger})$.

By Lemma III.2, there exists a unique invertible 2-cell $\delta_0 : t \Rightarrow R^{\dagger}$ such that

$$\begin{array}{ccc} \text{Diagram showing } \delta_0 \text{ and fix}_R^{\dagger} & = & \text{Diagram showing } \delta_0, t, \tau \text{ and } R \\ \text{Left: } 1 \xrightarrow{\delta_0} R^{\dagger} \xrightarrow{\text{fix}_R^{\dagger}} \Phi & & \text{Right: } 1 \xrightarrow{\delta_0} t \xrightarrow{\tau} R^{\dagger} \xrightarrow{\Phi} \Phi \end{array}$$

Now, for every 1-cell $f : A \rightarrow A$, there exists by Lemma III.1 a 1-cell $u_f : \Phi \rightarrow A$ in \mathcal{C} and a 2-cell

$$\begin{array}{ccc} \Phi & \xrightarrow{R} & \Phi \\ J(u_f) \circ & \Downarrow \mu_f & \circ J(u_f) \\ A & \xrightarrow{f} & A \end{array}$$

in \mathcal{D} . Since $(-)^{\dagger}$ is uniform with respect to J , we have a 2-cell $\text{unif}_{\mu_f}^{\dagger} : Ju_f R^{\dagger} \Rightarrow f^{\dagger}$ and we define $\delta_f : f^* \Rightarrow f^{\dagger}$ as:

$$\delta_f := 1 \xrightarrow{\text{unif}_{\mu_f}^{\dagger}} R^{\dagger} \xrightarrow{\text{fix}_R^{\dagger}} A \quad \text{and} \quad 1 \xrightarrow{\delta_0} R^{\dagger} \xrightarrow{J(u_f)} \Phi \xrightarrow{f^{\dagger}} B$$

Proposition III.5. *The category $\text{Fix}(\mathcal{D}, J)$ is contractible i.e. δ is an isomorphism of uniform pseudo-fixpoint operators and it is unique.*

IV. DINATURALITY

To obtain that the pseudo-fixpoint operator is pseudo-dinatural, we want to construct an invertible dinaturality 2-cell for every 1-cell $f : A \rightarrow B$ in \mathcal{D} ,

$$\begin{array}{ccccc} \mathcal{D}(f, A) & \xrightarrow{\mathcal{D}(A, A)} & (-)_A^* & \xrightarrow{\mathcal{D}(1, A)} & \mathcal{D}(1, f) \\ \mathcal{D}(B, A) & \xrightarrow{\text{dinat}^f} & & & \\ \mathcal{D}(B, f) & \xrightarrow{\mathcal{D}(B, B)} & (-)_B^* & \xrightarrow{\mathcal{D}(1, B)} & \end{array}$$

with components

$$\begin{array}{ccc} (gf)^* & \xrightarrow{A} & \\ 1 \not\Rightarrow \text{dinat}_g^f & \downarrow f & \\ (fg)^* & \xrightarrow{B} & \end{array}$$

for g in $\mathcal{D}(B, A)$. To do so, we need to assume further that the endofunctor $!! : \mathcal{C} \rightarrow \mathcal{C}$ has a pseudo-bifree algebra and that 2-category \mathcal{C} is cartesian. Note that it implies that co-Kleisli \mathcal{D} is cartesian as well and that the free functor $J : \mathcal{C} \rightarrow \mathcal{D}$ preserves the cartesian structure.

The following lemma uses the same argument as the strict case by Freyd [26].

Lemma IV.1. *Assume that $!! : \mathcal{C} \rightarrow \mathcal{C}$ has a pseudo bifree algebra. Then, if $R : !\Phi \rightarrow \Phi$ is a pseudo bifree $!$ -algebra,*

$$!!\Phi \xrightarrow{!R} !\Phi \xrightarrow{R} \Phi$$

is a pseudo bifree $!!$ -algebra.

Lemma IV.2. *For 1-cells $v, w : 1 \rightarrow \Phi$ in \mathcal{D} and 2-cells*

$$\begin{array}{ccc} 1 \xrightarrow{v} \Phi & \text{and} & 1 \xrightarrow{w} \Phi \\ \not\Rightarrow \nu & & \not\Rightarrow \omega \\ v & \xrightarrow{RR} & w \\ & \xrightarrow{\Phi} & \end{array}$$

in \mathcal{D} , there exists a unique invertible 2-cell $\lambda : w \Rightarrow v$ in \mathcal{D} such that

$$\begin{array}{ccc} 1 \xrightarrow{v} \Phi & = & 1 \xrightarrow{w} \Phi \\ \not\Rightarrow \nu & & \not\Rightarrow \omega \\ v & \xrightarrow{RR} & w \\ \not\Rightarrow \lambda & & \xrightarrow{RR} \end{array}$$

To construct the dinaturality 2-cells, we first start by constructing a 2-cell $\text{din} : t \Rightarrow (RR)^*$. Let din be the unique invertible 2-cell from t to $J(u_{RR})t$ (obtained from Lemma IV.2) such that:

$$\begin{array}{ccc} \text{Left: } 1 \xrightarrow{t} \Phi & \xrightarrow{R} & \Phi \\ \not\Rightarrow \tau & & \\ \text{Middle: } 1 \xrightarrow{t} \Phi & \xrightarrow{J(u_{RR})} & \Phi \\ \not\Rightarrow \text{din} & & \\ \text{Right: } 1 \xrightarrow{t} \Phi & \xrightarrow{R} & \Phi \\ \not\Rightarrow \tau & & \end{array} = \begin{array}{ccc} \text{Left: } 1 \xrightarrow{t} \Phi & \xrightarrow{R} & \Phi \\ \not\Rightarrow \text{din} & & \\ \text{Middle: } 1 \xrightarrow{t} \Phi & \xrightarrow{R} & \Phi \\ \not\Rightarrow \tau & & \\ \text{Right: } 1 \xrightarrow{t} \Phi & \xrightarrow{R} & \Phi \\ \not\Rightarrow \tau & & \end{array}$$

Once this isomorphism is obtained, we construct the dinaturality 2-cells dinat_g^f using uniformity. Consider the following endo-1-cell:

$$A \times B \xrightarrow{f \times g} B \times A \xrightarrow{\sigma} A \times B$$

where σ is the symmetry obtained in the standard way as the pairing $\langle \pi_1, \pi_2 \rangle$. Using Lemma III.1, there is a 1-cell $u_{\sigma(f \times g)} : \Phi \rightarrow A \times B$ in \mathcal{C} and a 2-cell $\mu_{\sigma(f \times g)}$:

$$\begin{array}{ccc} & R & \\ \Phi & \xrightarrow{\quad} & \Phi \\ \downarrow J(u_{\sigma(f \times g)}) & \swarrow \mu_{\sigma(f \times g)} & \downarrow J(u_{\sigma(f \times g)}) \\ A & \xrightarrow{f} & A \end{array}$$

From the two squares

$$\begin{aligned} \Pi_{\sigma(f \times g)}^1 &:= \begin{array}{ccccc} & R & R & R & \\ \Phi & \xrightarrow{\quad} & \Phi & \xrightarrow{\quad} & \Phi \\ \downarrow J u_{\sigma(f \times g)} & \swarrow \mu_{\sigma(f \times g)} & \downarrow J u_{\sigma(f \times g)} & \swarrow \mu_{\sigma(f \times g)} & \downarrow J u_{\sigma(f \times g)} \\ A \times B & \xrightarrow{\sigma(f \times g)} & A \times B & \xrightarrow{\sigma(f \times g)} & A \times B \\ \downarrow \pi_1 & & \downarrow \pi_2 & & \downarrow \pi_1 \\ A & \xrightarrow{f} & B & \xrightarrow{g} & A \end{array} \\ \Pi_{\sigma(f \times g)}^2 &:= \begin{array}{ccccc} & R & R & R & \\ \Phi & \xrightarrow{\quad} & \Phi & \xrightarrow{\quad} & \Phi \\ \downarrow J u_{\sigma(f \times g)} & \swarrow \mu_{\sigma(f \times g)} & \downarrow J u_{\sigma(f \times g)} & \swarrow \mu_{\sigma(f \times g)} & \downarrow J u_{\sigma(f \times g)} \\ A \times B & \xrightarrow{\sigma(f \times g)} & A \times B & \xrightarrow{\sigma(f \times g)} & A \times B \\ \downarrow \pi_2 & & \downarrow \pi_1 & & \downarrow \pi_2 \\ B & \xrightarrow{g} & A & \xrightarrow{f} & B \end{array} \end{aligned}$$

we obtain uniformity 2-cells

$$\begin{array}{ccc} (RR)^* & \rightarrow \Phi & (RR)^* & \rightarrow \Phi \\ \text{1} \swarrow \text{unif}_{\Pi_{\sigma(f \times g)}^1} & \downarrow J u_{\sigma(f \times g)} & \text{1} \swarrow \text{unif}_{\Pi_{\sigma(f \times g)}^2} & \downarrow J u_{\sigma(f \times g)} \\ A \times B & \xrightarrow{\quad} & A \times B & \xrightarrow{\quad} \\ \downarrow J \pi_1 & & \downarrow J \pi_2 & \\ (gf)^* & \rightarrow A & (fg)^* & \rightarrow B \end{array}$$

We can now construct the general dinaturality 2-cell dinat_g^f as in Figure 1.

Note that for **fix** and **unif**, we wrote the direction of 2-cells without needing to take inverses whereas for dinaturality, a back-and-forth is needed to construct the 2-cells. It means

that for the general directed case, we need both the lax and oplax directions in order to obtain dinaturality.

Proposition IV.3. *The 2-cells dinat_g^f we constructed yield a pseudo-dinatural transformation and they verify the coherence axiom between **dinat** and **unif**.*

We have now shown that the category of uniform pseudo-dinatural fixpoint operators $\text{DinFix}(\mathcal{D}, J)$ is inhabited, it remains to show that it is contractible to complete the proof of Theorem II.12.

Assume that there is another uniform pseudo-dinatural fixpoint operators $((-)^{\dagger}, \text{dinat}^{\dagger}, \text{unif}^{\dagger})$. By Proposition III.5, we already have a constructed morphism of uniform pseudo-fixpoint operators $\delta : ((-)^*, \text{fix}^*, \text{unif}^*) \rightarrow ((-)^{\dagger}, \text{fix}^{\dagger}, \text{unif}^{\dagger})$ where the 2-cells **fix** are induced from the 2-cells **dinat**. Therefore, it only remains to show:

Proposition IV.4. *The category $\text{DinFix}(\mathcal{D}, J)$ is contractible i.e. δ commutes with the dinaturality 2-cells and it is unique.*

V. EXAMPLES

We start by presenting examples where the pseudo-fixpoint operators are obtained as instances of Theorem II.12 and therefore for which the category of pseudo-fixpoint operators is contractible. In particular, the least and greatest fixpoint (i.e. the initial and final object of the category of pseudo-fixpoints) are isomorphic. We consider afterwards the example of polynomial functors which are not an instance of our theorem but where the axioms for pseudo-fixpoints presented in Section II are verified.

A. Limit-colimit coincidence theorem for 2-categorical structures

We give a brief reminder of the general recipe to obtain pseudo-bifree algebras for endofunctors on 2-categories using the machinery developed by Cattani, Fiore and Winskel [16], [17], [18]. Instead of considering preorder-enriched categories, we move to 2-categories or bicategories whose hom-categories have colimits of ω -chains and initial objects. The colimits of ω -chains of embedding-projection pairs in the preorder-enriched setting are generalized to pseudo-colimits of ω -chains of co-reflections (adjunctions with invertible unit) in the categorified setting.

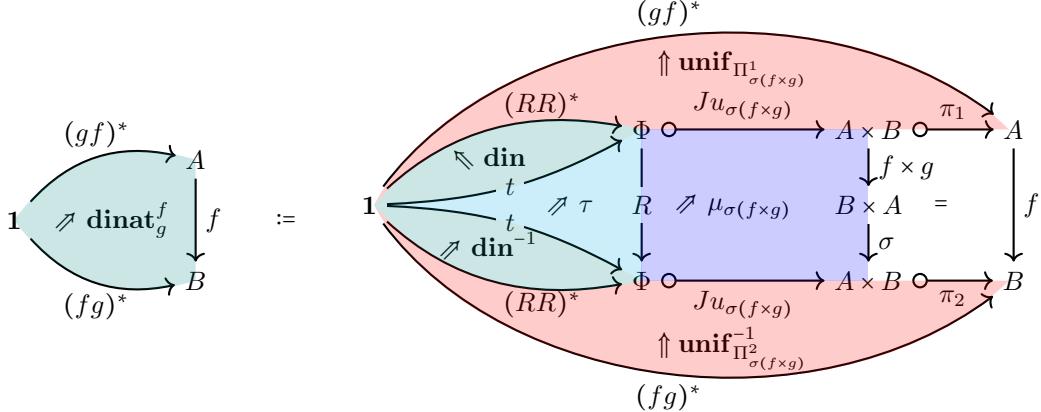
We say that a 2-category or bicategory \mathcal{C} is **Cat** $_{\omega}$ -enriched (**Cat** $_{\omega, \perp}$ -enriched) if for all objects A and B , its hom-category $\mathcal{C}(A, B)$ have colimits of ω -chains (and initial objects) and whose composition functors

$$\mathcal{C}(A, B) \times \mathcal{C}(B, C) \rightarrow \mathcal{C}(A, C)$$

preserve of colimits ω -chains (and initial objects) in both variables. The analogue of the category **Cppo** is the 2-category **Cat** $_{\omega}$ given by

- 0-cells: categories with colimits of ω -chains and initial objects;
- 1-cells: functors preserving colimits of ω -chains;

Figure 1. Construction of the dinaturality 2-cells



- 2-cells: natural transformations.

and its sub-2-category $\mathbf{Cat}_{\omega,\perp}$ which restricts the 1-morphisms to those preserving initial objects (corresponding to the category \mathbf{Cpo}_\perp in the preorder setting).

In order to obtain pseudo-bifree algebras, we make use of the following theorem:

Theorem V.1 ([18]). *Let \mathcal{C} be a $\text{Cat}_{\omega, \perp}$ -enriched 2-category. If \mathcal{C} has a pseudo-initial object and pseudo- ω -colimits of chains of coreflections (adjunctions with invertible units), then \mathcal{C} is Cat_ω -pseudo-algebraically compact. It means that for every pseudo-functor $T : \mathcal{C} \rightarrow \mathcal{C}$, if T is Cat_ω -enriched, i.e. for all A, B , the induced functor*

$$\mathcal{C}(A, B) \rightarrow \mathcal{C}(TA, TB)$$

preserves colimits of ω -chains, then T has a pseudo-bifree algebra.

B. Categorical domain theory

The canonical example is the 2-category $\mathbf{Cat}_{\omega, \perp}$ with the lifting 2-comonad $(-)_\perp : \mathbf{Cat}_{\omega, \perp} \rightarrow \mathbf{Cat}_{\omega, \perp}$ which freely adjoins initial objects. Both $(-)_\perp$ and $(-)_\perp(-)_\perp$ are \mathbf{Cat}_ω -enriched and therefore the coKleisli 2-category (which is equivalent to \mathbf{Cat}_ω) verifies that its category of pseudodinatural fixpoint operators uniform (with respect to the free functor $\mathbf{Cat}_{\omega, \perp} \rightarrow \mathbf{Cat}_\omega$) is contractible. The standard Lambek construction for initial algebras for a finitary endofunctor $f : \mathbb{A} \rightarrow \mathbb{A}$ on a category \mathbb{A} with an initial object \perp and ω -colimits by calculating the colimit of the diagram

$$\perp \rightarrow f(\perp) \rightarrow f^2(\perp) \rightarrow \dots$$

provides both the initial and final pseudo-fixpoint operators [36]. As a consequence of Theorem II.12, Lambek's construction verifies the dinaturality axioms and is uniform with respect to sub-2-category of functors preserving initial objects up to isomorphism.

We can also recover Adamek's result for *Scott-complete categories* which can be viewed as a categorification of Scott domains [14]. He considers the 2-category **SCC** given by

- 0-cells: Scott-complete categories (i.e finitely accessible category such that every diagram with a cocone has a colimit);
 - 1-cells: functors preserving directed colimits;
 - 2-cells: natural transformations.

and its sub-2-category SCC_\perp whose 1-cells are restricted to functors preserving directed colimits and initial objects. The lifting 2-comonad on SCC_\perp has the required bifree algebras so that the co-Kleisli SCC has a contractible category of pseudo-dinatural fixpoint operators uniform with respect to the free 2-functor $\text{SCC}_\perp \rightarrow \text{SCC}$.

C. Profunctors and linear logic models

Another example of $\text{Cat}_{\omega,\perp}$ -enriched bicategory is the bicategory of profunctors denoted by **Prof** [37]. For small categories \mathbb{A} and \mathbb{B} , a profunctor P from \mathbb{A} to \mathbb{B} is a functor $P : \mathbb{A} \times \mathbb{B}^{\text{op}} \rightarrow \text{Set}$ or equivalently a functor from \mathbb{A} to the presheaf category $\hat{\mathbb{B}}$. Profunctors can be seen as a categorification of **Rel** as a relation $R \subseteq A \times B$ corresponds to a profunctor between discrete categories such that each component is either the empty set or a singleton.

Many pseudo-comonad structures were considered on **Prof** leading to models of linear logic with various notion of substitution [3], [4], [38]. In particular, the free symmetric monoidal completion comonad on **Prof** yields the model of *generalized species of structures* which encompasses Joyal's combinatorial species and is also a categorification of the relational model of linear logic [4]. The morphisms in the co-Kleisli bicategory correspond to the notion of *analytic functors* which are generalized power series with quotients [39]. We can also consider the pseudo-comonads freely adjoining finite coproducts or finite colimits to generalize the category **Lin** with the \vee -semi-lattice comonad in Section I or simply the comonad freely adjoining an initial object. We refer the reader

to [40] for a general treatment of pseudo-(co)-monads on **Prof** and to [9] for other examples with applications to intersection typing systems.

The pseudo-comonads we consider verify the necessary conditions on bifree algebras as they are all Cat_ω -enriched and Theorem V.1 has been extended to bicategories and pseudo-functors [18]. For the colimit-completion cases, it is a straightforward consequence of the commutation of colimits and it only needs to be checked by hand for the free symmetric strict monoidal case.

Strictly speaking, we have only provided the notion of pseudo-fixpoint operators for 2-categories and not for bicategories. Even if we strictify the bicategory **Prof** to its biequivalent 2-category **Cocont** (concontinuous functors between presheaf categories), the comonads we consider are pseudo and their corresponding co-Kleisli are therefore bicategories and not 2-categories.

For space considerations, we do not give the proof of Theorem II.12 for bicategories in this paper but only state that we obtain as a corollary that species with the free symmetric strict monoidal completion, initial object, finite coproduct, finite colimit pseudo-comonads are all instances of this construction and therefore all have a contractible category of pseudo-dinatural fixpoint operators uniform with respect to the inclusion of profunctors into these generalized species.

D. Polynomial functors

Initial algebras (well-founded trees) and final coalgebras (non-well-founded trees) for polynomial functors have been extensively studied [41], [42], [43], [44] and are standard tools to model (co)inductive types in dependent type theories. If we fix a locally cartesian closed category \mathbb{C} , for objects I and J in \mathbb{C} a polynomial from I to J is a diagram of shape

$$\begin{array}{ccc} & E & \\ s \swarrow & \xrightarrow{p} & \searrow t \\ I & & J \end{array}$$

in the category \mathbb{C} . It induces a polynomial functor $\mathbb{C}/I \rightarrow \mathbb{C}/J$ between the slice categories which form a 2-category with 2-cells given by cartesian transformations. We say that \mathbb{C} has **W**-types if all polynomial endofunctors have initial algebras and that it has **M**-types if they have final coalgebras.

It is well-known that **W**-types and **M**-types do not coincide and they are therefore not an instance of the contractibility property of Theorem II.12. This example is therefore an adequate test to verify that the axioms we stated in Section II are not just valid in the restricted contractible case but provide a general notion of pseudo-fixpoint operators.

The 2-naturality axiom of the operator computing initial algebras (or final coalgebras) is well known and is sometimes called the “functoriality property”. Dinaturality for the initial algebra or final coalgebra operators is also known in the literature [26]. The statement is that for functors $F : \mathbb{A} \rightarrow \mathbb{B}$, $G : \mathbb{B} \rightarrow \mathbb{A}$ such that GF and FG have initial algebras, if $GFA \xrightarrow{a} A$ is GF -initial, then $FGFA \xrightarrow{Fa} FA$ is FG -initial. To our knowledge, the axioms of a pseudo-dinatural

transformation, while straightforward to check, have not been stated explicitly.

Uniformity on the other hand depends on how the initial algebras (or final coalgebras) are computed in the ambient 2-category. In general, if initial algebras are obtained as colimit constructions, then natural candidates for the 2-category of strict maps are functors preserving initial objects or cocontinuous functors whenever these notions are well-defined. Dually, if the pseudo-fixpoint operator is obtained by computing final coalgebras as certain limits then we can consider terminal object preserving functors or continuous functors as strict maps. In the case of polynomial functors over **Set**, **W**-types are uniform with respect to spans *i.e.* polynomial functors of shape

$$\begin{array}{ccc} & E & \xrightarrow{\cong} B \\ s \swarrow & & \searrow t \\ I & & J \end{array}$$

and **M**-types are uniform with respect to monomials which are polynomial functors of shape

$$\begin{array}{ccc} & E & \xrightarrow{!} 1 \\ s \swarrow & & \searrow t \\ I & & J \end{array}$$

where 1 is a singleton set.

CONCLUSION

We have presented the theory of fixpoint operators for 2-categories using a categorification of Plotkin-Simpson’s theorem as a guideline to derive the equations on the structural 2-cells in dimension 2. The concrete 2-categorical examples allow us to confirm that these equations are verified.

In future work, we aim to extend our theory to *parametrized* and *guarded* fixpoint operators. Adding parameters allows to consider richer contexts for terms and guardedness restricts the possible infinite behavior of fixpoints to ensure properties such as solvability or productivity are satisfied. Since a parametrized fixpoint operator verifying the Conway axioms is equivalent to a *traced monoidal category* with the cartesian product as the chosen tensor, we aim to use our formalism to develop the theory of traced monoidal bicategories and establish new connections with cyclic λ -calculi [45], [46]. We also want to formulate the theory of 2-dimensional fixpoints in a cartesian closed framework where the fixpoint operator is expressed internally as a family of 1-cells of type $(A \Rightarrow A) \rightarrow A$ bringing us closer to the intuition of fixpoint combinators for λ -calculus.

ACKNOWLEDGMENTS

I am grateful to the anonymous reviewers for their comments and to Marcelo Fiore, Nicola Gambino and Martin Hyland for valuable discussions on this work which was supported by EPSRC grant EP/V002309/1 and by the RealiSe PGRM Emergence project.

REFERENCES

- [1] A. Simpson and G. Plotkin, “Complete axioms for categorical fixed-point operators,” in *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 99CB36332)*. IEEE, 2000, pp. 30–41.
- [2] M. P. Fiore, *Axiomatic domain theory in categories of partial maps*. Cambridge University Press, 2004, vol. 14.
- [3] G. L. Cattani and G. Winskel, “Profunctors, open maps and bisimulation,” *Mathematical Structures in Computer Science*, vol. 15, no. 3, pp. 553–614, 2005.
- [4] M. Fiore, N. Gambino, M. Hyland, and G. Winskel, “The cartesian closed bicategory of generalised species of structures,” *J. Lond. Math. Soc.*, (2), vol. 77, no. 1, pp. 203–220, 2008. [Online]. Available: <http://dx.doi.org/10.1112/jlms/jdm096>
- [5] D. Mazza, L. Pellissier, and P. Vial, “Polyadic approximations, fibrations and intersection types,” *Proceedings of the ACM on Programming Languages*, vol. 2, no. POPL:6, 2018.
- [6] M. Fiore and P. Saville, “A type theory for cartesian closed bicategories,” in *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2019, pp. 1–13.
- [7] P.-A. Mellies, “Template games and differential linear logic,” in *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019, pp. 1–13.
- [8] M. Fiore and P. Saville, “Coherence and normalisation-by-evaluation for bicategorical cartesian closed structure,” in *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, 2020, pp. 425–439.
- [9] F. Olimpieri, “Intersection type distributors,” in *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 2021, pp. 1–15. [Online]. Available: <https://doi.org/10.1109/LICS52264.2021.9470617>
- [10] A. Kerinec, G. Manzonetto, and F. Olimpieri, “Why are proofs relevant in proof-relevant models?” *PACMPL*, vol. 7, no. POPL, pp. 8:1–8:31, 2023. [Online]. Available: <https://doi.org/10.1145/3571201>
- [11] D. Scott and C. Strachey, *Toward a mathematical semantics for computer languages*. Oxford University Computing Laboratory, Programming Research Group Oxford, 1971, vol. 1.
- [12] J. Lambek, “A fixpoint theorem for complete categories.” *Mathematische Zeitschrift*, vol. 103, pp. 151–161, 1968. [Online]. Available: <http://eudml.org/doc/170906>
- [13] D. J. Lehmann and M. B. Smyth, “Igebraic specification of data types: A synthetic approach,” *Mathematical systems theory*, vol. 14, no. 1, pp. 97–139, 1981.
- [14] J. Adámek, “A categorical generalization of scott domains,” *Mathematical Structures in Computer Science*, vol. 7, no. 5, p. 419–443, 1997.
- [15] P. Taylor, “The limit-colimit coincidence for categories,” *unpublished preprint, Imperial College, London*, 1988.
- [16] M. P. Fiore, *Axiomatic domain theory in categories of partial maps*. Cambridge University Press, 2004, vol. 14.
- [17] G. L. Cattani, M. Fiore, and G. Winskel, “A theory of recursive domains with applications to concurrency,” in *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, ser. LICS ’98. USA: IEEE Computer Society, 1998, p. 214.
- [18] G. L. Cattani and M. P. Fiore, “The bicategory-theoretic solution of recursive domain equations,” *Electronic Notes in Theoretical Computer Science*, vol. 172, pp. 203–222, 2007.
- [19] A. Pitts, “An elementary calculus of approximations,” 1987.
- [20] K. A. Ponto, *Fixed point theory and trace for bicategories*. The University of Chicago, 2007.
- [21] K. Ponto and M. Shulman, “Shadows and traces in bicategories,” *Journal of Homotopy and Related Structures*, vol. 8, no. 2, pp. 151–200, 2013.
- [22] A. K. Simpson, “A characterisation of the least-fixed-point operator by dinaturality,” *Theoretical Computer Science*, vol. 118, no. 2, pp. 301–314, 1993.
- [23] A. Joyal, R. Street, and D. Verity, “Traced monoidal categories,” in *Mathematical proceedings of the cambridge philosophical society*, vol. 119, no. 3. Cambridge University Press, 1996, pp. 447–468.
- [24] S. Eilenberg, “The category \mathcal{C} .”
- [25] G. Plotkin, “Domains. pisa notes, 1983,” *University of Edinburgh*.
- [26] P. Freyd, “Algebraically complete categories,” in *Category Theory*. Springer, 1991, pp. 95–104.
- [27] M. B. Smyth and G. D. Plotkin, “The category-theoretic solution of recursive domain equations,” *SIAM Journal on Computing*, vol. 11, no. 4, pp. 761–783, 1982.
- [28] J. Girard, “Linear logic,” *Theor. Comput. Sci.*, vol. 50, pp. 1–102, 1987. [Online]. Available: [https://doi.org/10.1016/0304-3975\(87\)90045-4](https://doi.org/10.1016/0304-3975(87)90045-4)
- [29] C. Grellois and P.-A. Mellies, “An infinitary model of linear logic,” in *International Conference on Foundations of Software Science and Computation Structures*. Springer, 2015, pp. 41–55.
- [30] P. S. Mulry, “Strong monads, algebras and fixed points,” *Applications of Categories in Computer Science*, vol. 177, pp. 202–216, 1992.
- [31] S. Bozapalides, *Théorie formelle des bicatégories*. Ehresmann, Bastiani, 1976.
- [32] ———, “Les fins cartésiennes généralisées,” *Archivum Mathematicum*, vol. 13, no. 2, pp. 75–87, 1977.
- [33] J. Climent Vidal and J. Soliveres Tur, “A 2-categorial generalization of the concept of institution,” *Studia Logica*, vol. 95, no. 3, pp. 301–344, 2010.
- [34] A. S. Corner, “A universal characterisation of codescent objects,” *arXiv preprint arXiv:1709.01332*, 2017.
- [35] K. Hirata, “Notes on lax ends,” *arXiv preprint arXiv:2210.01522*, 2022.
- [36] J. Lambek, “A fixpoint theorem for complete categories,” *Mathematische Zeitschrift*, vol. 103, no. 2, pp. 151–161, 1968.
- [37] J. Bénabou, “Distributors at work,” 2000, lecture notes written by Thomas Streicher. [Online]. Available: <https://www2.mathematik.tu-darmstadt.de/~streicher/FIBR/DiWo.pdf>
- [38] Z. Galal, “A profunctorial Scott semantics,” in *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- [39] M. Fiore, “Analytic functors between presheaf categories over groupoids,” *Theoretical Computer Science*, vol. 546, pp. 120–131, 2014.
- [40] M. Fiore, N. Gambino, M. Hyland, and G. Winskel, “Relative pseudomonads, kleisli bicategories, and substitution monoidal structures,” *Selecta Mathematica*, vol. 24, no. 3, pp. 2791–2830, 2018.
- [41] I. Moerdijk and E. Palmgren, “Wellfounded trees in categories,” *Annals of Pure and Applied Logic*, vol. 104, no. 1-3, pp. 189–218, 2000.
- [42] P. Aczel, J. Adámek, and J. Velebil, “A coalgebraic view of infinite trees and iteration,” *Electronic Notes in Theoretical Computer Science*, vol. 44, no. 1, pp. 1–26, 2001.
- [43] N. Gambino and M. Hyland, “Wellfounded trees and dependent polynomial functors,” in *International Workshop on Types for Proofs and Programs*. Springer, 2004, pp. 210–225.
- [44] B. van den Berg and F. De Marchi, “Non-well-founded trees in categories,” *Annals of Pure and Applied Logic*, vol. 146, no. 1, pp. 40–59, 2007.
- [45] M. Hasegawa, “Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi,” in *International Conference on Typed Lambda Calculi and Applications*. Springer, 1997, pp. 196–213.
- [46] N. Benton and M. Hyland, “Traced premonoidal categories,” *RAIRO-Theoretical Informatics and Applications*, vol. 37, no. 4, pp. 273–299, 2003.

A. Dinatural and strong dinatural transformations for 2-categories

We start by recalling the 1-categorical notions of dinatural and strong dinatural transformations and proceed with the 2-categorical generalizations.

1) Dinatural transformations:

Definition A.1. For categories \mathbb{C}, \mathbb{D} and functors $F, G : \mathbb{C}^{\text{op}} \times \mathbb{C} \rightarrow \mathbb{D}$, a *dinatural transformation* $\theta : F \rightrightarrows G$ consists of a family of 1-cells $\{\theta_c : F(c, c) \rightarrow G(c, c)\}_{c \in \mathcal{C}}$ indexed by the objects c in \mathbb{C} such that for every morphism $f : c \rightarrow d$ in \mathbb{C} , the following hexagon commutes:

$$\begin{array}{ccccc} & F(f, c) & \nearrow & \theta_c & \\ F(d, c) & \swarrow & F(c, c) & \xrightarrow{\quad} & G(c, c) \\ & F(d, f) & \nearrow & & G(c, f) \\ & & F(d, d) & \xrightarrow{\quad} & G(d, d) \\ & & \theta_d & \swarrow & G(f, d) \end{array}$$

Definition A.2. For 2-categories \mathcal{C}, \mathcal{D} and 2-functors $F, G : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$, a *lax dinatural transformation* $\theta : F \rightrightarrows G$ consists of:

- a family of 1-cells $\{\theta_c : F(c, c) \rightarrow G(c, c)\}_{c \in \mathcal{C}}$ indexed by the objects c in \mathcal{C} ;
- for every 1-cell $f : c \rightarrow d$ in \mathcal{C} , a 2-cell:

$$\begin{array}{ccccc} & F(f, c) & \nearrow & \theta_c & \\ F(d, c) & \swarrow & F(c, c) & \xrightarrow{\quad} & G(c, c) \\ & \Downarrow \theta_f & & & G(c, d) \\ & F(d, f) & \nearrow & & G(c, f) \\ & & F(d, d) & \xrightarrow{\quad} & G(d, d) \\ & & \theta_d & \swarrow & G(f, d) \end{array}$$

satisfying the following axioms:

- 1) *unity*: for every object $c \in \mathcal{C}$, $\theta_{1_c} = \text{id}_{\theta_c}$,
- 2) *1-naturality*: for every 1-cells $f : c \rightarrow d$ and $g : d \rightarrow e$ in \mathcal{C} ,

$$\begin{array}{ccccccc} \theta_{gf} & = & F(e, c) & = & F(d, c) & = & G(c, e) \\ & & \swarrow F(g, c) & & \swarrow F(d, f) & & \swarrow G(c, g) \\ & & F(e, d) & & F(d, d) & & G(d, e) \\ & & \swarrow F(e, f) & & \swarrow \theta_d & & \swarrow G(d, g) \\ & & F(e, e) & & G(d, d) & & G(d, e) \\ & & \swarrow F(e, g) & & \swarrow G(d, g) & & \swarrow G(d, e) \\ & & \theta_e & & \theta_g & & \theta_e \end{array}$$

- 3) *2-naturality*: for every 2-cell $\alpha : f \Rightarrow f'$ in \mathcal{C} :

$$\begin{array}{ccc}
\begin{array}{c}
\text{Diagram showing a hexagon of 2-cells for a lax dinatural transformation } \theta : F \rightrightarrows G \\
\text{Top row: } F(c,c) \xrightarrow{\theta_c} G(c,c) \\
\text{Bottom row: } F(d,d) \xrightarrow{\theta_d} G(d,d) \\
\text{Left column: } F(c,c) \xrightarrow{F(f,c)} F(d,c) \xrightarrow{F(d,f')} F(d,d) \\
\text{Right column: } G(c,c) \xrightarrow{G(c,f)} G(c,d) \xrightarrow{G(f',d)} G(d,d) \\
\text{Diagonal: } F(d,c) \xrightarrow{F(d,f)} F(d,d) \quad G(c,d) \xrightarrow{G(f,d)} G(d,d) \\
\text{Vertical: } F(d,c) \xrightarrow{F(d,f')} F(d,d) \quad G(c,d) \xrightarrow{G(f',d)} G(d,d)
\end{array} & = &
\begin{array}{c}
\text{Diagram showing a hexagon of 2-cells for a strict dinatural transformation } \theta : F \rightrightarrows G \\
\text{Top row: } F(c,c) \xrightarrow{\theta_c} G(c,c) \\
\text{Bottom row: } F(d,d) \xrightarrow{\theta_d} G(d,d) \\
\text{Left column: } F(c,c) \xrightarrow{F(f,c)} F(d,c) \xrightarrow{F(d,f')} F(d,d) \\
\text{Right column: } G(c,c) \xrightarrow{G(c,f)} G(c,d) \xrightarrow{G(f',d)} G(d,d) \\
\text{Diagonal: } F(d,c) \xrightarrow{F(d,f)} F(d,d) \quad G(c,d) \xrightarrow{G(f,d)} G(d,d) \\
\text{Vertical: } F(d,c) \xrightarrow{F(d,f')} F(d,d) \quad G(c,d) \xrightarrow{G(f',d)} G(d,d)
\end{array}
\end{array}$$

For an *oplax dinatural transformation*, the 2-cells θ_f go in the opposite direction. When the 2-cells θ_f are invertible, we obtain the notion of *pseudo dinatural transformation* and when they are strict identities, we obtain *strict dinatural transformations*.

Definition A.3. A modification between lax dinatural transformation $\Phi : \theta \Rightarrow \delta : F \rightrightarrows G : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$ consists of a family of 2-cells $\{\Phi_c : \theta_c \Rightarrow \delta_c\}_{c \in \mathcal{C}}$ such that for every 1-cell $f : c \rightarrow d$ in \mathcal{C} the following equality holds:

$$\begin{array}{ccc}
\begin{array}{c}
\text{Diagram showing a hexagon of 2-cells for a modification } \Phi \\
\text{Top row: } F(c,c) \xrightarrow{\theta_c} G(c,c) \xrightarrow{G(c,f)} G(c,d) \\
\text{Bottom row: } F(d,d) \xrightarrow{\theta_d} G(d,d) \xrightarrow{G(f,d)} G(c,d) \\
\text{Left column: } F(c,c) \xrightarrow{F(f,c)} F(d,c) \xrightarrow{F(d,f')} F(d,d) \\
\text{Right column: } G(c,c) \xrightarrow{G(c,f)} G(c,d) \xrightarrow{G(f',d)} G(d,d) \\
\text{Diagonal: } F(d,c) \xrightarrow{F(d,f)} F(d,d) \quad G(c,d) \xrightarrow{G(f,d)} G(d,d) \\
\text{Vertical: } F(d,c) \xrightarrow{F(d,f')} F(d,d) \quad G(c,d) \xrightarrow{G(f',d)} G(d,d)
\end{array} & = &
\begin{array}{c}
\text{Diagram showing a hexagon of 2-cells for a modification } \Phi \\
\text{Top row: } F(c,c) \xrightarrow{\theta_c} G(c,c) \xrightarrow{G(c,f)} G(c,d) \\
\text{Bottom row: } F(d,d) \xrightarrow{\theta_d} G(d,d) \xrightarrow{G(f,d)} G(c,d) \\
\text{Left column: } F(c,c) \xrightarrow{F(f,c)} F(d,c) \xrightarrow{F(d,f')} F(d,d) \\
\text{Right column: } G(c,c) \xrightarrow{G(c,f)} G(c,d) \xrightarrow{G(f',d)} G(d,d) \\
\text{Diagonal: } F(d,c) \xrightarrow{F(d,f)} F(d,d) \quad G(c,d) \xrightarrow{G(f,d)} G(d,d) \\
\text{Vertical: } F(d,c) \xrightarrow{F(d,f')} F(d,d) \quad G(c,d) \xrightarrow{G(f',d)} G(d,d)
\end{array}
\end{array}$$

Similarly to the 1-categorical case, we cannot compose lax dinatural transformations horizontally and therefore the following data

- 0-cells: mixed variance 2-functors $F, G : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$;
- 1-cells: lax dinatural transformations $\theta : F \rightrightarrows G$;
- 2-cells: modifications $\Phi : \theta \Rightarrow \delta$ between them.

does not constitute a 2-category and we need the notion of strong lax dinatural transformation to make it compositional.

2) Strong dinatural transformations:

Definition A.4. For categories \mathbb{C}, \mathbb{D} and functors $F, G : \mathbb{C}^{\text{op}} \times \mathbb{C} \rightarrow \mathbb{D}$, a *strong dinatural transformation* $\gamma : F \rightrightarrows G$ consists of a family of 1-cells $\{\gamma_c : F(c,c) \rightarrow G(c,c)\}_{c \in \mathbb{C}}$ indexed by the objects c in \mathbb{C} such that for every morphism $f : c \rightarrow d$ in \mathbb{C} and for every span $F(c,c) \xleftarrow{q_c} Q \xrightarrow{q_d} F(d,d)$ in \mathbb{D} , if the square

$$\begin{array}{ccc}
& q_c \nearrow F(c,c) & \xrightarrow{\gamma_c} G(c,c) \\
Q & \downarrow & \downarrow \gamma_f \\
& q_d \searrow F(d,d) & \xrightarrow{G(f,d)} G(d,d)
\end{array}$$

commutes, then so does the hexagon:

$$\begin{array}{ccc}
\begin{array}{c}
\text{Diagram showing a hexagon of 2-cells for a strong dinatural transformation } \gamma \\
\text{Top row: } F(c,c) \xrightarrow{\gamma_c} G(c,c) \xrightarrow{G(c,f)} G(c,d) \\
\text{Bottom row: } F(d,d) \xrightarrow{\gamma_d} G(d,d) \xrightarrow{G(f,d)} G(c,d) \\
\text{Left column: } F(c,c) \xrightarrow{q_c} F(d,c) \xrightarrow{F(d,f')} F(d,d) \\
\text{Right column: } G(c,c) \xrightarrow{q_c} G(d,c) \xrightarrow{G(d,f)} G(d,d) \\
\text{Diagonal: } F(d,c) \xrightarrow{F(d,f)} F(d,d) \quad G(d,c) \xrightarrow{G(f,d)} G(d,d) \\
\text{Vertical: } F(d,c) \xrightarrow{F(d,f')} F(d,d) \quad G(d,c) \xrightarrow{G(f',d)} G(d,d)
\end{array} & = &
\begin{array}{c}
\text{Diagram showing a hexagon of 2-cells for a strong dinatural transformation } \gamma \\
\text{Top row: } F(c,c) \xrightarrow{\gamma_c} G(c,c) \xrightarrow{G(c,f)} G(c,d) \\
\text{Bottom row: } F(d,d) \xrightarrow{\gamma_d} G(d,d) \xrightarrow{G(f,d)} G(c,d) \\
\text{Left column: } F(c,c) \xrightarrow{q_c} F(d,c) \xrightarrow{F(d,f')} F(d,d) \\
\text{Right column: } G(c,c) \xrightarrow{q_c} G(d,c) \xrightarrow{G(d,f)} G(d,d) \\
\text{Diagonal: } F(d,c) \xrightarrow{F(d,f)} F(d,d) \quad G(d,c) \xrightarrow{G(f,d)} G(d,d) \\
\text{Vertical: } F(d,c) \xrightarrow{F(d,f')} F(d,d) \quad G(d,c) \xrightarrow{G(f',d)} G(d,d)
\end{array}
\end{array}$$

For functors $F, G, H : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbb{D}$ and strong dinatural transformations $\gamma : F \rightrightarrows G$ and $\delta : G \rightrightarrows H$, the transformation

$$\{\delta_c \gamma_c : F(c, c) \rightarrow H(c, c)\}_{c \in \mathcal{C}}$$

is also strongly dinatural so that strong dinatural transformations form a category. To obtain the 2-dimensional analogue, we first need to consider the notion of lax wedge [31], [32], [33], [34], [35]:

Definition A.5 (lax wedge). Let \mathcal{C}, \mathcal{D} be 2-categories and $F : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$ be a 2-functor. For an object Q of \mathcal{D} , a *lax wedge of Q over F* consists of:

- a family of 1-cells $\{q_c : Q \rightarrow F(c, c)\}_{c \in \mathcal{C}}$ indexed by the objects c in \mathcal{C} ;
- for every 1-cell $f : c \rightarrow d$ in \mathcal{C} , a 2-cell:

$$\begin{array}{ccc} q_c & \nearrow F(c, f) & \\ Q & \Downarrow q_f & F(c, d) \\ q_d & \searrow F(f, d) & \end{array}$$

satisfying the following axioms:

- 1) unity: $q_{1_c} = \text{id}_{q_c}$
- 2) 1-naturality: for 1-cells $f : c \rightarrow d$ and $g : d \rightarrow e$,

$$\begin{array}{ccc} q_c & \nearrow F(c, f) & \\ Q & \Downarrow q_{gf} & F(c, e) \\ q_e & \searrow F(gf, d) & \end{array} = \begin{array}{ccccc} & F(c, f) & & F(c, d) & \\ q_c & \nearrow & \Downarrow q_f & \nearrow & F(c, g) \\ Q & \xrightarrow{q_d} & F(d, d) & \xrightarrow{F(f, d)} & F(c, e) \\ q_e & \searrow & \Downarrow q_g & \searrow & F(f, g) \\ & F(e, e) & \xrightarrow{F(g, e)} & F(d, e) & \end{array}$$

- 3) 2-naturality: for every 2-cell $\alpha : f \Rightarrow f'$,

$$\begin{array}{ccc} q_c & \nearrow F(c, f) & \\ Q & \Downarrow q_{f'} & F(c, d) \\ q_d & \searrow F(f', d) & \end{array} = \begin{array}{ccc} q_c & \nearrow F(c, f) & \\ Q & \Downarrow q_f & F(c, d) \\ q_d & \searrow F(f, d) & \end{array}$$

Definition A.6 (morphism of lax wedges). For two lax conwedges (Q, q) and (P, p) over $F : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$, a lax morphism from (P, p) to (Q, q) consists of

- a 1-cell $u : P \rightarrow Q$ in \mathcal{D} ,
- a family of 2-cells $\{\Gamma_c : p_c \rightarrow q_c u\}_{c \in \mathcal{C}}$ indexed by the objects c in \mathcal{C}

such that for all $f : c \rightarrow d$ in \mathcal{C} , the following two diagrams are equal:

$$\begin{array}{ccc}
\begin{array}{c}
\text{Diagram showing two configurations of lax wedges } (P, q) \text{ and } (Q, p) \text{ over } F. \\
\text{Left side: } P \xrightarrow{u} Q. \text{ Lax wedge } (Q, q) \text{ has components } p_c, q_c, q_f, q_d. \text{ Lax wedge } (P, p) \text{ has components } F(c, c), F(c, d), F(d, d), F(f, d). \\
\text{Right side: } Q \xrightarrow{p} F(c, c). \text{ Lax wedge } (F(c, c), p) \text{ has components } p_c, p_d, p_f. \text{ Lax wedge } (F(d, d), q) \text{ has components } F(c, d), F(f, d).
\end{array} & = &
\begin{array}{c}
\text{Diagram showing the equality between the two configurations above.}
\end{array}
\end{array}$$

Definition A.7 (strong lax dinatural transformation). For 2-categories \mathcal{C}, \mathcal{D} and 2-functors $F, G : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$, a *strong lax dinatural transformation* $\gamma : F \doteqdot G$ consists of:

- a family of 1-cells $\{\gamma_c : F(c, c) \rightarrow G(c, c)\}_{c \in \mathcal{C}}$ indexed by the objects c in \mathcal{C} ;
- for every lax wedge (Q, q) over F and for every 1-cell $f : c \rightarrow d$ in \mathcal{C} , a 2-cell:

$$\begin{array}{ccccc}
& q_c & \nearrow \gamma_c & G(\gamma_{c,f}) & \\
F(c, c) & \xrightarrow{\quad} & G(c, c) & \searrow & \\
\Downarrow \gamma_{q,f} & & & & G(c, d) \\
& q_d & \nearrow \gamma_d & G(\gamma_{f,d}) & \\
F(d, d) & \xrightarrow{\quad} & G(d, d) & \searrow &
\end{array}$$

satisfying the following axioms:

- 1) unity: for every object $c \in \mathcal{C}$, $\gamma_{q,1_c} = \text{id}_{\gamma_c q_c}$,
- 2) 1-naturality: for every 1-cells $f : c \rightarrow d$ and $g : d \rightarrow e$,

$$\begin{array}{ccc}
\gamma_{q,gf} & = &
\begin{array}{ccccc}
& q_c & \nearrow \gamma_c & G(\gamma_{c,f}) & \\
F(c, c) & \xrightarrow{\quad} & G(c, c) & \searrow & G(c, d) \\
\Downarrow \gamma_{q,f} & & & & \nearrow G(\gamma_{f,d}) \\
& q_d & \nearrow \gamma_d & G(\gamma_{d,e}) & \nearrow G(\gamma_{e,c}) \\
F(d, d) & \xrightarrow{\quad} & G(d, d) & \searrow & G(d, e) \\
\Downarrow \gamma_{q,g} & & & & \nearrow G(\gamma_{d,e}) \\
& q_e & \nearrow \gamma_e & G(\gamma_{e,c}) & \\
F(e, e) & \xrightarrow{\quad} & G(e, e) & \searrow & G(c, e)
\end{array} \\
& = &
\begin{array}{c}
G(c, e)
\end{array}
\end{array}$$

- 3) 2-naturality: for every 2-cell $\alpha : f \Rightarrow f'$ in \mathcal{C} :

$$\begin{array}{ccc}
\begin{array}{c}
\text{Diagram showing 2-naturality for a 2-cell } \alpha : f \Rightarrow f' \text{ in } \mathcal{C}. \\
\text{Left side: } Q \xrightarrow{q_c} F(c, c) \xrightarrow{\gamma_c} G(c, c). \text{ Lax wedge } (F(d, d), q_d) \text{ has components } F(c, d), G(c, d), G(\gamma_{c,d}), G(\gamma_{c,f'}), G(\gamma_{c,f}). \\
\text{Right side: } Q \xrightarrow{q_c} F(c, c) \xrightarrow{\gamma_c} G(c, c). \text{ Lax wedge } (F(d, d), q_d) \text{ has components } F(c, d), G(c, d), G(\gamma_{c,d}), G(\gamma_{f,d}), G(\gamma_{f,f'}), G(\gamma_{f',d}).
\end{array} & = &
\begin{array}{c}
\text{Diagram showing the equality between the two configurations above.}
\end{array}
\end{array}$$

- 4) for every morphism of lax wedges $(u, \Gamma) : (P, p) \rightarrow (Q, q)$ over F and for every $f : c \rightarrow d$,

$$\begin{array}{ccc}
\begin{array}{c}
\text{Diagram showing a lax wedge } (Q, q) \text{ over } F \text{ and a 1-cell } f : c \rightarrow d \text{ in } \mathcal{C} \\
P \xrightarrow{u} Q \quad P \xrightarrow{p_c} F(c, c) \xrightarrow{\gamma_c} G(c, c) \xrightarrow{G(\gamma_c, \delta)} G(c, d) \\
Q \xrightarrow{q_d} F(d, d) \xrightarrow{\gamma_d} G(d, d) \xrightarrow{G(f, \delta)} G(c, d)
\end{array} & = &
\begin{array}{c}
P \xrightarrow{p_d} F(c, c) \xrightarrow{\gamma_c} G(c, c) \xrightarrow{G(\gamma_c, \delta)} G(c, d) \\
P \xrightarrow{u} Q \xrightarrow{q_d} F(d, d) \xrightarrow{\gamma_d} G(d, d) \xrightarrow{G(f, \delta)} G(c, d)
\end{array}
\end{array}$$

Definition A.8. A *modification between strong lax dinatural transformation* $\Phi : \gamma \Rightarrow \delta : F \rightrightarrows G$ consists of a family of 2-cells $\{\Phi_c : \gamma_c \Rightarrow \delta_c\}_{c \in \mathcal{C}}$ such that for every lax wedge (Q, q) over F and 1-cell $f : c \rightarrow d$ in \mathcal{C} the following equality holds:

$$\begin{array}{ccc}
\begin{array}{c}
Q \xrightarrow{q_c} F(c, c) \xrightarrow{\gamma_c} G(c, c) \xrightarrow{G(\gamma_c, \delta)} G(c, d) \\
Q \xrightarrow{q_d} F(d, d) \xrightarrow{\gamma_d} G(d, d) \xrightarrow{G(f, \delta)} G(c, d)
\end{array} & = &
\begin{array}{c}
Q \xrightarrow{q_c} F(c, c) \xrightarrow{\gamma_c} G(c, c) \xrightarrow{G(\gamma_c, \delta)} G(c, d) \\
Q \xrightarrow{q_d} F(d, d) \xrightarrow{\gamma_d} G(d, d) \xrightarrow{G(f, \delta)} G(c, d)
\end{array}
\end{array}$$

Definition A.9. For 2-categories \mathcal{C} and \mathcal{D} , we denote by $\text{StDin}(\mathcal{C}, \mathcal{D})$ the following data:

- 0-cells: mixed variance 2-functors $F, G : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{D}$;
- 1-cells: strong lax dinatural transformations $\gamma : F \rightrightarrows G$;
- 2-cells: modifications $\Phi : \gamma \Rightarrow \delta$ between them.

Proposition A.10. For 2-categories \mathcal{C} and \mathcal{D} , $\text{StDin}(\mathcal{C}, \mathcal{D})$ defined above forms a 2-category.

B. Proofs of Section III

Lemma III.2. There exists a 1-cell $t : \mathbf{1} \rightarrow \Phi$ and an invertible 2-cell $\tau : t \Rightarrow Rt$ in \mathcal{D}

$$\begin{array}{ccc}
& \begin{array}{c} t \\ \nearrow \tau \\ \mathbf{1} \end{array} & \begin{array}{c} \Phi \\ \downarrow R \\ \Phi \end{array} \\
& \text{---} & \text{---} \\
& \begin{array}{c} t \\ \searrow \tau \\ \mathbf{1} \end{array} & \begin{array}{c} \Phi \\ \downarrow R \\ \Phi \end{array}
\end{array}$$

satisfying the following property: for any 1-cells $v, w : \mathbf{1} \rightarrow \Phi$ in \mathcal{D} and invertible 2-cells $\nu : v \Rightarrow Rv$ and $\omega : w \Rightarrow Rw$, there exists a unique invertible 2-cell $\psi : v \Rightarrow w$ in \mathcal{D} such that

$$\begin{array}{ccc}
\begin{array}{c}
\mathbf{1} \xrightarrow{w} \Phi \quad \mathbf{1} \xrightarrow{v} \Phi \\
\text{---} \quad \text{---} \\
\mathbf{1} \xrightarrow{w} \Phi \xrightarrow{\omega} \Phi \quad \mathbf{1} \xrightarrow{v} \Phi \xrightarrow{\nu} \Phi
\end{array} & = &
\begin{array}{c}
\mathbf{1} \xrightarrow{w} \Phi \quad \mathbf{1} \xrightarrow{v} \Phi \\
\text{---} \quad \text{---} \\
\mathbf{1} \xrightarrow{w} \Phi \xrightarrow{\psi} \Phi \xrightarrow{\omega} \Phi \quad \mathbf{1} \xrightarrow{v} \Phi \xrightarrow{\nu} \Phi
\end{array}
\end{array}$$

Proof. By Lemma II.10, R is part of an adjoint equivalence $(R : !\Phi \rightarrow \Phi, L : \Phi \rightarrow !\Phi, \eta : \text{id} \xrightarrow{\cong} RL, \varepsilon : LR \xrightarrow{\cong} \text{id})$ where $L : \Phi \rightarrow !\Phi$ is pseudo-final. Therefore, there exists a 1-cell $t : !\mathbf{1} \rightarrow \Phi$ and a 2-cell ξ in \mathcal{C} as below:

$$\begin{array}{ccc}
& \begin{array}{c} \delta_1 \\ \mathbf{1} \xrightarrow{t} !!\mathbf{1} \\ \text{---} \\ \mathbf{1} \xrightarrow{t} \Phi \xrightarrow{\xi} !\Phi \end{array} &
\begin{array}{c} \delta_1 \\ \mathbf{1} \xrightarrow{t} !!\mathbf{1} \\ \text{---} \\ \mathbf{1} \xrightarrow{t} \Phi \xrightarrow{\xi} !\Phi \\ \text{---} \\ \Phi \xrightarrow{L} !\Phi \end{array}
\end{array}$$

and we define τ as the following 2-cell in \mathcal{C}

$$\tau := \begin{array}{c} !1 \xrightarrow{\delta_1} !!1 \\ t \downarrow \nearrow \xi \downarrow !t \\ \Phi \xrightarrow{L} !\Phi \xrightarrow{R} \Phi \\ \uparrow \eta \quad \downarrow \\ 1 \end{array}$$

which corresponds to a 2-cell with boundaries $t \Rightarrow Rt$ in \mathcal{D} as desired.

Assume now that we have 2-cells $\nu : v \Rightarrow Rv$ and $\omega : w \Rightarrow Rw$ in \mathcal{D} . By the universal property of L , there exists a unique 2-cell $\psi : v \Rightarrow w$ such that

$$\begin{array}{ccc} \begin{array}{c} !1 \xrightarrow{\delta_1} !!1 \\ v \xrightarrow{\psi} w \nearrow \omega \downarrow !w \\ \Phi \xrightarrow{R} !\Phi \xrightarrow{\varepsilon} 1 \\ \downarrow L \quad \downarrow \\ \Phi \xrightarrow{R} !\Phi \end{array} & = & \begin{array}{c} !1 \xrightarrow{\delta_1} !!1 \\ v \xrightarrow{\nu} !v \xrightarrow{\psi} !w \downarrow !w \\ \Phi \xrightarrow{R} !\Phi \xrightarrow{\varepsilon} 1 \\ \downarrow L \quad \downarrow \\ \Phi \xrightarrow{R} !\Phi \end{array} \end{array}$$

in \mathcal{C} . Cancelling the 2-cells ε on both sides using the adjunction identities

$$\begin{array}{ccc} \begin{array}{c} !1 \xrightarrow{\delta_1} !!1 \\ v \xrightarrow{\psi} w \nearrow \omega \downarrow !w \\ \Phi \xrightarrow{R} !\Phi \xrightarrow{\varepsilon} 1 \\ \downarrow \eta \quad \downarrow R \\ \Phi \xrightarrow{R} !\Phi \end{array} & = & \begin{array}{c} !1 \xrightarrow{\delta_1} !!1 \\ v \xrightarrow{\nu} !v \xrightarrow{\psi} !w \downarrow !w \\ \Phi \xrightarrow{R} !\Phi \xrightarrow{\varepsilon} 1 \\ \downarrow \eta \quad \downarrow R \\ \Phi \xrightarrow{R} !\Phi \end{array} \end{array}$$

and rewriting the equality above in \mathcal{D} , we obtain that there exists a unique invertible 2-cell $\psi : v \Rightarrow w$ in \mathcal{D} such that:

$$\begin{array}{ccc} \begin{array}{c} w \rightarrow \Phi \\ 1 \xrightarrow{\psi} w \nearrow \omega \xrightarrow{R} \Phi \\ v \xrightarrow{\psi} w \nearrow \omega \xrightarrow{R} \Phi \end{array} & = & \begin{array}{c} w \rightarrow \Phi \\ 1 \xrightarrow{\psi} v \xrightarrow{\nu} w \nearrow \omega \xrightarrow{R} \Phi \end{array} \end{array}$$

□

Lemma III.3 (Naturality of fix). *For a 2-cell $\alpha : f \Rightarrow g$ in \mathcal{D} , we have $\text{fix}_g \alpha^* = (\alpha \cdot \alpha^*) \text{fix}_f$.*

Proof. Immediate unfolding of the corresponding 2-cells. □

Proposition III.4. *The 2-cells unif_γ we constructed yield a strong pseudo-dinatural transformation and they verify the coherence axiom between fix and unif .*

Proof. For the strong pseudo-dinatural axioms, we will only prove the 2-naturality axiom, as the other cases work similarly. Assume that we have the following equality in \mathcal{D} :

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \\
 \begin{array}{c} A \xrightarrow{f} A \\ \Downarrow J\theta \\ B \xrightarrow{g} B \end{array} & = & \begin{array}{c} A \xrightarrow{f} A \\ \Downarrow \rho \\ B \xrightarrow{g} B \end{array} \\
 \text{Diagram 2:} & & \\
 \begin{array}{c} A \xrightarrow{f} A \\ \Downarrow \gamma \\ B \xrightarrow{g} B \end{array} & & \begin{array}{c} A \xrightarrow{f} A \\ \Downarrow J\theta \\ B \xrightarrow{g} B \end{array}
 \end{array}$$

Let $\phi : ruf_f \Rightarrow u_g$ be the unique 2-cell in \mathcal{C} such that:

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \\
 \begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \Downarrow \mu_f \\ Juf_f \\ \Downarrow \rho \\ J\phi \\ \Downarrow \mu_g \\ Jr \\ \Downarrow \gamma \\ B \xrightarrow{g} B \end{array} & = & \begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \Downarrow \mu_g \\ Juf_f \\ \Downarrow \mu_g \\ J\phi \\ \Downarrow \rho \\ Jr \\ \Downarrow \gamma \\ B \xrightarrow{g} B \end{array}
 \end{array}$$

so that $\mathbf{unif}_\rho = J\phi \cdot t : J(r)J(u_f)t \Rightarrow J(u_g)t$. We want to show that $\mathbf{unif}_\gamma = \mathbf{unif}_\rho(J\theta \cdot (J(u_f)t))$ which is immediate from the equality below:

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \\
 \begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \Downarrow \mu_f \\ Juf_f \\ \Downarrow \rho \\ J\phi \\ \Downarrow \mu_g \\ Jr \\ \Downarrow \gamma \\ B \xrightarrow{g} B \end{array} & = & \begin{array}{c} \Phi \xrightarrow{R} \Phi \\ \Downarrow \mu_g \\ Juf_f \\ \Downarrow \mu_g \\ J\phi \\ \Downarrow \rho \\ Jr \\ \Downarrow \gamma \\ B \xrightarrow{g} B \end{array}
 \end{array}$$

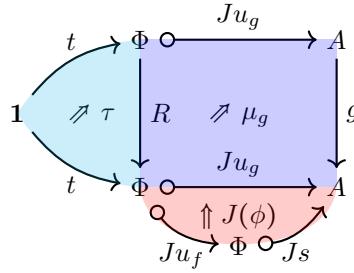
To prove that the operator $((-)^*, \mathbf{fix}, \mathbf{unif})$ we constructed is a pseudo-fixpoint operator on \mathcal{D} uniform with respect to J , it only remains to show the coherence axiom between \mathbf{fix} and \mathbf{unif} i.e. that for a 2-cell

$$\begin{array}{c} A \xrightarrow{f} A \\ \Downarrow \gamma \\ B \xrightarrow{g} B \end{array}$$

in \mathcal{D} , we have:

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \\
 \begin{array}{c} g^* \\ \nearrow \mathbf{fix}_g \\ 1 \\ \searrow \mathbf{unif}_\gamma \\ f^* \\ \nearrow \mathbf{fix}_f \\ A \circ J_s \\ \searrow \\ B \xrightarrow{g} B \end{array} & = & \begin{array}{c} g^* \\ \nearrow \mathbf{fix}_g \\ 1 \\ \searrow \mathbf{unif}_\gamma \\ f^* \\ \nearrow \mathbf{fix}_f \\ A \circ J_s \\ \searrow \\ B \xrightarrow{g} B \end{array}
 \end{array}$$

The left-hand side is equal to:



where $\phi : s u_f \Rightarrow u_g$ is the unique invertible 2-cell in \mathcal{C} such that:

$$\begin{array}{ccc}
 \text{Diagram LHS} & = & \text{Diagram RHS}
 \end{array}$$

The LHS diagram shows a complex commutative square with nodes A, B, and intermediate nodes. It includes arrows R , $J u_f$, $J u_g$, f , g , μ_f , μ_g , γ , and $J(\phi)$. The RHS diagram shows a simplified version of the same structure.

so that the left-hand side is equal to the diagram below

which corresponds to the right-hand side of the desired equality. \square

Proposition III.5. *The category $\text{Fix}(\mathcal{D}, J)$ is contractible i.e. δ is an isomorphism of uniform pseudo-fixpoint operators and it is unique.*

Proof. We first show that δ is a morphism of pseudo-fixpoint operators by proving that it commutes with the fix 2-cells:

$$\begin{array}{ccc}
 \text{Diagram LHS} & = & \text{Diagram RHS}
 \end{array}$$

The LHS diagram shows a complex commutative square with nodes A, B, and intermediate nodes. It includes arrows f^\dagger , f^* , fix_f^\dagger , fix_f^* , δ_f , and f . The RHS diagram shows a simplified version of the same structure.

The left hand diagram is equal to:

$$\begin{array}{ccc}
 \text{Diagram LHS} & = & \text{Diagram RHS}
 \end{array}$$

The LHS diagram shows a complex commutative square with nodes A, B, and intermediate nodes. It includes arrows f^\dagger , f , fix_f^\dagger , R^\dagger , δ_0 , $\text{unif}_{\mu_f}^\dagger$, and $J u_f$. The RHS diagram shows a simplified version of the same structure.

where the equality above follows from the coherence between fix^\dagger and unif^\dagger . By definition of δ_0 , the right-hand diagram is equal to:

and we obtain the desired equality. We now need to show that δ is a morphism of uniform pseudo-fixpoint operators, *i.e.* for every square in \mathcal{D} :

$$\begin{array}{ccc} & A & \\ f & \nearrow & \searrow \\ J(s) & & J(s) \\ \downarrow & \swarrow \gamma & \downarrow \\ B & \xrightarrow{g} & B \end{array}$$

we have:

By definition of unif_γ^* and the 2-naturality axiom for unif^\dagger , the left hand diagram is equal to:

where ϕ is the unique 2-cell $s u_f \Rightarrow u_g$ such that:

The right hand diagram is equal to

where the equality above follows from the 1-naturality axiom for unif^\dagger .

For uniqueness, assume that there exists another morphism of uniform pseudo-fixpoint operators $\beta : (-)^* \rightarrow (-)^\dagger$. We proceed to show that for every $f : A \rightarrow A$, $\delta_f = \beta_f$. Let ϕ_0 be the unique invertible 2-cell $1_\Phi \Rightarrow u_R$ in \mathcal{C} such that:

which we obtain from Lemma III.1. We show that

by proving the following equality:

and using the universal property of δ_0 . By definition of ϕ_0 , the left-hand side is equal to:

Since β is a morphism of pseudo fixpoint operators, the right-hand side is equal to:

$$\begin{array}{c}
 R^\dagger \\
 \text{---} \circlearrowleft \uparrow \beta_R \quad \Phi \\
 1 \xrightarrow{t} \Phi \circ J u_R \quad \Phi \\
 \nearrow \text{fix}_R^* J u_R \\
 t \quad \Phi \circ \uparrow J \phi_0 \quad \Phi \\
 \Phi \circ \quad 1_\Phi
 \end{array}$$

so both sides are equal as desired by definition of fix_R^* . We therefore obtain that

$$\delta_f =
 \begin{array}{c}
 1_\Phi \\
 \text{---} \circlearrowleft \downarrow J \phi_0 \quad \Phi \\
 1 \xrightarrow{t} \Phi \circ \downarrow \beta_R \quad \Phi \\
 \downarrow R^\dagger \quad \text{unif}_{\mu_f}^* \\
 f^\dagger \quad A
 \end{array}$$

Since β is a morphism of uniform pseudo-fixpoint operators, δ_f is equal to:

$$\delta_f =
 \begin{array}{c}
 1_\Phi \\
 \text{---} \circlearrowleft \downarrow J \phi_0 \quad \Phi \\
 1 \xrightarrow{t} \Phi \circ \downarrow J u_R \quad \Phi \\
 \downarrow f^* \quad \text{unif}_{\mu_f}^* \\
 \downarrow \beta_f \quad f^\dagger \quad A
 \end{array}$$

Using the universal property of $\text{unif}_{\mu_f}^*$, we obtain that $\text{unif}_{\mu_f}^*(J \phi_0 \cdot t) = \text{id}$ giving us the desired conclusion. \square

Lemma IV.2. For 1-cells $v, w : 1 \rightarrow \Phi$ in \mathcal{D} and 2-cells

$$\begin{array}{ccc}
 1 & \xrightarrow{v} & \Phi \\
 \nearrow \nu & & \downarrow RR \\
 & & \Phi
 \end{array}
 \quad \text{and} \quad
 \begin{array}{ccc}
 1 & \xrightarrow{w} & \Phi \\
 \nearrow \omega & & \downarrow RR \\
 & & \Phi
 \end{array}$$

in \mathcal{D} , there exists a unique invertible 2-cell $\lambda : w \Rightarrow v$ in \mathcal{D} such that

$$\begin{array}{ccc}
 1 & \xrightarrow{v} & \Phi \\
 \nearrow v \quad \nearrow \nu & & \downarrow RR \\
 \nearrow \lambda & & \Phi \\
 w & \xrightarrow{w} & \Phi
 \end{array}
 =
 \begin{array}{ccc}
 1 & \xrightarrow{v} & \Phi \\
 \nearrow w \quad \nearrow \lambda \quad \nearrow \omega & & \downarrow RR \\
 w & \xrightarrow{w} & \Phi
 \end{array}$$

Proof. The 2-cell ν in \mathcal{D} has the following boundary in \mathcal{C} :

$$\begin{array}{ccccc}
 !1 & \xrightarrow{\delta_1} & !!1 & \xrightarrow{\delta_{!!1}} & !!!1 \\
 \downarrow !v & = & \downarrow !!v \\
 !\Phi & \xrightarrow{\delta_\Phi} & !!\Phi & & \\
 \nearrow \nu & & & & \\
 v & & & & \\
 \downarrow !R & & & & \\
 !\Phi & & & & \\
 \downarrow R & & & & \\
 \Phi & & & &
 \end{array}$$

Since $!LL$ is a pseudo-final $!!$ -coalgebra, there is a unique isomorphism $\lambda : w \Rightarrow v$ such that:

$$\begin{array}{ccc}
 \begin{array}{c}
 !1 \xrightarrow{\delta_1} !!1 \xrightarrow{\delta_{!!1}} !!!1 \\
 \text{---} \\
 w \text{ (shaded)} \xrightarrow{\lambda} v \\
 \text{---} \\
 \text{---} \\
 \text{---} \\
 \Phi \xrightarrow{L} !\Phi \xrightarrow{!L} !!\Phi
 \end{array}
 & = &
 \begin{array}{c}
 !1 \xrightarrow{\delta_1} !!1 \xrightarrow{\delta_{!!1}} !!!1 \\
 \text{---} \\
 w \text{ (shaded)} \xrightarrow{\lambda} v \text{ (shaded)} \\
 \text{---} \\
 \text{---} \\
 \text{---} \\
 \Phi \xrightarrow{L} !\Phi \xrightarrow{!L} !!\Phi
 \end{array}
 \end{array}$$

Cancelling the 2-cells ε and $!\varepsilon$ on both sides using the adjunction identities and rewriting the equality above in \mathcal{D} , we obtain the desired equality in \mathcal{D} . \square

Proposition IV.3. *The 2-cells dinat_g^f we constructed yield a pseudo-dinatural transformation and they verify the coherence axiom between **dinat** and **unif**.*

Proof. Coherence between dinat and unif: for two squares in \mathcal{D} of the form

$$\begin{array}{ccccc}
 & & f & & \\
 & A & \xrightarrow{} & B & \xrightarrow{g} A \\
 & \circ & & \circ & \circ \\
 \rho * \gamma = & Js & \Downarrow \gamma & Jr & \Downarrow \rho \\
 & \downarrow & & \downarrow & \downarrow \\
 & C & \xrightarrow{h} D & \xrightarrow{k} C
 \end{array}$$

we have:

$$\begin{array}{ccc}
 \begin{array}{c}
 (kh)^* \\
 \text{---} \\
 1 \xrightarrow{\text{dinat}_k^h} C \\
 \text{---} \\
 (fg)^* \xrightarrow{\text{unif}_{\gamma * \rho}} B \\
 \text{---} \\
 B \xrightarrow{Jr} D
 \end{array}
 & = &
 \begin{array}{c}
 (kh)^* \\
 \text{---} \\
 1 \xrightarrow{\text{unif}_{\rho * \gamma}} C \\
 \text{---} \\
 (gf)^* \xrightarrow{\text{dinat}_g^f} A \\
 \text{---} \\
 A \xrightarrow{Js} C \\
 \text{---} \\
 B \xrightarrow{Jr} D
 \end{array}
 \end{array}$$

where $\gamma * \rho$ corresponds to the following 2-cell:

$$\begin{array}{ccccc}
 & B & \xrightarrow{g} & A & \xrightarrow{f} B \\
 \gamma * \rho = & \downarrow Jr & \swarrow \rho & \downarrow Js & \searrow \gamma \\
 & D & \xrightarrow{k} & C & \xrightarrow{h} D
 \end{array}$$

The right-hand diagram is equal to:

and the left-hand diagram is equal to:

Let $\phi : (s \times r)u_{\sigma(f \times g)} \Rightarrow u_{\sigma(h \times k)}$ be the unique invertible 2-cell in \mathcal{C} such that

The left-hand diagram of the desired equality is then equal to

From the universal property of **unif**, we can now verify that

$$\text{unif}_{\Pi_{\sigma(h \times k)}^1}(\pi_1 \cdot J\phi \cdot (RR)^*) = \text{unif}_{\rho * \gamma}(Js \cdot \text{unif}_{\Pi_{\sigma(f \times g)}^1}) \quad \text{and}$$

$$(\pi_2 \cdot J\phi^{-1} \cdot (RR)^*) \text{unif}_{\Pi_{\sigma(h \times k)}^2}^{-1} \text{unif}_{\gamma * \rho} = Jr \cdot \text{unif}_{\Pi_{\sigma(f \times g)}^2}^{-1}$$

to obtain the desired equality.

Coherence between dinat and fix: for 1-cells $f : A \rightarrow B$ and $g : B \rightarrow A$ in \mathcal{D} , we want to show:

The right-hand diagram is equal to:

$$\begin{array}{ccc}
 & \Phi \circ \xrightarrow{Ju_{fg}} B & \\
 t \nearrow \tau \quad R \quad \nearrow \mu_{fg} & \downarrow & \\
 1 \circ \xrightarrow{Ju_{fg}} B & &
 \end{array}$$

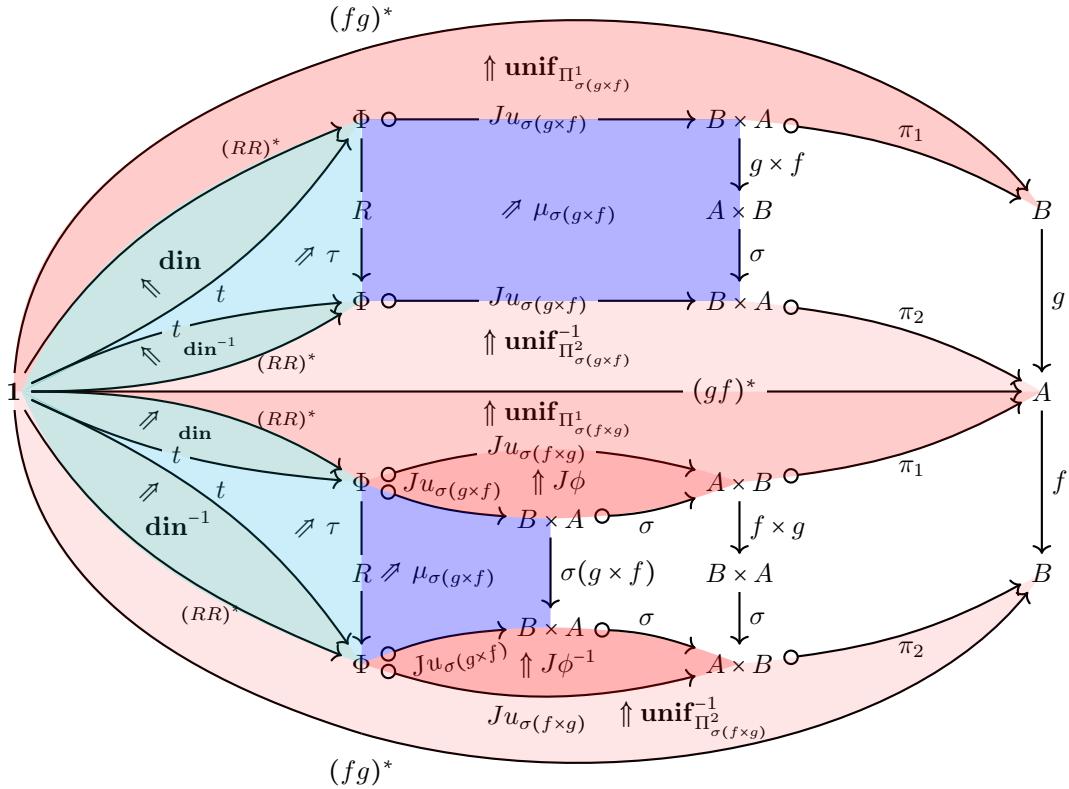
and the left-hand diagram is equal to:

$$\begin{array}{c}
 (fg)^* \\
 \uparrow \text{unif}_{\Pi^1_{\sigma(g \times f)}} \\
 \Phi \circ \xrightarrow{Ju_{\sigma(g \times f)}} B \times A \circ \xrightarrow{\pi_1} B \\
 \nearrow \mu_{\sigma(g \times f)} \quad \downarrow g \times f \\
 R \circ \xrightarrow{Ju_{\sigma(g \times f)}} B \times A \circ \xrightarrow{\pi_2} A \\
 \uparrow \text{unif}_{\Pi^2_{\sigma(g \times f)}}^{-1} \\
 \Phi \circ \xrightarrow{Ju_{\sigma(f \times g)}} A \times B \circ \xrightarrow{\pi_1} A \\
 \nearrow \mu_{\sigma(f \times g)} \quad \downarrow f \times g \\
 R \circ \xrightarrow{Ju_{\sigma(f \times g)}} A \times B \circ \xrightarrow{\pi_2} B \\
 \uparrow \text{unif}_{\Pi^2_{\sigma(f \times g)}}^{-1} \\
 (fg)^*
 \end{array}$$

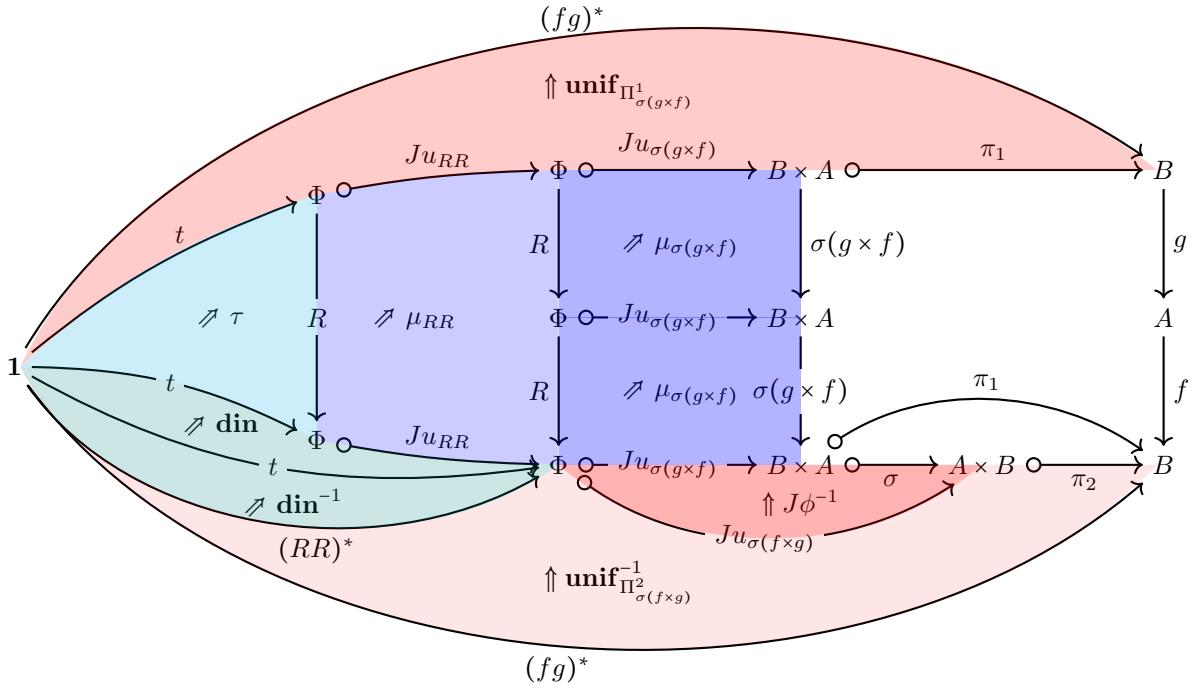
We use a similar reasoning as before and let $\phi : \sigma u_{\sigma(g \times f)} \Rightarrow u_{\sigma(f \times g)}$ be the unique invertible 2-cell in \mathcal{C} such that

$$\begin{array}{ccc}
 \Phi \xrightarrow{R} \Phi & & \\
 J u_{\sigma(g \times f)} \quad \swarrow \mu_{\sigma(g \times f)} & & J u_{\sigma(g \times f)} \\
 \downarrow \quad \downarrow \sigma & & \downarrow \quad \downarrow \sigma \\
 J \phi \circ \xrightarrow{g \times f} A \times B \xrightarrow{\sigma} B \times A & = & J u_{\sigma(f \times g)} \\
 \downarrow \quad \downarrow \sigma & & \downarrow \quad \downarrow \sigma \\
 A \times B \xrightarrow{f \times g} B \times A \xrightarrow{\sigma} A \times B & & A \times B \xrightarrow{f \times g} B \times A \xrightarrow{\sigma} A \times B
 \end{array}$$

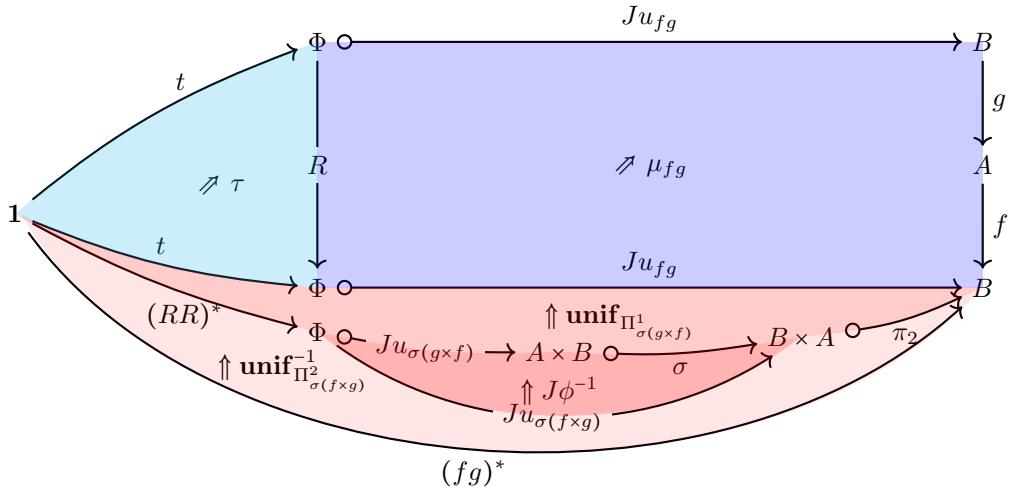
so that we can rewrite the previous diagram as:



Using the universal property of **unif**, we can show that $\text{unif}_{\Pi_{\sigma(g \times f)}^1}^{-1} \text{unif}_{\Pi_{\sigma(f \times g)}^1}(\pi_1 \cdot J\phi \cdot t) = \text{id}$ so that we can simplify the diagram to

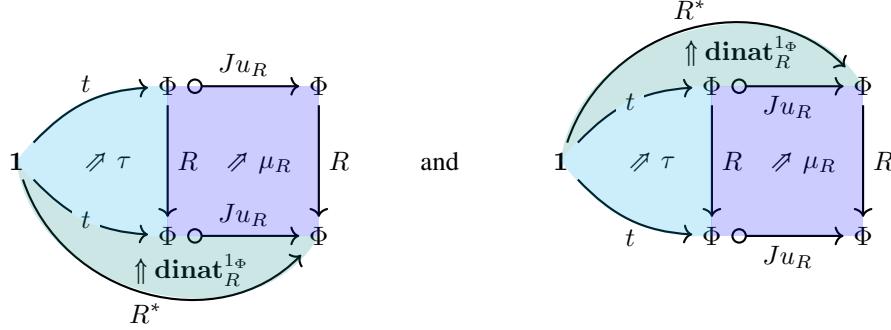


Using the definition of $\text{unif}_{\Pi_{\sigma(g \times f)}^1}$ and the coherence between **unif** and **fix** we rewrite the diagram above as:

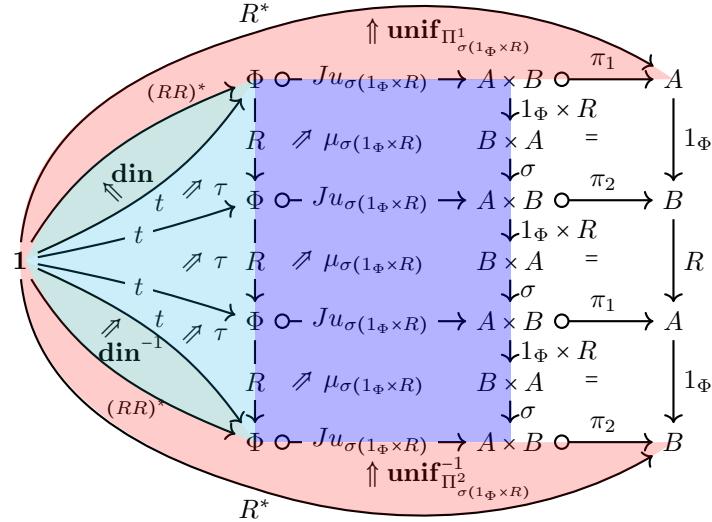


It remains to verify that $\text{unif}_{\Pi^1_{\sigma(g \times f)}}(\pi_2 \cdot J\phi^{-1} \cdot (RR)^*) \text{unif}_{\Pi^2_{\sigma(f \times g)}}^{-1} = \text{id}$ and we obtain the desired equality.

Pseudo-dinaturality axioms for dinat: We only show the unity axiom as the others work similarly. We want to show that for all $f : A \rightarrow A$, $\text{dinat}_f^{1_A} = \text{id}_{f^*}$. We start by proving it for $f = R$ and obtain the general statement by uniformity. We first note that the two diagrams below



are both equal to:



Using Lemma III.2, we conclude that $\text{dinat}_R^{1_\Phi} = \text{id}_{R^*}$.

For a general $f : A \rightarrow A$ in \mathcal{D} , to show that $\text{dinat}_f^{1_A} = \text{id}_{f^*}$, we make use of the coherence axiom between **unif** and **dinat** we proved previously. Consider the two squares:

$$\begin{array}{ccccc} & \Phi & \xrightarrow{1_\Phi} & \Phi & \xrightarrow{R} \Phi \\ J u_f \downarrow & = & J u_f \downarrow & \not\approx \mu_f & \downarrow J u_f \\ A & \xrightarrow{1_A} & A & \xrightarrow{f} & A \end{array}$$

we obtain:

$$\begin{array}{ccc} \text{Left Square: } & \begin{array}{c} f^* \\ \swarrow \text{dinat}_f^{1_A} \\ 1 \end{array} & \begin{array}{c} A \\ \searrow \text{unif}_{\mu_f} \\ R^* \end{array} \\ & \text{Right Square: } & \begin{array}{c} f^* \\ \uparrow \text{unif}_{\mu_f} \\ 1 \end{array} & \begin{array}{c} A \\ \searrow \text{dinat}_R^{1_\Phi} \\ R^* \end{array} \\ & = & & = \end{array}$$

since we have shown that $\text{dinat}_R^{1_\Phi} = \text{id}_{R^*}$, we obtain that $\text{dinat}_f^{1_A} = \text{id}_{f^*}$ from the equality above. \square

Lemma A.11. *For the pseudo-bifree algebra R , the 2-cell dinat_R^R is equal to:*

$$\text{dinat}_R^R = 1 \begin{array}{c} \nearrow \text{din} \\ \nearrow t \\ \nearrow \text{dim}^{-1} \end{array} \begin{array}{c} \nearrow \tau \\ \nearrow t \\ \nearrow \text{dim} \end{array} \begin{array}{c} (RR)^* \\ \searrow R \\ \searrow \Phi \end{array}$$

Proof. Using Lemma III.2, there exists a unique invertible 2-cell $\psi : t \Rightarrow (RR)^*$ such that:

$$\begin{array}{ccc} \text{Left Square: } & \begin{array}{c} (RR)^* \\ \swarrow \text{dinat}_R^R \\ 1 \end{array} & \begin{array}{c} R \\ \searrow t \\ \searrow \Phi \end{array} \\ & \text{Right Square: } & \begin{array}{c} (RR)^* \\ \swarrow \psi \\ 1 \end{array} & \begin{array}{c} R \\ \searrow t \\ \searrow \Phi \end{array} \\ & = & & = \end{array}$$

It suffices to show that $\psi = \text{din}$ by proving that the equality below holds:

$$\begin{array}{ccc} \text{Left Square: } & \begin{array}{c} t \\ \nearrow \tau \\ \nearrow t \\ \nearrow \psi \end{array} & \begin{array}{c} \Phi \circ J u_{RR} \\ \searrow R \\ \searrow \Phi \end{array} \\ & \text{Right Square: } & \begin{array}{c} t \\ \nearrow \psi \\ \nearrow t \\ \nearrow \tau \\ \nearrow t \\ \nearrow \tau \\ \nearrow R \\ \searrow R \\ \searrow \Phi \end{array} \\ & = & = \end{array}$$

By definition of ψ and the coherence axiom between **dinat** and **fix**, the left-hand diagram is equal to

and we obtain the desired equality. \square

Proposition IV.4. *The category $\text{DinFix}(\mathcal{D}, J)$ is contractible i.e. δ commutes with the dinaturality 2-cells and it is unique.*

Proof. Let $((-)^{\dagger}, \text{dinat}^{\dagger}, \text{unif}^{\dagger})$ be another pseudo-dinatural fixpoint operators uniform with respect to J on \mathcal{D} . We show that the morphism $\delta : ((-)^*, \text{unif}^*) \rightarrow ((-)^{\dagger}, \text{unif}^{\dagger})$ we defined in Section III also commutes with the structural 2-cells **dinat**, i.e. it satisfies the following coherence for every $f : A \rightarrow B$ and $g : B \rightarrow A$:

As before, we first show it for $f = g = R$

and obtain the general result using the coherence between **unif** and **dinat**. Using Lemma A.11, the right-hand diagram is equal to:

where $\phi_0 : 1_\Phi \Rightarrow J(u_R)$ is the unique invertible 2-cell such that $R \cdot J\phi_0 = \mu_R(J\phi_0 \cdot R)$ (see proof of Proposition III.5).

Let $\zeta : R^\dagger \Rightarrow (RR)^\dagger$ be the unique invertible 2-cell (obtained from Lemma IV.2) such that

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \text{Diagram 2:} \\
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \zeta} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow R^\dagger} \Phi \xrightarrow{\Phi} R \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow \text{fix}_R^\dagger} \Phi \xrightarrow{\Phi} R \\
 \text{Bottom: } 1 \xrightarrow{\text{ } \nearrow \text{fix}_R^\dagger} R^\dagger \xrightarrow{\Phi} \Phi
 \end{array} & = &
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \text{fix}_{RR}^\dagger} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \zeta} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow R^\dagger} R^\dagger \xrightarrow{\Phi} \Phi
 \end{array}
 \end{array}$$

It plays a similar role as $\text{din} : t \Rightarrow (RR)^*$ for the operator $(-)^*$ as we can show that:

$$\text{dinat}_R^{R^\dagger} = 1 \xrightarrow{\text{ } \nearrow \zeta} (RR)^\dagger \xrightarrow{\Phi} \Phi$$

$$\begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow R^\dagger} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \text{fix}_R^\dagger} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow \zeta^{-1}} R^\dagger \xrightarrow{\Phi} (RR)^\dagger
 \end{array}$$

We can also prove that:

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \text{Diagram 2:} \\
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \delta_{RR}} (RR)^* \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \text{din}} (RR)^* \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow J\phi_0^{-1}} \Phi \xrightarrow{\Phi} Ju_R
 \end{array} & = &
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \zeta} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \delta_R} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow Ju_R} \Phi \xrightarrow{\Phi} Ju_R
 \end{array}
 \end{array}$$

$$\begin{array}{ccc}
 \text{Diagram 3:} & & \text{Diagram 4:} \\
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \delta_R} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \text{din}^{-1}} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow J\phi_0} \Phi \xrightarrow{\Phi} Ju_R
 \end{array} & = &
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \zeta^{-1}} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \delta_{RR}} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow Ju_R} (RR)^* \xrightarrow{\Phi} \Phi
 \end{array}
 \end{array}$$

We can now conclude:

$$\begin{array}{ccc}
 \text{Diagram 1:} & & \text{Diagram 2:} \\
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \tau} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \delta_{RR}} (RR)^* \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow \text{din}^{-1}} (RR)^* \xrightarrow{\Phi} \Phi
 \end{array} & = &
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \zeta} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \delta_R} R^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow \text{fix}_R^\dagger} R^\dagger \xrightarrow{\Phi} \Phi
 \end{array}
 \end{array}$$

$$\begin{array}{ccc}
 \text{Diagram 3:} & & \text{Diagram 4:} \\
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \delta_{RR}} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \text{din}^{-1}} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow J\phi_0} (RR)^\dagger \xrightarrow{\Phi} \Phi
 \end{array} & = &
 \begin{array}{c}
 \text{Left: } 1 \xrightarrow{\text{ } \nearrow \text{dinat}_R^{R^\dagger}} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Middle: } 1 \xrightarrow{\text{ } \nearrow \delta_{RR}} (RR)^\dagger \xrightarrow{\Phi} \Phi \\
 \text{Right: } 1 \xrightarrow{\text{ } \nearrow (RR)^*} (RR)^\dagger \xrightarrow{\Phi} \Phi
 \end{array}
 \end{array}$$

For general morphisms $f : A \rightarrow B$ and $g : B \rightarrow A$ in \mathcal{D} , we consider the squares:

$$\begin{array}{ccccc}
& & R & & \\
\Phi & \xrightarrow{\quad} & \Phi & \xrightarrow{\quad} & \Phi \\
\downarrow Ju_{\sigma(f \times g)} & \swarrow \mu_{\sigma(f \times g)} & \downarrow Ju_{\sigma(f \times g)} & \swarrow \mu_{\sigma(f \times g)} & \downarrow Ju_{\sigma(f \times g)} \\
A \times B & \xrightarrow{\quad} & A \times B & \xrightarrow{\quad} & A \times B \\
\pi_1 \downarrow & \sigma(f \times g) & \pi_2 \downarrow & \sigma(f \times g) & \pi_1 \downarrow \\
A & \xrightarrow{f} & B & \xrightarrow{g} & A
\end{array}$$

and obtain from the coherence between **unif** and **dinat**:

$$\begin{array}{ccc}
\text{(left diagram)} & = & \text{(right diagram)} \\
\begin{array}{c}
\text{Diagram showing coherence between } \text{dinat}_g^{f\dagger} \text{ and } \text{unif}_{\Pi^2_{\sigma(f \times g)}}^{\dagger}. \\
\text{It shows paths from } 1 \text{ to } A \text{ and } B \text{ through } f \text{ and } g. \\
\text{Red paths: } (RR)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Blue paths: } (gf)^\dagger \text{ to } A \text{ and } B \text{ via } f \text{ and } g. \\
\text{Green paths: } (fg)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}).
\end{array} & = & \begin{array}{c}
\text{Diagram showing coherence between } \text{dinat}_R^{R\dagger} \text{ and } \text{unif}_{\Pi^1_{\sigma(f \times g)}}^{\dagger}. \\
\text{It shows paths from } 1 \text{ to } A \text{ and } B \text{ through } f \text{ and } g. \\
\text{Red paths: } (RR)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Blue paths: } (gf)^\dagger \text{ to } A \text{ and } B \text{ via } f \text{ and } g. \\
\text{Green paths: } (fg)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Purple paths: } (RR)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } A \text{ via } J\pi_1(u_{\sigma(f \times g)}).
\end{array}
\end{array}$$

Therefore, we have:

$$\begin{array}{ccc}
\text{(left diagram)} & = & \text{(right diagram)} \\
\begin{array}{c}
\text{Diagram showing coherence between } \text{dinat}_g^{f\dagger} \text{ and } \delta_{fg}. \\
\text{It shows paths from } 1 \text{ to } A \text{ and } B \text{ through } f \text{ and } g. \\
\text{Red paths: } (RR)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Blue paths: } (gf)^\dagger \text{ to } A \text{ and } B \text{ via } f \text{ and } g. \\
\text{Orange paths: } (fg)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Yellow paths: } (fg)^* \text{ to } B \text{ via } f \text{ and } g.
\end{array} & = & \begin{array}{c}
\text{Diagram showing coherence between } \text{dinat}_R^{R\dagger} \text{ and } \delta_{fg}. \\
\text{It shows paths from } 1 \text{ to } A \text{ and } B \text{ through } f \text{ and } g. \\
\text{Red paths: } (RR)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Blue paths: } (gf)^\dagger \text{ to } A \text{ and } B \text{ via } f \text{ and } g. \\
\text{Green paths: } (fg)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } B \text{ via } J\pi_2(u_{\sigma(f \times g)}). \\
\text{Yellow paths: } (fg)^* \text{ to } B \text{ via } f \text{ and } g. \\
\text{Purple paths: } (RR)^\dagger \text{ to } \Phi \text{ and } \Phi \text{ to } A \text{ via } J\pi_1(u_{\sigma(f \times g)}). \\
\text{Orange paths: } (gf)^\dagger \text{ to } A \text{ and } B \text{ via } f \text{ and } g. \\
\text{Yellow paths: } (fg)^* \text{ to } B \text{ via } f \text{ and } g.
\end{array}
\end{array}$$

$$\begin{array}{ccc}
& \text{(Top Diagram)} & \\
\begin{array}{c} = 1 \\ \text{Left side: } \end{array} & \begin{array}{c} (gf)^\dagger \\ \uparrow \text{unif}_{\Pi^1_{\sigma(f \times g)}}^\dagger \\ (RR)^\dagger \rightarrow \Phi \circ Ju_{\sigma(f \times g)} \rightarrow A \times B \circ \pi_1 \rightarrow A \\ \nearrow \text{dinat}_R^{R\dagger} \quad R \nearrow \mu_{\sigma(f \times g)} \\ (RR)^\dagger \rightarrow \Phi \circ Ju_{\sigma(f \times g)} \rightarrow A \times B \circ \pi_2 \rightarrow B \\ \nearrow \delta_{RR} \quad (RR)^* \\ (RR)^* \uparrow \text{unif}_{\Pi^2_{\sigma(f \times g)}}^{*-1} \\ (fg)^* \end{array} & \begin{array}{c} = 1 \\ \text{Right side: } \end{array} \\
& \begin{array}{c} (gf)^\dagger \\ \uparrow \text{unif}_{\Pi^1_{\sigma(f \times g)}}^\dagger \\ (RR)^\dagger \rightarrow \Phi \circ Ju_{\sigma(f \times g)} \rightarrow A \times B \circ \pi_1 \rightarrow A \\ \nearrow \delta_{RR} \quad (RR)^* \\ (RR)^* \rightarrow \Phi \circ Ju_{\sigma(f \times g)} \rightarrow A \times B \circ \pi_2 \rightarrow B \\ \nearrow \mu_{\sigma(f \times g)} \quad B \times A \\ \downarrow \sigma \quad f \\ (fg)^* \end{array} &
\end{array}$$

$$\begin{array}{ccc}
& \text{(Bottom Diagram)} & \\
\begin{array}{c} = 1 \\ \text{Left side: } \end{array} & \begin{array}{c} (gf)^* \\ \uparrow \delta_{gf} \\ \uparrow \text{unif}_{\Pi^1_{\sigma(f \times g)}}^* \\ (RR)^* \rightarrow \Phi \circ Ju_{\sigma(f \times g)} \rightarrow A \times B \circ \pi_1 \rightarrow A \\ \nearrow \text{dinat}_R^{R*} \quad R \\ (RR)^* \rightarrow \Phi \circ Ju_{\sigma(f \times g)} \rightarrow A \times B \circ \pi_2 \rightarrow B \\ \nearrow \delta_{gf} \quad (gf)^\dagger \\ (gf)^\dagger \uparrow \text{unif}_{\Pi^2_{\sigma(f \times g)}}^{*-1} \\ (fg)^\dagger \end{array} & \begin{array}{c} = 1 \\ \text{Right side: } \end{array} \\
& \begin{array}{c} (gf)^\dagger \\ \uparrow \delta_{gf} \\ (gf)^\dagger \rightarrow A \\ (gf)^\dagger \rightarrow B \\ \nearrow \text{dinat}_g^{f*} \\ (fg)^\dagger \end{array} &
\end{array}$$

□

Enriched Lawvere Theories for Operational Semantics

John C. Baez and Christian Williams
Department of Mathematics
U. C. Riverside
Riverside, CA
92521 USA

baez@math.ucr.edu, cwill041@ucr.edu

Enriched Lawvere theories are a generalization of Lawvere theories that allow us to describe the operational semantics of formal systems. For example, a graph-enriched Lawvere theory describes structures that have a *graph* of operations of each arity, where the vertices are operations and the edges are *rewrites* between operations. Enriched theories can be used to equip systems with operational semantics, and maps between enriching categories can serve to translate between different forms of operational and denotational semantics. The Grothendieck construction lets us study all models of all enriched theories in all contexts in a single category. We illustrate these ideas with the *SKI*-combinator calculus, a variable-free version of the lambda calculus.

1 Introduction

Formal systems are not always explicitly connected to how they operate in practice. Lawvere theories [20] are an excellent formalism for describing algebraic structures obeying equational laws, but they do not specify how to compute in such a structure, for example taking a complex expression and simplifying it using rewrite rules. Recall that a Lawvere theory is a category with finite products T generated by a single object t , for “type”, and morphisms $t^n \rightarrow t$ representing n -ary operations, with commutative diagrams specifying equations. There is a theory for groups, a theory for rings, and so on. We can specify algebraic structures of a given kind in some category C with finite products by a product-preserving functor $\mu : T \rightarrow C$. This is a simple and elegant form of *denotational* semantics. However, Lawvere theories know nothing of *operational* semantics. Our goal here is to address this using “enriched” Lawvere theories.

In a Lawvere theory, the objects are types and the morphisms are terms; however, there are no relations between terms, only equations. The process of computing one term into another should be given by hom-objects with more structure. In operational semantics, program behavior is often specified by labelled transition systems, or labelled directed multigraphs [26]. The edges of such a graph represent rewrites:

$$(\lambda x.x + x \ 2) \xrightarrow{\beta} 2 + 2 \xrightarrow{+} 4$$

We can use an enhanced Lawvere theory in which, rather than merely *sets* of morphisms, there are *graphs* or perhaps *categories*. Enriched Lawvere theories are exactly for this purpose.

In a theory T enriched in a category V of some kind of “directed object”, including graphs, categories, and posets, the theory has the following interpretation:

types:	objects of T
terms:	morphisms of T
equations between terms:	commuting diagrams
rewrites between terms:	“edges” in hom in V

To be clear, this is not a new idea. Using enriched Lawvere theories for operational semantics has been explored in the past. For example, category-enriched theories have been studied by Seely [29] for the λ -calculus, and poset-enriched ones by Ghani and Lüth [23] for understanding “modularity” in term rewriting systems. They have been utilized extensively by Power, enriching in ω -complete partial orders to study recursion [13] – in fact, there the simplified “natural number” enriched theories which we explore were implicitly considered.

The goal of this paper is to give a simple unified explanation of enriched Lawvere theories and some of their applications to operational semantics. We aim our explanations at readers familiar with category theory but not yet enriched categories. To reduce the technical overhead we only consider enrichment over cartesian closed categories.

In general for a cartesian closed category V , a V -**theory** is a V -enriched Lawvere theory with natural number arities. We consider V as a choice of “method of computation” for algebraic theories. The main idea of this paper is that product-preserving functors between enriching categories allow for the translation between different kinds of semantics. This translation could be called “change of computation”—or, following standard mathematical terminology, **change of base**.

Because operational semantics uses graphs to represent terms and rewrites, one might expect some category like Gph , the category of directed multigraphs, to be our main example of enriching category: that is, the “thing” of n -ary operations, or n -variable terms in a theory, is a directed graph whose edges are rewrites. This is known as *small-step* operational semantics, meaning each edge represents a single instance of a rewrite rule.

When studying formal languages, one wants to pass from this local view to a global view: given a term, one cares about its possible evolutions after not only one rewrite but any finite sequence of rewrites. We study how programs operate in finite time. In computer science, this corresponds to defining a rewrite relation and forming its transitive closure, called *big-step* operational semantics. This is the classic example which change of base aims to generalize.

However, there is a subtlety. We may try to model the translation from small-step to big-step operational semantics using the “free category” functor, which for any directed multigraph forms the category whose objects are vertices and morphisms are finite paths of edges. However, this functor does *not* preserve products. One might hope to cure this using a better-behaved variant of directed multigraphs, such as reflexive graphs. One advantage of reflexive graphs is that each vertex has a distinguished edge from it to itself; these describe rewrites that “do nothing”. Thus, in a product of reflexive graphs there are edges describing the process of rewriting one factor while doing nothing in the other. This lets us handle parallelism. Unfortunately, as we shall explain, the free category functor from reflexive graphs to categories still fails to preserve products.

To obtain a product-preserving change of base taking us from small-step to big-step operational semantics, it seems the cleanest solution is to generalize graphs to *simplicial sets*. A simplicial set is a contravariant functor from the category Δ of finite linear orders and monotone maps to the category of sets and functions. It can be visualized as a space built from “simplices”, which generalize triangles to any dimension: point, line, triangle, tetrahedron, etc. For an introduction to simplicial sets, see Friedman [12]. We use $sSet$ to denote the category of simplicial sets, namely $Set^{\Delta^{op}}$.

Simplicial sets allow one to generalize rewriting to *higher-dimensional* rewriting, but this is not our focus here. Indeed, we only need two facts about simplicial sets in this paper:

- There is a full and faithful embedding of RGph, the category of reflexive graphs, in $s\text{Set}$, so we can think of a reflexive graph as a special kind of simplicial set (namely one whose n -simplices for $n > 1$ are all degenerate).
- The free category functor $\text{FC}: s\text{Set} \rightarrow \text{Cat}$, often called “realization”, preserves products.

We thus obtain a spectrum of cartesian closed categories V to enrich over, each connected to the next by a product-preserving functor, which allow us to examine the computation of term calculi in various ways:

Simplicial Sets	$s\text{Set}$ -theories represent “small-step” operational semantics: — an edge is a <i>single</i> term rewrite.
Categories	Cat -theories represent “big-step” operational semantics: (Often this means a rewrite to a normal form. We use the term more generally.) — a morphism is a <i>finite sequence</i> of rewrites.
Posets	Pos -theories represent “full-step” operational semantics: — a boolean is the <i>existence</i> of a big-step rewrite.
Sets	Set-theories represent denotational semantics: — an element is a <i>connected component</i> of the rewrite relation.

In Section 2 we review Lawvere theories as a more explicit, but equivalent, presentation of finitary monads. In Section 3, we recall the basics of enrichment over cartesian closed categories. In Section 4 we give the central definition of V -theory, adapted from the work of Lucyshyn-Wright [22]. Using his work we show that a V -theory T gives a monadic adjunction between V and the V -category of models of T in V . This generalizes a fundamental result for Lawvere theories.

In Section 5 we discuss how suitable functors between enriching categories induce *change of base*: they transform theories, and their models, from one method of rewriting to another. Our main examples arise from this chain of adjunctions:

$$\begin{array}{ccccc} & \text{sSet} & \text{Cat} & \text{Pos} & \text{Set} \\ & \text{FC} \curvearrowleft \perp \curvearrowright \text{FP} & & \text{FS} \curvearrowleft \perp \curvearrowright \text{UP} & \\ & \text{UsS} \curvearrowleft \perp \curvearrowright \text{UC} & & & \end{array}$$

The right adjoints here automatically preserve finite products, but the left adjoints do as well, and these are what we really need:

- The functor $\text{FC}: s\text{Set} \rightarrow \text{Cat}$ maps a simplicial set (for example a reflexive graph) to the category it freely generates. Change of base along FC maps small-step operational semantics to big-step operational semantics.
- The functor $\text{FP}: \text{Cat} \rightarrow \text{Pos}$ maps a category C to the poset whose elements are objects of C , with $c \leq c'$ iff C has a morphism from c to c' . Change of base along FP maps big-step operational semantics to full-step operational semantics.
- The functor $\text{FS}: \text{Pos} \rightarrow \text{Set}$ maps a poset P to the set of “components” of P , where $p, p' \in P$ are in the same component if $p \leq p'$. Change of base along FS maps full-step operational semantics to denotational semantics.

In Section 6 we show that models of all V-theories for all enriching V can be assimilated into one category using the Grothendieck construction. In Section 7 we bring all the strands together and demonstrate these concepts in applications. First we consider the SKI-combinator calculus, and then we show how theories enriched over the category of labelled graphs can be used to study bisimulation.

Acknowledgements

This paper builds upon the ideas of Mike Stay and Greg Meredith presented in “Representing operational semantics with enriched Lawvere theories” [30]. We appreciate their offer to let us develop this work further for use in the innovative distributed computing system RChain, and gratefully acknowledge the support of Pyrofex Corporation. We also thank Richard Garner, Todd Trimble and others at the n -Category Café for classifying cartesian closed categories where every object is a finite coproduct of copies of the terminal object [2].

2 Lawvere Theories

Algebraic structures are traditionally treated as sets equipped with operations obeying equations, but we can generalize such structures to live in any category with finite products. For example, given any category C with finite products, we can define a monoid internal to C to consist of:

$$\begin{aligned} \text{an object } & M \\ \text{an identity element } & e: 1 \rightarrow M \\ \text{and multiplication } & m: M^2 \rightarrow M \\ \text{obeying the associative law } & m \circ (m \times M) = m \circ (M \times m) \\ \text{and the right and left unit laws } & m \circ (e \times \text{id}_M) = \text{id}_M = m \circ (\text{id}_M \times e). \end{aligned}$$

Lawvere theories formalize this idea. For example, there is a Lawvere theory $\text{Th}(\text{Mon})$, the category with finite products freely generated by an object t equipped with an identity element $e: 1 \rightarrow t$ and multiplication $m: t^2 \rightarrow t$ obeying the associative law and unit laws listed above. This captures the “Platonic idea” of a monoid internal to a category with finite products. A monoid internal to C then corresponds to a functor $\mu: \mathbf{T} \rightarrow \mathbf{C}$ that preserves finite products.

In more detail, let N be any skeleton of the category of finite sets FinSet . Because N is the free category with finite coproducts on 1, N^{op} is the free category with finite products on 1. A **Lawvere theory** is a category with finite products \mathbf{T} equipped with a functor $\tau: N^{\text{op}} \rightarrow \mathbf{T}$ that is the identity on objects and preserves finite products. Thus, a Lawvere theory is essentially a category generated by one object $\tau(1) = t$ and n -ary operations $t^n \rightarrow t$, as well as the projection and diagonal morphisms of finite products.

For efficiency let us call a functor that preserves finite products **cartesian**. Lawvere theories are the objects of a category Law whose morphisms are cartesian functors $f: \mathbf{T} \rightarrow \mathbf{T}'$ that obey $f\tau = \tau'$. More generally, for any category with finite products C, a **model** of the Lawvere theory \mathbf{T} in C is a cartesian functor $\mu: \mathbf{T} \rightarrow \mathbf{C}$. The models of \mathbf{T} in C are the objects of a category $\text{Mod}(\mathbf{T}, \mathbf{C})$, in which the morphisms are natural transformations.

A theory can thus have models in many different contexts. For example, there is a Lawvere theory $\text{Th}(\text{Mon})$, the theory of monoids, described as above. Ordinary monoids are models of this theory in Set , while topological monoids are models of this theory in Top .

For completeness, it is worthwhile to mention the *presentation* of a Lawvere theory: how exactly does the above “sketch” of $\text{Th}(\text{Mon})$ produce a category with finite products? It is precisely analogous to the

presentation of an algebra by generators and relations: we form the free category with finite products on the data given, and impose the required equations. The result is a category whose objects are powers of t , and whose morphisms are composites of products of the morphisms in $\text{Th}(\text{Mon})$, projections, deletions, symmetries and diagonals. A detailed account was given by Barr and Wells [4, Chap. 4].

In 1965, Linton [21] proved that Lawvere theories correspond to “finitary monads” on the category of sets. For every Lawvere theory T , there is an adjunction:

$$\begin{array}{ccc} & F & \\ \text{Set} & \perp & \text{Mod}(T, \text{Set}). \\ & U & \end{array}$$

The functor

$$U: \text{Mod}(T, \text{Set}) \rightarrow \text{Set}$$

sends each model μ to its underlying set, $X = \mu(\tau(1))$. Its left adjoint, the free model functor

$$F: \text{Set} \rightarrow \text{Mod}(T, \text{Set}),$$

sends each finite set $n \in \mathbb{N}$ to the representable functor $T(\tau(n), -): T \rightarrow \text{Set}$, and in general any set X to the colimit of all such representables as n ranges over the poset of finite subsets of X . In rough terms, $F(X)$ is the model of all n -ary operations from T on the set X .

If we momentarily abbreviate $\text{Mod}(T, \text{Set})$ as Mod , we obtain an adjunction

$$\text{Mod}(F(n), \mu) = \text{Mod}(T(\tau(n), -), \mu) \cong \mu(\tau(n)) \cong \mu(\tau(1))^n = \text{Set}(n, U(\mu))$$

where the left isomorphism arises from the Yoneda lemma, and the right isomorphism from the product preservation of μ .

This adjunction induces a monad T on Set :

$$T(X) = \int^{n \in \mathbb{N}} X^n \times T(n, 1). \quad (1)$$

The integral here is a coend, essentially a coproduct quotiented by the equations of the theory and the equations induced by the cartesian structure of the category. This forms the set of all terms that can be constructed from applying the operations to the elements, subject to the equations of the theory. The monad constructed this way is always **finitary**: that is, it preserves filtered colimits [1], or its action on sets is determined by its action on finite sets.

Conversely, for a monad T on Set , its Kleisli category $\text{Kl}(T)$ is the category of all free algebras of the monad, which has all coproducts. There is a functor $k: \text{Set} \rightarrow \text{Kl}(T)$ that is the identity on objects and preserves coproducts. Thus,

$$k^{\text{op}}: \text{Set}^{\text{op}} \rightarrow \text{Kl}(T)^{\text{op}}$$

is a cartesian functor, and restricting its domain to \mathbb{N}^{op} is a Lawvere theory k_T . To see what this is doing, note that:

$$\text{Kl}(T)^{\text{op}}(n, m) = \text{Kl}(T)(m, n) = \text{Set}(m, T(n))$$

where the latter is considered as m n -ary operations in the Lawvere theory k_T . When T is finitary, the monad arising from this Lawvere theory is naturally isomorphic to T itself.

This correspondence sets up an equivalence between the category Law of Lawvere theories and the category of finitary monads on Set. There is also an equivalence between the category $\text{Mod}(\mathbf{T}, \text{Set})$ of models of a Lawvere theory \mathbf{T} and the category of algebras of the corresponding finitary monad T . Furthermore, all this generalizes with Set replaced by any “locally finitely presentable” category [1]. For more details see [4, 20, 24].

One final point, provided to us by Mike Stay: while monads are often associated with functional programming languages such as Haskell, Lawvere theories correspond to *interfaces* or abstract classes in object-oriented programming. In these one declares various constants, types, and abstract functions satisfying tests, and then one implements the interface by assigning these elements, sets, functions, and equations—precisely a model in Set. While people think of monads as the main example of “categories in programming”, in fact Lawvere theories are ubiquitous.

3 Enrichment

To incorporate the aspect of computation, we now turn to Lawvere theories that have hom-*objects* rather than mere hom-*sets*. To do this we use enriched category theory [18] and replace sets with objects of a cartesian closed category V , called the “enriching” category or “base”. A V -enriched category or **V -category** C is:

a collection of objects	$\text{Ob}(C)$
a hom-object function	$C(-, -) : \text{Ob}(C) \times \text{Ob}(C) \rightarrow \text{Ob}(V)$
composition morphisms	$\circ_{a,b,c} : C(b, c) \times C(a, b) \rightarrow C(a, c) \quad \forall a, b, c \in \text{Ob}(C)$
identity-assigned morphisms	$id_a : 1_V \rightarrow C(a, a) \quad \forall a \in \text{Ob}(C)$

such that composition is associative and unital. A **V -functor** $F : C \rightarrow D$ is:

a function	$F : \text{Ob}(C) \rightarrow \text{Ob}(D)$
a collection of morphisms	$F_{ab} : C(a, b) \rightarrow D(F(a), F(b)) \quad \forall a, b \in C$

such that F preserves composition and identity. A **V -natural transformation** $\alpha : F \Rightarrow G$ is:

$$\text{a family } \alpha_a : 1_V \rightarrow D(F(a), G(a)) \quad \forall a \in \text{Ob}(C)$$

such that α is “natural” in a : an evident square commutes. There is a 2-category VCat of V -categories, V -functors, and V -natural transformations.

We can construct new V -categories from old ones by taking products or opposites, in obvious ways. There is also a V -category denoted \underline{V} with the same objects as V and with hom-objects given by the internal hom:

$$\underline{V}(v, w) = w^v \quad \forall v, w \in V.$$

The concepts of adjunction and monad generalize straightforwardly to V -categories, and when we speak of an adjunction or monad in the enriched context this generalization is what we mean [18]. For example, there is an adjunction

$$\underline{V}(u \times v, w) \cong \underline{V}(u, w^v)$$

called “currying”.

We can generalize products and coproducts to the enriched context. Given a V -category C , the **V -coproduct** of an n -tuple of objects $b_1, \dots, b_n \in \text{Ob}(C)$ is an object b equipped with a V -natural isomorphism

$$C(b, -) \cong \prod_{i=1}^n C(b_i, -).$$

If such an object exists, we denote it by $\sum_{i=1}^n b_i$. This makes sense even when $n = 0$: a 0-ary V -coproduct in C is called a **V -initial object** and denoted as 0_C . When V is cartesian closed, any finite coproduct that exists in V is also a V -coproduct in \underline{V} . In particular,

$$u^{v+w} \cong u^v \times u^w \text{ and } w^0 \cong 1_V$$

whenever 0 is an initial object of V . Conversely, any finite V -coproduct that exists in V is also a coproduct in the usual sense.

Similarly, a **V -product** of objects $b_1, \dots, b_n \in \text{Ob}(C)$ is an object b equipped with a V -natural isomorphism

$$C(-, b) \cong \prod_{i=1}^n C(-, b_i). \quad (2)$$

If such an object b exists, we denote it by $\prod_{i=1}^n b_i$. A 0-ary product in C is called a **V -terminal object** and denoted as 1_C . Whenever V is cartesian closed, the finite products in V are also V -products in \underline{V} . In particular,

$$(u \times v)^w \cong u^w \times v^w \text{ and } 1_V^w \cong 1_V$$

where our chosen terminal object 1_V is also V -terminal. Conversely, any finite V -product in V is also a product in the usual sense.

A general V -category C does not exactly have projections from a V -product to its factors, since given two objects $c, c' \in \text{Ob}(C)$ there is not, fundamentally, a *set* of morphisms from c to c' . Instead there is the hom-object $C(c, c')$, which is an object of V . However, any object v of V has a set of **elements**, namely morphisms $f: 1_V \rightarrow v$. Elements of $C(c, c')$ act like morphisms from c to c' .

In particular, any V -product $b = \prod_{i=1}^n b_i$ gives rise to elements

$$p_i: 1_V \rightarrow C(b, b_i)$$

which serve as substitutes for the projections in a usual product. These elements are defined as composites

$$1_V \xrightarrow{id_b} C(b, b) \xrightarrow{\sim} \prod_{i=1}^n C(b, b_i) \rightarrow C(b, b_i)$$

where the isomorphism comes from Eq. (2) and the last arrow is a projection in V .

Even better, we can bundle up all these elements p_i into a single element

$$p: 1_V \rightarrow \prod_{i=1}^n C(b, b_i)$$

which serves as a substitute for the universal cone in a usual product. Starting from p we can recover the V -natural isomorphism in Eq. (2) as follows:

$$C(-, b) \xrightarrow{\sim} 1_V \times C(-, b) \xrightarrow{p \times 1} \prod_{i=1}^n C(b, b_i) \times C(-, b) \longrightarrow \prod_{i=1}^n C(-, b_i) \quad (3)$$

where the last arrow is given by composition. Thus, we say a **universal cone** exhibiting b as the V -product of objects b_1, \dots, b_n is an element $p: 1_V \rightarrow \prod_{i=1}^n C(b, b_i)$ such that the V -natural transformation $C(-, b) \rightarrow \prod_{i=1}^n C(-, b_i)$ given by Eq. (3) is an isomorphism.

The advantage of this reformulation is that we can say a V -functor $F: C \rightarrow D$ **preserves finite V -products** if for every universal cone $p: 1_V \rightarrow \prod_{i=1}^n C(b, b_i)$ exhibiting b as the V -product of the objects b_i , the composite

$$1_V \xrightarrow{p} \prod_{i=1}^n C(b, b_i) \xrightarrow{\prod_i F} D(F(b), F(b_i))$$

is universal cone exhibiting $F(b)$ as the V -product of the objects $F(b_i)$.

A bit more subtly, generalizing the exponentials in V , a V -category C can have “powers”. Given $v \in \text{Ob}(V)$, we say an object $c^v \in \text{Ob}(C)$ is a v -**power** of $c \in \text{Ob}(C)$ if it is equipped with a V -natural isomorphism

$$C(-, c^v) \cong C(-, c)^v. \quad (4)$$

In the special case $V = \text{Set}$ this forces c^v to be the v -fold product of copies of c . As with V -products, it is useful to repackage the isomorphism of Eq. (4) so we can say what it means for a V -functor to preserve v -powers. First, note that this isomorphism gives rise to an element

$$q: 1_V \rightarrow C(c^v, c)^v,$$

namely the composite

$$1_V \xrightarrow{id_{c^v}} C(c^v, c^v) \xrightarrow{\sim} C(c^v, c)^v.$$

Conversely, any element $q: 1_V \rightarrow C(c^v, c)^v$ determines a V -natural transformation $e: C(-, c^v) \rightarrow C(-, c)^v$, and we say e is a **universal cone** if this V -natural transformation is an isomorphism. Next, suppose C and D are V -categories with v -powers. We say a V -functor $F: C \rightarrow D$ **preserves v -powers** if it maps universal cones to universal cones.

There are just a few more technicalities. A category is **locally finitely presentable** if it is the category of models for a finite limits theory, and an object is **finite** if its representable functor is **finitary**: that is, it preserves filtered colimits [1]. A V -category C is **locally finitely presentable** if its underlying category C_0 is locally finitely presentable, C has finite powers, and $(-)^x: C_0 \rightarrow C_0$ is finitary for all finitely presentable x . The details are not crucial here: all categories to be considered are locally finitely presentable. We will use V_f to denote the full subcategory of V of finite objects: in sSet , these are simplicial sets with finitely many n -simplices for each n .

4 Enriched Lawvere Theories

Power introduced the notion of enriched Lawvere theory about twenty years ago, “in seeking a general account of what have been called notions of computation” [27]. The original definition is as follows: for a symmetric monoidal closed category $(V, \otimes, 1)$, a “ V -enriched Lawvere theory” is a V -category T that has powers by objects in V_f , equipped with an identity-on-objects V -functor

$$\tau: V_f^{\text{op}} \rightarrow T$$

that preserves these powers. A “model” of a V -theory is a V -functor $\mu: T \rightarrow V$ that preserves powers by finite objects of V . There is a category $\text{Mod}(T, V)$ whose objects are models and whose morphisms

are V -natural transformations. The monadic adjunction and equivalence of Section 2 generalize to the enriched setting.

In this paper, however, we only consider *natural number* arities, while still retaining enrichment. To do this we use the work of Lucyshyn-Wright [22], who along with Power [25] has generalized Power's original ideas to allow a more flexible choice of arities. We also limit ourselves to the case where the tensor product of V is cartesian. This has a significant simplifying effect, yet it suffices for many cases of interest in computer science.

Thus, in all that follows, we let $(V, \times, 1_V)$ be a cartesian closed category equipped with chosen finite coproducts of the terminal object 1_V , say

$$n_V = \sum_{i \in n} 1_V.$$

Define N_V to be the full subcategory of V containing just these objects n_V . There is also a V -category \underline{N}_V whose objects are those of N_V and whose hom-objects are given as in V . We define the V -category of **arities** for V to be

$$A_V := \underline{N}_V^{\text{op}}.$$

We shall soon see that A_V has finite V -products.

Definition 1. We define a V -theory (T, τ) to be a V -category T equipped with a V -functor

$$\tau: A_V \rightarrow T$$

that is the identity on objects and preserves finite V -products.

Definition 2. A **model** of T in a V -category C is a V -functor

$$\mu: T \rightarrow C$$

that preserves finite V -products.

Just as all the objects of a Lawvere theory are finite products of a single object, we shall see that all the objects of T are finite V -products of the object

$$t = \tau(1_V).$$

Definition 3. We define $V\text{Law}$, the **category of V -theories**, to be the category for which an object is a V -theory and a morphism from (T, τ) to (T', τ') is a V -functor $f: T \rightarrow T'$ that preserves finite V -products and has $f\tau = \tau'$.

Definition 4. For every V -theory (T, τ) and every V -category C with finite V -products, we define $\text{Mod}(T, C)$, the **category of models** of (T, τ) in C , to be the category for which an object is a V -functor $\mu: T \rightarrow C$ that preserves finite V -products and a morphism is a V -natural transformation.

The basic monadicity results for Lawvere theories generalize to V -theories when V is complete and cocomplete, as in the main examples we consider: $V = \text{sSet}, \text{Cat}, \text{Pos}$, and Set . Under this extra assumption $V\text{Law}$ and $\text{Mod}(T, C)$ can be promoted to V -categories, which we call $\underline{V}\text{Law}$ and $\underline{\text{Mod}}(T, C)$. Furthermore, there is a V -functor

$$U: \underline{\text{Mod}}(T, V) \rightarrow V$$

sending any model $\mu: T \rightarrow V$ to its underlying object $\mu(t) \in V$. Recall that monads and adjunctions make sense in $V\text{Cat}$, just as they do in Cat . The V -functor U has a left adjoint

$$F: V \rightarrow \underline{\text{Mod}}(T, V),$$

and $\underline{\text{Mod}}(T, V)$ is equivalent to the V -category of algebras of the resulting monad $T = UF$. More precisely:

Theorem 5. Suppose V is cartesian closed, complete and cocomplete, and has chosen finite coproducts of the terminal object. Let (T, τ) be a V -theory. Then there is a monadic adjunction

$$\begin{array}{ccc} \underline{V} & \perp & \underline{\text{Mod}}(T, V). \\ F \swarrow \quad \downarrow & \perp & \searrow U \\ \uparrow & \perp & \downarrow \\ U & \perp & \underline{\text{Mod}}(T, V). \end{array}$$

Proof. This follows from Lucyshyn-Wright's general theory [22], so our task is simply to explain how. He allows V to be a symmetric monoidal category, and uses a more general concept of algebraic theory with a system of arities given by any fully faithful symmetric monoidal V -functor $j: J \rightarrow \underline{V}$. For us $J = \underline{N}_V$ and $j: \underline{N}_V \rightarrow \underline{V}$ is the obvious inclusion; this is his Example 3.7.

Lucyshyn-Wright defines a **J -theory** to be a V -functor $\tau: J^{\text{op}} \rightarrow T$ that is the identity on objects and preserves powers by objects in J (or more precisely, their images under j). For us $J^{\text{op}} = A_V$. So, to apply his theory, we need to show that a V -functor $\tau: A_V \rightarrow T$ preserves powers by objects in N_V if and only if it preserves finite V -products. This is Lemma 16 below.

He defines a model (or “algebra”) of a J -theory to be a V -functor $\mu: T \rightarrow \underline{V}$ that preserves powers by objects in J . He defines a morphism of models to be a V -natural transformation between such V -functors. So, to apply his theory, we also need to show that when $J = \underline{N}_V$, a V -functor $\mu: T \rightarrow \underline{V}$ preserves powers by objects of J if and only if it preserves finite V -products. This is Lemma 17 below.

A technical concept fundamental to Lucyshyn-Wright's theory is that of an **eleutheric** system of arities $j: J \rightarrow \underline{V}$. This is one where the left Kan extension of any V -functor $f: J \rightarrow \underline{V}$ along j exists and is preserved by each V -functor $\underline{V}(x, -): \underline{V} \rightarrow \underline{V}$. In Example 7.5.5 he shows that $j: \underline{N}_V \rightarrow \underline{V}$ is eleutheric when V is countably cocomplete. In Thm. 8.9 shows that when $j: J \rightarrow \underline{V}$ is eleutheric, and has equalizers, we may form the V -category $\underline{\text{Mod}}(T, V)$, and that the forgetful V -functor

$$U: \underline{\text{Mod}}(T, V) \rightarrow \underline{V}$$

is monadic. This is the result we need. So, our theorem actually holds whenever V is cartesian closed, with equalizers and countable colimits, and has chosen finite coproducts of the initial object. \square

Before turning to examples, a word about Lucyshyn-Wright's construction of the left adjoint F and the monad T is in order. These rely on the “free model” on an object $n_V \in V$. This is the enriched generalization of the free model described in Section 2: it is the composite of $\tau^{\text{op}}: A_V^{\text{op}} \rightarrow T^{\text{op}}$ with the enriched Yoneda embedding $y: T^{\text{op}} \rightarrow [\mathbf{T}, V]$:

$$\begin{aligned} A_V^{\text{op}} &\xrightarrow{\tau^{\text{op}}} T^{\text{op}} \xrightarrow{y} [\mathbf{T}, V] \\ n_V &\mapsto t^{n_V} \mapsto T(t^{n_V}, -) \end{aligned}$$

Since an object of V does not necessarily have a “poset of finite subobjects” over which to take a filtered colimit (as in Set), the extension of this “free model” functor $y\tau^{\text{op}}$ to all of V is specified by a somewhat higher-powered generalization: it is the left Kan extension of $y\tau^{\text{op}}$ along j .

$$\begin{array}{ccc} N_V & \xrightarrow{y\tau^{\text{op}}} & [\mathbf{T}, V] \\ j \searrow & \Downarrow \eta & \nearrow \tau \\ & V & \end{array}$$

$F := \text{Lan}_j y\tau^{\text{op}}$

This is the universal “best solution” to the problem of making the triangle commute up to a \mathbf{V} -natural transformation. That is, for any functor $G: \mathbf{V} \rightarrow [\mathbf{T}, \mathbf{V}]$ and \mathbf{V} -natural transformation $\theta: y\tau^{\text{op}} \Rightarrow Gj$, the latter factors uniquely through η . From the adjunction between \mathbf{V} and the category of models $\text{Mod}(\mathbf{T}, \mathbf{V})$ we obtain a \mathbf{V} -enriched monad

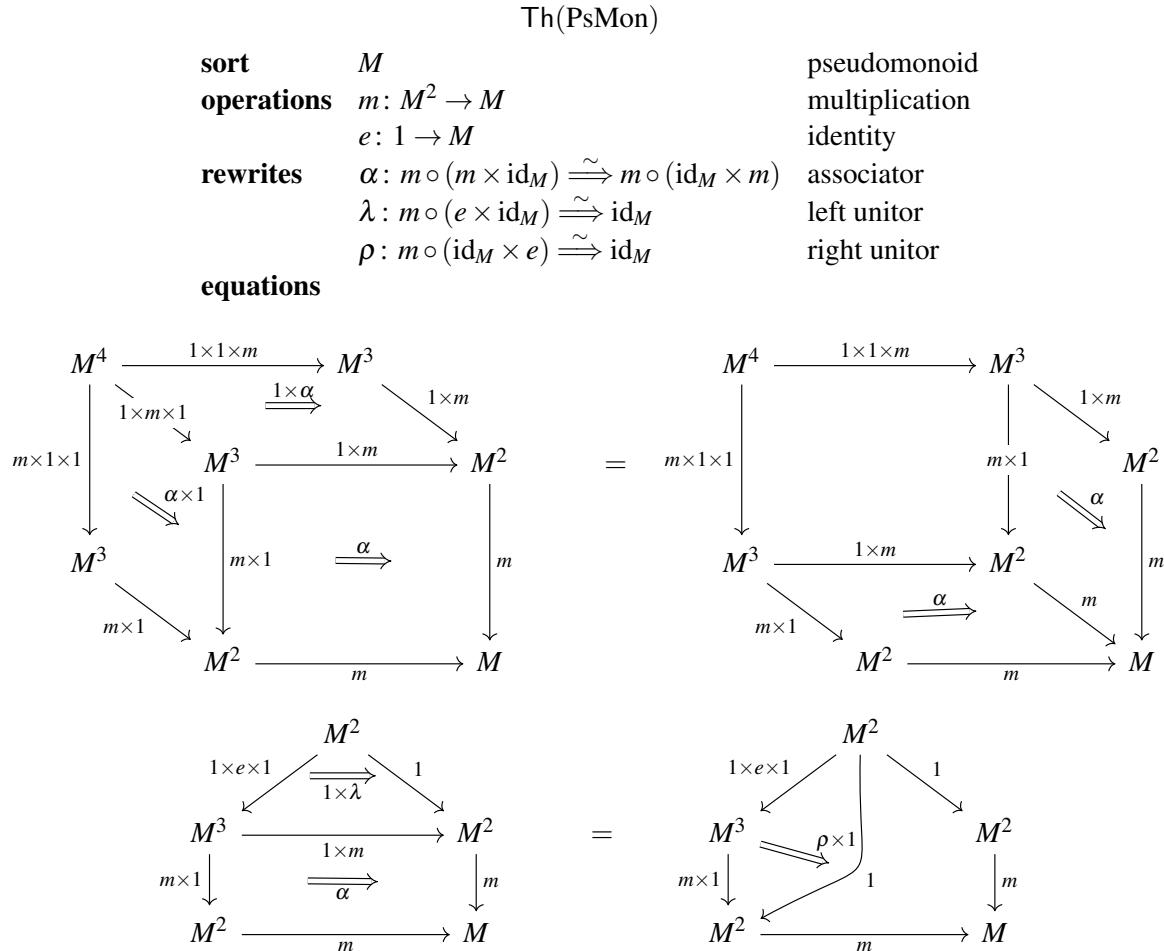
$$T = UF: \mathbf{V} \rightarrow \mathbf{V},$$

and this has a more concrete formula as an enriched coend:

$$T(V) = \int^{n_{\mathbf{V}} \in \mathbb{N}_{\mathbf{V}}} V^{n_{\mathbf{V}}} \times \mathbf{T}(t^{n_{\mathbf{V}}}, t).$$

We next give two examples of a rather abstract nature, where we show how Cat -enriched Lawvere theories can describe categories with extra structure. In Section 7 we study examples more directly connected to operational semantics.

Example 6. When $\mathbf{V} = \text{Cat}$, a \mathbf{V} -category is a 2-category, so a \mathbf{V} -theory deserves to be called a **2-theory**. For example, let $\mathbf{T} = \text{Th}(\text{PsMon})$ be the 2-theory of pseudomonoids. A pseudomonoid [9] is a weakened version of a monoid: rather than associativity and unitality *equations*, it has 2-isomorphisms called the associator and unitors, which we can treat as *rewrite rules*. To equate various possible rewrite sequences, these 2-isomorphisms must obey equations called “coherence laws”. Power [19] has introduced “enriched sketches” as a way of presenting enriched Lawvere theories. Informally, here is a presentation of the 2-theory for pseudomonoids:



We write the equations as commutative diagrams merely for convenience; they could also be written as equations in a more traditional style. The top diagram expresses the pentagon identity for the associator, while the bottom one expresses the usual coherence law involving the left and right unitors.

Models of $T = \text{Th}(\text{PsMon})$ in Cat are monoidal categories: let us explore this example in more detail. A model of T is a finite-product-preserving 2-functor $\mu: T \rightarrow \text{Cat}$, which sends

$$\begin{aligned} t &\mapsto C \\ m &\mapsto \otimes: C^2 \rightarrow C \\ e &\mapsto I: 1 \rightarrow C \\ \alpha &\mapsto a: \otimes \circ (\otimes \times 1_C) \Rightarrow \otimes \circ (1_C \times \otimes) \\ \lambda &\mapsto \ell: I \circ 1_C \Rightarrow 1_C \\ \rho &\mapsto r: 1_C \circ I \Rightarrow 1_C \end{aligned}$$

such that the coherence laws of the rewrites are preserved. Thus, a model is a category equipped with a tensor product \otimes and unit object I such that these operations are associative and unital up to natural isomorphism; so these models are precisely monoidal categories.

Given two models $\mu, \nu: T \rightarrow \text{Cat}$, a morphism of models is a 2-natural transformation $\varphi: \mu \Rightarrow \nu$; this amounts to a strict monoidal functor $\varphi: (C, \otimes_C, I_C) \rightarrow (D, \otimes_D, I_D)$. The strictness arises because morphisms between models are 2-natural transformations rather than pseudonatural transformations. There is a substantial amount of theory on pseudomonads and pseudoalgebras [6, 10], but to the authors' knowledge the theory-monad correspondence has not yet been extended to include the case of weak naturality.

Finally, because Cat is complete and cocomplete, the category of models $\text{Mod}(T, \text{Cat})$ can be promoted to a 2-category $\underline{\text{Mod}}(T, \text{Cat})$. This is the 2-category of monoidal categories, strict monoidal functors, and monoidal natural transformations.

We can accomplish the same thing on the monad side: a Cat -enriched monad is called a **2-monad**, and T gives rise to the “free monoidal category” 2-monad T on Cat [6]. To apply this 2-monad to $C \in \text{Cat}$ we first form the free model on C by taking a left Kan extension as above, and then evaluate this model at the generating object. In the same way that the (underlying set of the) free monoid on a set X consists of all finite strings of elements of X , $T(C)$ is the monoidal category consisting of all finite tensor products of objects of C and all morphisms built from those of C by composition and tensoring together with associators and unitors obeying the necessary coherence laws. Morphisms of these algebras are strict monoidal functors, while 2-morphisms are natural transformation. We thus have a 2-equivalence between $\underline{\text{Mod}}(T, \text{Cat})$ and the 2-category of algebras of T .

In this way, 2-theories generalize equipping *set*-like objects with operations obeying equations to equipping *category*-like objects with operations obeying equations up to transformations that obey equations of their own. In particular, this gives us a way to present graphical calculi such as string diagrams – the language of monoidal categories.

Example 7. Enrichment generalizes operations in more ways than by weakening equations to coherent isomorphisms. We can also use 2-theories to describe other structures that make sense inside 2-categories, such as adjunctions.

For example, we may define a cartesian category X to be one equipped with right adjoints to the diagonal $\Delta_X: X \rightarrow X \times X$ and the unique functor $!_X: X \rightarrow 1_{\text{Cat}}$. These right adjoints are a functor $m: X^2 \rightarrow X$ describing binary products in X and a functor $e: 1 \rightarrow X$ picking out the terminal object in X . We can capture the fact that they are right adjoints by providing them with units and counits and imposing the triangle equations. There is thus a 2-theory $\text{Th}(\text{Cart})$ whose models in Cat are categories with chosen

finite products. More generally a model of this 2-theory in any 2-category C with finite products is called a **cartesian object** in C .

$\text{Th}(\text{Cart})$		
type	X	cartesian object
operations	$m: X^2 \rightarrow X$ $e: 1 \rightarrow X$	product terminal element
rewrites	$\Delta: \text{id}_X \Rightarrow m \circ \Delta_X$ $\pi: \Delta_X \circ m \Rightarrow \text{id}_{X^2}$ $\top: \text{id}_X \Rightarrow e \circ !_X$ $\varepsilon: !_X \circ e \Rightarrow \text{id}_1$	unit of adjunction between m and Δ_X counit of adjunction between m and Δ_X unit of adjunction between e and $!_X$ counit of adjunction between e and $!_X$
equations		
	$\begin{array}{ccc} \Delta_X & \begin{array}{c} \Downarrow \\ \Delta_X \circ \Delta \end{array} & \xrightarrow{1} \\ \Delta_X \circ m \circ \Delta_X & \xrightarrow{\pi \circ \Delta_X} & \Delta_X \end{array}$	$\begin{array}{ccc} m & \begin{array}{c} \Downarrow \\ \Delta \circ m \end{array} & \xrightarrow{1} \\ m \circ \Delta_X \circ m & \xrightarrow{m \circ \pi} & m \end{array}$
	$\begin{array}{ccc} !_X & \begin{array}{c} \Downarrow \\ !_X \circ \top \end{array} & \xrightarrow{1} \\ !_X \circ e \circ !_X & \xrightarrow{\varepsilon \circ !_X} & !_X \end{array}$	$\begin{array}{ccc} e & \begin{array}{c} \Downarrow \\ \top \circ e \end{array} & \xrightarrow{1} \\ e \circ !_X \circ e & \xrightarrow{e \circ \varepsilon} & e \end{array}$

Again we write the equations as commutative diagrams, but this time commutative triangles of 2-morphisms in $\text{Th}(\text{Cart})$. These are the triangle equations that force m to be the right adjoint of Δ_X and e to be the right adjoint of $!_X$. A model of $\text{Th}(\text{Cart})$ is a category with chosen binary products and a chosen terminal object; morphisms in $\text{Mod}(\text{Th}(\text{Cart}), \text{Cat})$ are functors that strictly preserve this extra structure.

The subtle interplay between the cartesian structure of $\text{Th}(\text{Cart})$ and the cartesian structure of the object $X \in \text{Th}(\text{Cart})$ is an example of the “microcosm principle”: objects with a given structure are most generally defined in a context that has the same sort of structure. As seen in the previous example, we can also define pseudomonoids in any 2-category with finite products, but this is excess to requirements: one can in fact define them more generally in any monoidal 2-category [9].

In fact, if we let arities be finite categories, we would have Cat -theories of categories with finite limits and colimits. However, for the purposes of this paper we are using only natural number arities. This suffices for constructing $\text{Th}(\text{Cart})$ and also $\text{Th}(\text{CoCart})$, the theory of categories with chosen binary coproducts and a chosen initial object. Various other kinds of categories—distributive categories, rig categories, etc.—can also be expressed using Cat -theories with natural number arities. This gives a systematic formalization of these categories, internalizes them to new contexts, and allows for the generation of 2-monads that describe them.

5 Change of Base

We now have the tools to formulate the main idea: a choice of enrichment for Lawvere theories corresponds to a choice of *computation*, and changing enrichments corresponds to a *change of computation*. We propose a general framework in which one can translate between different forms of computation: small-step, big-step, full-step operational semantics, and denotational semantics.

5.1 General results

Suppose that V and W are enriching categories of the sort we are considering: cartesian closed categories equipped with chosen finite coproducts of the terminal object. Suppose $F: V \rightarrow W$ preserves finite products. This induces a **change of base** functor $F_*: \text{VCat} \rightarrow \text{WCat}$ [7] which takes any V -category C and produces a W -category $F_*(C)$ with the same objects but with

$$F_*(C)(a, b) := F(C(a, b))$$

for all objects a, b . Composition in $F_*(C)$ is defined by

$$F(C(b, c)) \times F(C(a, b)) \xrightarrow{\sim} F(C(b, c) \times C(a, b)) \xrightarrow{F(\circ_{a, b, c})} F(C(a, b)).$$

The identity-assigning morphisms are given by

$$1_W \xrightarrow{\sim} F(1_V) \xrightarrow{F(id_a)} F(C(a, a)).$$

Moreover, if $f: C \rightarrow D \in \text{VCat}$ is a V -functor, there is a W -functor $F_*(f): F_*(C) \rightarrow F_*(D)$ that on objects equals f and on hom-objects equals $F(f)$. If $\alpha: f \Rightarrow g$ is a V -natural transformation and $c \in \text{Ob}(C)$, then we define

$$F_*(\alpha)_c: 1_W \xrightarrow{\sim} F(1_V) \xrightarrow{F(\alpha_c)} F(D(f(c), g(c))).$$

Thus, change of base actually gives a 2-functor from the 2-category of V -categories, V -functors and V -natural transformations to the corresponding 2-category for W .

In fact, the change of base operation gives a 2-functor

$$\begin{array}{ccc} \text{CartCat} & \xrightarrow{(-)_*} & \text{2Cat} \\ (F: V \rightarrow W) & \mapsto & (F_*: \text{VCat} \rightarrow \text{WCat}) \end{array}$$

where CartCat is the 2-category of cartesian closed categories equipped with chosen finite coproducts of the terminal object, finite product preserving functors preserving these chosen finite coproducts, and natural transformations. In particular, if V has not just finite coproducts of the terminal object but all coproducts of this object, there is a map of adjunctions

$$\begin{array}{ccccc} & \text{Set} & \xleftarrow{\perp} & V & \xleftarrow{\quad} \text{Cat} \xrightarrow{\perp} \text{VCat.} \\ & \swarrow \perp \curvearrowright & & \xleftarrow{V(1, -)} & \swarrow \perp \curvearrowright \\ & & & & \xleftarrow{(V(1, -))_*} \end{array}$$

Each set X is mapped to the X -indexed coproduct of the terminal object in V and conversely each object v of V is represented in Set by the hom-set from the unit to v . The latter induces the “underlying category” change of base, which forgets the enrichment. The former induces the “free V -enrichment” change of base, whereby ordinary Set-categories are converted to V -categories, denoted $C \mapsto \underline{C}$. These form an adjunction, because 2-functors preserve adjunctions.

We now study how change of base affects theories and their models. We start by asking when a functor $F : V \rightarrow W$ induces a change of base $F_* : V\text{Cat} \rightarrow W\text{Cat}$ that “preserves enriched theories”. That is, given a V -theory

$$\tau : A_V \rightarrow T$$

we want to determine conditions for the base-changed functor

$$F_*(\tau) : F_*(A_V) \rightarrow F_*(T)$$

to induce a W -theory in a canonical way. Recall that we require V and W to be cartesian closed, equipped with chosen finite coproducts of their terminal objects. We thus expect the following conditions to be sufficient: F should be cartesian, and it should preserve the chosen finite coproducts of the terminal object:

$$F(n_V) = n_W$$

for all n .

Given these conditions there is a W -functor, in fact an isomorphism

$$\tilde{F} : A_W \rightarrow F_*(A_V).$$

On objects this maps n_W to n_V , and on hom-objects it is simply the identity from

$$A_W(m_W, n_W) = n_W^{m_W} = (n^m)_W$$

to

$$F(A_V(m_V, n_V)) = F(n_V^{m_V}) = F((n^m)_V) = (n^m)_W$$

where we use Lemma 13 in these computations.

Using this we obtain a composite W -functor

$$A_W \xrightarrow{\tilde{F}} F_*(A_V) \xrightarrow{F_*(\tau_V)} F_*(T).$$

This is the identity on objects and preserves finite V -products because each of the factors has these properties. It is thus a W -theory.

Theorem 8. Let V, W be cartesian closed categories with chosen finite coproducts of their terminal objects, and let $F : V \rightarrow W$ be a cartesian functor that preserves these chosen coproducts. Then F_* preserves enriched theories: that is, for every V -theory $\tau_V : A_V \rightarrow T$, the W -functor

$$\tau_W := F_*(\tau_V) \circ \tilde{F} : A_W \rightarrow F_*(T)$$

is a W -theory. Moreover, F_* preserves models: for every model $\mu : T \rightarrow C$ of (T, τ_V) , the W -functor $F_*(\mu) : F_*(T) \rightarrow F_*(C)$ is a model of $(F_*(T), \tau_W)$.

Proof. We have shown the first part. For the second, by Lemma 17 it suffices to assume that μ preserves finite N_V -powers and check that $F_*(\mu)$ preserves N_W -powers. We leave this as an exercise to the reader. \square

Hence, any cartesian functor that preserves chosen finite coproducts of the terminal object gives a change of base. It thus provides for a method of translating formal languages between various “modes of operation”. Moreover, this reasoning generalizes to *multisorted* V-theories, enriched theories which have multiple sorts: given any $n \in \mathbb{N}$, the monoidal subcategory $(\text{N}_V)^n$ is also an eleutheric system of arities, so Lucyshyn-Wright’s monadicity theorem still applies.

5.2 Examples

Now let us look at some examples. The most important changes of base are the left adjoints in this diagram from Sec. 1:

$$\begin{array}{ccccc} & \text{FC} & \text{FP} & \text{FS} & \\ \text{sSet} & \begin{array}{c} \swarrow \perp \searrow \\ \text{UsS} \end{array} & \text{Cat} & \begin{array}{c} \swarrow \perp \searrow \\ \text{UC} \end{array} & \text{Pos} & \begin{array}{c} \swarrow \perp \searrow \\ \text{UP} \end{array} & \text{Set} \end{array}$$

The first step describes the translation from small-step to big-step operational semantics. As already mentioned, we need to use simplicial sets rather than graphs; let us now say more about why.

A first attempt might use directed multigraphs. Such graphs have directed edges and allow multiple edges between any pair of vertices. The category Gph of directed multigraphs is Set^G where G is the category with two objects v and e and two morphisms $s, t: e \rightarrow v$. The “free category” functor $F: \text{Gph} \rightarrow \text{Cat}$ gives for every graph G a category $F(G)$ as follows:

$$\begin{array}{lll} \text{objects} & \text{vertices of } G \\ \text{morphisms} & (e_1, e_2, \dots, e_n): s(e_1) \rightarrow t(e_n) : \forall i < n \quad t(e_i) = s(e_{i+1}) \\ \text{composition} & (e_1, e_2, \dots, e_m) \circ (e'_1, e'_2, \dots, e'_n) = (e'_1, \dots, e'_n, e_1, \dots, e_m) : t(e'_n) = s(e_1). \end{array}$$

The morphisms in $F(G)$ are called **edge paths**. Just as an edge describes a single rewrite in small-step operational semantics, an edge path describes a sequence of rewrites in big-step operational semantics. The edge paths with $n = 0$ serve as identity morphisms.

Unfortunately, $F: \text{Gph} \rightarrow \text{Cat}$ does not preserve products, so it is not a valid base change. To see this, let G_1 be $\{0 \xrightarrow{e} 1\}$, the graph with two vertices and one edge. The product $G_1 \times G_1$ looks like this:

$$\begin{array}{ccc} (0,0) & & (0,1) \\ & \searrow^{(e,e)} & \\ (1,0) & & (1,1). \end{array}$$

Thus the free category $F(G_1 \times G_1)$ has just one non-identity morphism. On the other hand $F(G_1) \times F(G_1)$ has five non-identity morphisms, shown here:

$$\begin{array}{ccc} (0,0) & \xrightarrow{(\text{id}_0, e)} & (0,1) \\ (e, \text{id}_0) \downarrow & \searrow^{(e,e)} & \downarrow (e, \text{id}_1) \\ (1,0) & \xrightarrow{(\text{id}_1, e)} & (1,1). \end{array}$$

where we write id for identity morphisms and e for the edge path consisting of the single edge e . Note that the triangles in this diagram commute. In terms of rewriting, the category $F(G_1 \times G_1)$ only allows

the rewrite $e: 0 \rightarrow 1$ to occur simultaneously in both factors, while $F(G_1) \times F(G_1)$ allows it to occur independently in either factor, in a commuting way.

To solve this problem one, might try to use reflexive graphs. Such graphs have directed edges and allows multiple edges between any pair of vertices; further, each vertex v is equipped with a distinguished **identity edge** $i(v)$ from v to itself. The category RGph of reflexive graphs is Set^R , where R is the category with two objects v and e , two morphisms $s, t: e \rightarrow v$, and a morphism $i: v \rightarrow e$ obeying $si = ti = 1_v$. There is a free category functor $F': \text{RGph} \rightarrow \text{Cat}$, which is like the free category functor for Gph except that we identify an edge path (e_1, \dots, e_n) with the same path having e_i omitted when e_i is an identity edge. Thus, the identity edges of a reflexive graph R become identity morphisms in $F'(R)$.

The advantage of reflexive graphs is that they allow rewrites in a product to occur independently in either factor. For example, let R_1 be the reflexive graph with two vertices and one non-identity edge, $\{0 \xrightarrow{e} 1\}$ (where we do not draw identity edges). The product $R_1 \times R_1$ has five non-identity edges:

$$\begin{array}{ccc} (0,0) & \xrightarrow{(i(0),e)} & (0,1) \\ (e,i(0)) \downarrow & \searrow (e,e) & \downarrow (e,i(1)) \\ (1,0) & \xrightarrow{(i(1),e)} & (1,1). \end{array}$$

Thus, the free category $F'(R_1 \times R_1)$ has two *noncommuting* triangles. On the other hand, $F'(R_1) \times F'(R_1)$ is the product of the category with a single non-identity morphism $e: 0 \rightarrow 1$ with itself, so it is this category:

$$\begin{array}{ccc} (0,0) & \xrightarrow{(\text{id}_0,e)} & (0,1) \\ (e,\text{id}_0) \downarrow & \searrow (e,e) & \downarrow (e,\text{id}_1) \\ (1,0) & \xrightarrow{(\text{id}_1,e)} & (1,1) \end{array}$$

with two commuting triangles. Thus $F': \text{RGph} \rightarrow \text{Cat}$ again fails to preserve products, though in some sense it comes closer. Simply put, while $F'(R_1 \times R_1)$ allows rewrites to be done independently in either factor, these rewrites fail to commute.

To solve this problem we shall consider RGph as a full subcategory of the category of simplicial sets, sSet . To do this, we treat a reflexive graph as a simplicial set with only degenerate simplices for $n > 1$. There is a left adjoint $\text{FC}: \text{sSet} \rightarrow \text{Cat}$, usually called **realization**, and this functor preserves products [16, Prop. B.0.15]. For example, if we treat R_1 above as a simplicial set and take the product $R_1 \times R_1$ in sSet , this contains triangles that force the triangles in $\text{FC}(R_1 \times R_1)$ to commute. Thus, realization provides a useful base change to translate from small-step operational semantics to big-step operational semantics.

The other functors in our chain of left adjoints are simpler. The “free poset” functor $\text{FP}: \text{Cat} \rightarrow \text{Pos}$ maps any category C to the poset whose elements are objects of C , with $c \leq c'$ iff C contains a morphism from c to c' . This is a valid change of base—i.e., it preserves finite products—because the product of posets is defined in the same way as the product of categories. If we apply this change of base to a model of a Cat -enriched theory, we obtain a model of a Pos -enriched theory that says for any pair of terms the presence or absence of a rewrite sequence from one to the other, without distinguishing between different sequences. We call this **full-step operational semantics**.

Finally, we can pass to the purely abstract realm where all computation is already complete. For this we take the left adjoint $\text{FS}: \text{Pos} \rightarrow \text{Set}$ to the functor $\text{UP}: \text{Set} \rightarrow \text{Pos}$ sending any set to the discrete

poset on that set. The functor FS collapses each connected component of the poset to a point; this too preserves finite products. If we apply this change of base to a model of a Pos -enriched theory, we obtain a model of a Set-enriched theory that extracts its denotational semantics by identifying all terms related by rewrites. If the rewrites are terminating and confluent, we can choose a representative term for each equivalence class: the unique term that admits no nontrivial rewrites.

6 The Category of All Models

In addition to base change, there are two other natural and useful ways to go between models of enriched theories. Suppose V is any cartesian closed category with chosen finite coproducts of the terminal object. Let $\text{VMod}(T, C)$ be the category of models of a V -theory T in a V -category C with finite V -products, as in Defn. 4. A morphism of V -theories $f: T \rightarrow T'$ induces a **change of theory** functor between the respective categories of models

$$f^*: \text{VMod}(T', C) \rightarrow \text{VMod}(T, C)$$

defined as pre-composition with f . Similarly, a V -product-preserving V -functor $g: C \rightarrow C'$ induces a **change of context** functor

$$g_*: \text{VMod}(T, C) \rightarrow \text{VMod}(T, C')$$

defined as post-composition with g .

These translations, as well as change of base, can all be packed up nicely using the *Grothendieck construction*: given any functor $F: D \rightarrow \text{Cat}$, there is a category $\int F$ that encapsulates all of the categories in the image of F , defined as follows:

$$\begin{aligned} \text{objects } & (d, x): d \in D, x \in F(d) \\ \text{morphisms } & (f: d \rightarrow d', a: F(f)(x) \rightarrow x') \\ \text{composition } & (f, a) \circ (f', a') = (f \circ f', a \circ F(f)(a')). \end{aligned}$$

Moreover there is a functor $p_F: \int F \rightarrow D$ given as follows:

$$\begin{aligned} \text{on objects } & p_F: (d, x) \mapsto d \\ \text{on morphisms } & p_F: (f, a) \mapsto f. \end{aligned}$$

For more details see [7, 15]. We noted in Section 4 that VLaw and $\text{Mod}(T, C)$ can be promoted to V -categories when V is complete and cocomplete: this and further conditions imply that we can use the enriched Grothendieck construction [5], but we focus on the ordinary Grothendieck construction for simplicity.

First, this construction lets us bring together all models of all different V -theories in all different contexts into one category. All the V -theories are objects of VLaw , as in Defn. 3. We can also create a category of all “ V -contexts”.

Definition 9. Let VCon , the **category of V -contexts** be the category for which an object is a V -category with finite V -products and a morphism is a functor that preserves finite V -products.

There is a functor

$$\text{VMod}: \text{VLaw}^{\text{op}} \times \text{VCon} \rightarrow \text{Cat}$$

that sends any object (T, C) to $\text{VMod}(T, C)$ and any morphism (f, g) to $f^*g_* = g_*f^*$. The functoriality of VMod summarizes the contravariant change-of-theory and the covariant change-of-context above.

Applying the Grothendieck construction we obtain a category $\int \mathbf{V}\mathbf{Mod}$. Technically an object of $\int \mathbf{V}\mathbf{Mod}$ is a triple (T, C, μ) , but more intuitively it is a model $\mu: T \rightarrow C$ of any V -theory T in any V -context C . Similarly, a morphism

$$(f, g, \alpha): (T, C, \mu) \rightarrow (T', C', \mu')$$

in $\mathbf{V}\mathbf{Mod}$ consists of:

- a morphism of V -theories $f: T' \rightarrow T$,
- a V -functor $g: C \rightarrow C'$ that preserves finite V -products, and
- a V -natural transformation $\alpha: g \circ \mu \circ f \Rightarrow \mu'$.

This is a natural way to map between different models of different theories in different contexts.

We can go further by creating a category that even contains all choices of enriching categories V :

Definition 10. Let \mathbf{Enr} be the category for which an object is a cartesian closed category V with chosen finite coproducts of the terminal object, and a morphism is a cartesian functor $F: V \rightarrow W$ preserving the chosen finite coproducts of the initial object.

There is a functor

$$\mathbf{Mod}: \mathbf{Enr} \rightarrow \mathbf{Cat}$$

that maps any object V to $\int \mathbf{V}\mathbf{Mod}$ and any morphism $F: V \rightarrow W$ to a functor

$$\mathbf{Mod}(F): \int \mathbf{V}\mathbf{Mod} \rightarrow \int \mathbf{W}\mathbf{Mod}$$

that has the following effect:

- $\mathbf{Mod}(F)$ maps any object (T, C, μ) to the object $(F_*(T), F_*(C), F_*(\mu))$.
- $\mathbf{Mod}(F)$ maps any morphism (f, g, α) to the morphism $(F_*(f), F_*(g), F_*(\alpha))$.

Thus, we can use the Grothendieck construction once more to pack up all choices of enrichment into one big category:

Theorem 11. There is a category $\int \mathbf{Mod}$ in which:

- An object is a choice of cartesian closed category V with chosen finite coproducts of the terminal object, a V -theory T , a V -category C with finite V -products, and a model $\mu: T \rightarrow C$.
- A morphism is a cartesian functor $F: V \rightarrow W$ preserving the chosen finite coproducts of the terminal object and a morphism $(f, g, \alpha): (F_*(T), F_*(C), F_*(\mu)) \rightarrow (T, C, \mu)$ in $\mathbf{W}\mathbf{Mod}$.

This category allows us to formally treat morphisms between objects of “different kinds”, something we often use informally, for example when speaking of a map from a set to a ring, or a group to a topological group. There are many unexplored questions about the large, heterogeneous categories which arise from the Grothendieck construction, regarding what unusual structure may be gained, such as limits and colimits with objects of different types, or identifying “processes” in which the kinds of objects change in an essential way. However, for our purposes we need only recognize that enriched Lawvere theories can be assimilated into one category, providing a single place in which to study change of base, change of theory, and change of context.

7 Applications

In computer science literature, enriched algebraic theories have primarily been studied in the context of “computational effects” [13]. Stay and Meredith have proposed that enriched Lawvere theories can be utilized for the design of programming languages [31]. To circumvent the question of *variable binding*, there is another approach which instead uses an enriched theory as a “compiler” which translates a language with binding to one without. This idea comes from the subject of combinatory logic.

7.1 The SKI-combinator calculus

The λ -calculus is an elegant formal language which is the foundation of functional computation, the model of intuitionistic logic, and the internal logic of cartesian closed categories: this is the Curry–Howard–Lambek correspondence [3].

Terms are constructed recursively by *variables*, *application*, and *abstraction*, and the basic rewrite is *beta reduction*, which substitutes the applied term for the bound variable:

$$\begin{aligned} M, N := & x \mid (MN) \mid \lambda x. M \\ (\lambda x. M N) \Rightarrow & M[N/x]. \end{aligned}$$

Despite the apparent simplicity, there are complications regarding substitution. Consider the term $M = \lambda x.(\lambda y.(xy))$: if this is applied to the variable y , then $(M y) \Rightarrow \lambda y.(y y)$ — but this is not intended, because the y in M is just a placeholder, it is “bound” by whatever will be plugged in, while the y being substituted is “free”, meaning it can refer to some other value or function in the program. Hence whenever a free variable is to be substituted for a bound variable, we need to rename the bound variable to prevent “variable capture” (e.g. $(My) \Rightarrow \lambda z.(y z)$).

This problem was noticed early in the history of mathematical foundations, even before the λ -calculus, and so Moses Schönfinkel invented **combinatory logic** [28], a basic form of logic without the red tape of variable binding, hence without functions in the usual sense. The *SKI*-calculus is the “variable-free” representation of the λ -calculus; λ -terms are translated via “abstraction elimination” into strings of combinators and applications. This is a technique for programming languages to minimize the subtleties of variables.

The insight of Stay and Meredith [30] is that even though enriched Lawvere theories have no variables, they can be used to study some programming languages through abstraction elimination. When representing such a language as a sSet-theory, vertices—i.e., 0-simplices—in the simplicial set $\text{hom}(1, t)$ serve as closed terms. More generally, vertices in $\text{hom}(t^n, t)$ serve as terms with n free variables. Rewrite rules going between such terms are edges—i.e., 1-simplices—in $\text{hom}(t^n, t)$.

To illustrate this, here is the theory of the *SKI*-calculus:

$$\text{Th}(\text{SKI})$$

type	t
term constructors	$S: 1 \rightarrow t$ $K: 1 \rightarrow t$ $I: 1 \rightarrow t$ $(--): t^2 \rightarrow t$
structural congruence	n/a
rewrites	$\sigma: (((S-) =) \equiv) \Rightarrow ((- \equiv) (= \equiv))$ $\kappa: ((K-) =) \Rightarrow -$ $\iota: (I-) \Rightarrow -$

These rewrites are implicitly universally quantified; i.e., they apply to arbitrary subterms $-$, $=$, \equiv without any variable binding involved, by using the cartesian structure of the category. They are edges with vertices as follows:

Here l, r denote the unitors and s the symmetry of the product.

These abstract rules are evaluated on concrete terms by “plugging in” via precomposition. For example:

$$\begin{array}{ccc}
 ((KS)I): & \begin{array}{c} 1 \xrightarrow{S \times I} t^2 \xrightarrow{((K-) =)} t \\ \parallel \qquad \qquad \downarrow \qquad \parallel \end{array} \\
 \kappa \circ (S \times I) \downarrow & & \\
 S: & \begin{array}{c} 1 \xrightarrow{S \times I} t^2 \xrightarrow{-} t \\ \parallel \qquad \qquad \parallel \end{array}
 \end{array}$$

A model of this theory is a sSet-functor $\mu : \text{Th}(\text{SKI}) \rightarrow \text{sSet}$ that preserves finite sSet-products. This gives a simplicial set $\mu(t)$. The images of the nullary operations S, K, I under μ are distinguished vertices of $\mu(t)$, because μ preserves the terminal object, which “points out” vertices. The image of the binary operation $(- -)$ gives for every pair of vertices $(u, v) \in \mu(t)^2$ a vertex $(u \ v)$ in $\mu(t)$ which stands for their application. In this way all possible terms built from S, K, I and application give vertices in $\mu(t)$. Similarly, rewrites going between these terms give edges in $\mu(t)$. Thus, μ gives a map of simplicial sets

$$\mu_{1,t} : \text{Th(SKI)}(1,t) \rightarrow \text{sSet}(1,\mu(t))$$

that maps the “syntactic” graph of all closed terms and rewrites to the “semantic” graph: each rewrite between terms in the theory is sent to a rewrite between the images of these terms in the model.

The fact that $\mu((--)) : \mu(t)^2 \rightarrow \mu(t)$ is not just a function but a map of simplicial sets means that pairs of edges $(a \rightarrow b, c \rightarrow d)$ in $\text{Th}(\text{SKI})(1, t)$ are sent to edges $(a b) \rightarrow (c d)$ in $\text{sSet}(1, \mu(t))$. This gives the full complexity of the theory: given a large term (program), there are many different ways it can be computed—and some take fewer steps than others:

$$\begin{array}{ccc}
((K S) (((S K) I) (I K))) & \xrightarrow{((K S) \sigma)} & ((K S) ((K (I K)) (I (I K)))) \\
\downarrow & & \downarrow ((K S) I) (I (I K))) \\
((K S) ((K K) (I (I K)))) & & \\
\downarrow & & \downarrow ((K S) ((K K) (I I))) \\
((K S) ((K K) (I K))) & & \\
\downarrow & & \downarrow ((K S) ((K K) I)) \\
((K S) ((K K) K)) & & \\
\downarrow & & \downarrow ((K S) \kappa) \\
S & \xleftarrow{\kappa} & ((K S) K)
\end{array}$$

More generally, the image $\mu(t)^n$ is a simplicial set whose vertices are *SKI*-terms with n free variables and whose edges are n -tuples of rewrites between such terms. This is because the enriched functor μ gives maps of simplicial sets

$$\mu_{t^n, t} : \text{Th}(\text{SKI})(t^n, t) \rightarrow \text{sSet}(\mu(t)^n, \mu(t)).$$

As the n -ary operations and rewrites thereof are built up from application and the three rewrites, everything works the same way as in the case $n = 0$.

This process is intuitive, but how do we actually define the model, as a functor, to pick out a specific graph? There are many models of $\text{Th}(\text{SKI})$, but in particular we care about the canonical *free* model, which means that $\mu(t)$ is simply the graph of all closed terms and rewrites in the *SKI*-calculus. This utilizes the enriched adjunction of Thm. 5:

$$\begin{array}{ccccc}
& & f_{\text{sSet}} & & \\
& \swarrow & & \searrow & \\
\underline{\text{sSet}} & & \perp & & \underline{\text{Mod}(\text{Th}(\text{SKI}), \text{sSet})} \\
& \searrow & & \swarrow & \\
& & u_{\text{sSet}} & &
\end{array}$$

Then the canonical model of closed terms and rewrites is simply the free model on the empty graph, $f_{\text{sSet}}(\emptyset)$, i.e. the V -functor $T(1, -) : T \rightarrow V$. Hence for us, the syntax and semantics of the *SKI* combinator calculus are unified in the model

$$\mu_{\text{SKI}}^{\text{sSet}} := \text{Th}(\text{SKI})(1, -) : \text{Th}(\text{SKI}) \rightarrow \text{sSet}.$$

Here we reap the benefits of the abstract construction: the graph $\mu_{\text{SKI}}^{\text{sSet}}(t)$ represents the small-step operational semantics of the *SKI*-calculus:

$$(\mu(a) \rightarrow \mu(b) \in \mu_{\text{SKI}}^{\text{sSet}}(t)) \iff (a \Rightarrow b \in \text{Th}(\text{SKI})(1, t)).$$

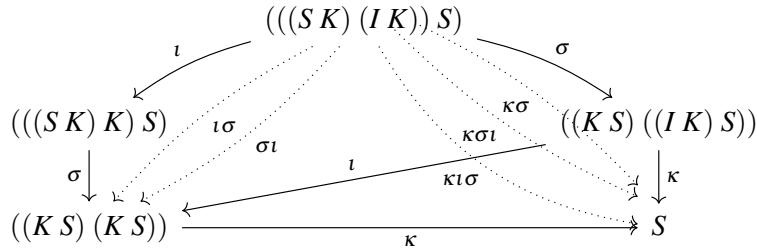
We can now consider the base changes in Sec. 5.2, to translate between several important kinds of computation for the *SKI*-calculus. Given the above description of $\text{Th}(\text{SKI})$ as enriched in sSet , we can apply the “free category” realization functor to the hom-objects, turning these reflexive graphs into categories.

Here we enjoy the fact that this functor indeed preserves products, which is essential for considering tuples of programs running in parallel: for example if we designate $G_n := \text{Th}(\text{SKI})(t^n, t)$, then the fact that $\text{FC}(G_m \times G_n) \cong \text{FC}(G_m) \times \text{FC}(G_n)$ ensures that the execution of an m -term program and an n -term program simultaneously (but independently) is the same as executing one, then the other.

Thus FC translates the theory of *SKI* from “small-step” to “big-step” operational semantics: $\text{FC}_*(\text{Th}(\text{SKI}))$ is the theory of the *SKI* calculus, but now with hom-categories whose morphisms represent finite sequences of rewrite edges in the original theory.

We can continue these base-changes to “full-step” and denotational semantics, by applying the “free poset” and “free set” (connected components) functors to the hom-objects of this theory. This process demonstrates the idea of having a “spectrum” of detail with which to analyze the semantics of a programming language, or general algebraic theory.

For example, consider the following computation:



The solid arrows are the one-step rewrites of the initial sSet -theory; applying FC_* gives the dotted composites, and FP_* asserts that all composites between any two objects are equal. Finally, FS_* collapses the whole diagram to S . This is a simple demonstration of the basic stages of computation: small-step, big-step, full-step, and denotational semantics.

7.2 Change of theory

We can equip term calculi with *reduction contexts*, which determine when rewrites are valid, thus giving the language a certain **evaluation strategy**. For example, the “weak head normal form” is given by only allowing rewrites on the left-hand side of the term.

We can do this for $\text{Th}(\text{SKI})$ by adding a reduction context marker as a unary operation, and a structural congruence rule which pushes the marker to the left-hand side of an application; lastly we modify the rewrite rules to be valid only when the marker is present:

$$\text{Th}(\text{SKI} + R)$$

sort	t
term constructors	$S, K, I: 1 \rightarrow t$
	$R: t \rightarrow t$
	$(--): t^2 \rightarrow t$
structural congruence	$R(-- =) = (R-- =)$ $RR = R$
rewrites	$\sigma_r: (((RS--) =) \equiv) \Rightarrow ((R-- \equiv) (= \equiv))$ $\kappa_r: ((RK--) =) \Rightarrow R-$ $\iota_r: (RI--) \Rightarrow R-$

The *SKI*-calculus is thereby equipped with “lazy evaluation”, an essential paradigm in modern programming. This represents a broad potential application of equipping theories with computational methods, such as evaluation strategies.

Moreover, these equipments can be added or removed as needed: using change-of-theory, we can utilize a “free reduction” sSet-functor $f_R: \text{Th}(\text{SKI}) \rightarrow \text{Th}(\text{SKI} + \mathbf{R})$:

$$\begin{array}{lll} \text{objects} & t^n & \mapsto t^n \\ \text{hom-vertices} & S, K, I & \mapsto S, K, I \\ & (--) & \mapsto R(--) \\ \text{hom-edges} & \sigma, \kappa, \iota & \mapsto \sigma_r, \kappa_r, \iota_r \end{array}$$

This essentially interprets ordinary *SKI* as having every subterm be a reduction context. This is a sSet-functor because its hom component consists of graph-homomorphisms

$$f_{n,m}: \text{Th}(\text{SKI})(t^n, t^m) \rightarrow \text{Th}(\text{SKI} + \mathbf{R})(t^n, t^m)$$

which simply send each application to its postcomposition with R , and each rewrite to its “marked” correspondent.

So, by precomposition this induces the change of theory on categories of models:

$$f_R^*: \text{Mod}(\text{Th}(\text{SKI} + \mathbf{R}), \mathbf{C}) \rightarrow \text{Mod}(\text{Th}(\text{SKI}), \mathbf{C})$$

for all semantic categories \mathbf{C} , which forgets the reduction contexts.

Similarly, there is a sSet-functor $u_R: \text{Th}(\text{SKI} + \mathbf{R}) \rightarrow \text{Th}(\text{SKI})$ which forgets reduction contexts, by sending $\sigma_r, \kappa_r, \iota_r \mapsto \sigma, \kappa, \iota$ and $R \mapsto id$; this latter is the only way that the marked reductions can be mapped coherently to the unmarked. However, this means that u_R^* does not give the desired change-of-theory of “freely adjoining contexts”, because collapsing R to the identity eliminates the significance of the marker.

This illustrates a key aspect of categorical universal algebra: because change-of-theory is given by precomposition and is thus contravariant, *properties* (equations) and *structure* (operations) can only be removed. This is a necessary limitation, at least in the present setup, but there are ways to make do. These abstract theories are not floating in isolation but are implemented in code: one can simply use a “maximal theory” with all pertinent structure, then selectively forget as needed.

8 Conclusion

We have shown how enriched Lawvere theories provide a framework for unifying the structure and behavior of formal languages. Enriching theories in category-like structures reifies operational semantics by incorporating rewrites between terms, and appropriate functors between enriching categories induce change-of-base functors between categories of enriched theories and models—this simplified condition is obtained by using only natural number arities. This idea is motivated by an example sequence of such functors, which provide a spectrum of detail in which to study the rewriting properties of a theory.

Change of base, along with change of theory and change of context, can be used to create a single category Mod , which consists of all models of all enriched Lawvere theories in all contexts. We have demonstrated these concepts with the theory of combinatory logic, $\text{Th}(\text{SKI})$, describing a change of base from small-step operational semantics to big-step to full-step to denotational semantics.

Finally, we suggest that there are many interesting change-of-base functors, by considering an endofunctor on the category of labelled transition systems, which quotients by the bisimulation relation and is indeed a change of base.

A Natural Number Arities

In this appendix we prove the lemmas required for Theorem 5 and our study of base change in Section 5. Throughout we assume V is cartesian closed with chosen n -fold coproducts n_V of its terminal object.

We begin with a study of N_V , the full subcategory of V on the objects n_V . First we must resolve a potential ambiguity. On the one hand, for any object b of V we can form the exponential b^{n_V} . On the other hand, we can take the product of n copies of b , which we call b^n . Luckily these are the same, or at least naturally isomorphic:

Lemma 12. The functors $(-)^{n_V} : V \rightarrow V$ and $(-)^n : V \rightarrow V$ are naturally isomorphic.

Proof. If $a, b \in V$, then

$$\begin{aligned} V(a, b^{n_V}) &\cong V(a \times n_V, b) && \text{hom-tensor adjunction} \\ &= V(a \times (n \cdot 1_V), b) && \text{definition of } n_V \\ &\cong V(n \cdot (a \times 1_V), b) && \text{products distribute over coproducts} \\ &\cong V(n \cdot a, b) && \text{unitality} \\ &\cong V(a, b^n) && \text{definition of coproduct} \\ &\cong V(a, b^n) && \text{definition of product.} \end{aligned}$$

Each of these isomorphisms is natural in a and b , so by the Yoneda lemma $(-)^{n_V} \cong (-)^n$. \square

We can now understand coproducts, products and exponentials in N_V :

Lemma 13. If V is any cartesian closed category with chosen coproducts of the initial object then N_V is cartesian closed, with finite coproducts. The unique initial object of N_V is 0_V . The binary coproducts in N_V are unique, given by

$$m_V + n_V = (m + n)_V.$$

The unique terminal object of N_V is 1_V , and the binary products are unique, given by

$$m_V \times n_V = (mn)_V.$$

Exponentials in N_V are also unique, given by

$$m_V^{n_V} = (m^n)_V.$$

Proof. In V we know that 0_V is an initial object and 1_V is a terminal object, by definition. Since the subcategory N_V is skeletal 0_V is the unique initial object and 1_V is the unique terminal object in N_V . Similarly, in V we have defined $(m+n)_V$ to be a coproduct of m_V and n_V , so in N_V it is the unique such, and we can unambiguously write

$$m_V + n_V = (m + n)_V.$$

Products distribute over coproducts in any cartesian closed category, so in V we have

$$m_V \times n_V \cong (1_V + \cdots + 1_V) \times (1_V + \cdots + 1_V) \cong (mn)_V$$

where in the second step we use the distributive law twice. It follows that N_V has finite products, and since this subcategory is skeletal they are unique, given by

$$m_V \times n_V = (mn)_V.$$

Finally, by Lemma 12 we have

$$m_V^{n_V} \cong m_V^n \cong \prod_{i=1}^n m_V \cong (m^n)_V.$$

It follows that N_V has exponentials, and since this subcategory is skeletal they are unique, given by

$$m_V^{n_V} = (m^n)_V. \quad \square$$

We warn the reader that $\hom(m_V, n_V)$ may not have n^m elements. It does in $s\text{Set}$, Cat , Pos and of course Set , but not in $V = \text{Set}^k$, where $|\hom(m_V, n_V)| = n^{km}$. In fact, whenever N_V has finite hom-sets it is equivalent to FinSet^k for some k . The reason is that 2_V is an internal Boolean algebra in V , so its set of elements $\hom(1_V, 2_V)$ must be some Boolean algebra B in Set . A further argument due to Garner and Trimble shows that N_V is completely characterized, up to equivalence, by this Boolean algebra, and any Boolean algebra can occur [2]. If this Boolean algebra is finite it must be isomorphic to $\{0, 1\}^k$ for some $k \geq 0$. In this case, N_V is equivalent to FinSet^k .

Now suppose C is a V -category. The question arises whether the power of an object $c \in C$ by n_V must also be the V -product of n copies of c . The answer is yes:

Lemma 14. Let C be a V -category and $c \in \text{Ob}(C)$. Then the power c^{n_V} exists if and only if the n -fold V -product c^n exists, in which case they are isomorphic.

Proof. In Section 3 we saw that an object $b \in \text{Ob}(C)$ is an n -fold V -product of copies of c precisely when it is equipped with a universal cone

$$p: 1_V \rightarrow C(b, c)^n.$$

Similarly, b is an n_V -power of c when it is equipped with a universal cone

$$q: 1_V \rightarrow C(b, c)^{n_V}.$$

The universality properties have the same form, and by Lemma 12 the functors $(-)^n: V \rightarrow V$ and $(-)^{n_V}: V \rightarrow V$ are naturally isomorphic. Thus, given either sort of universal cone we get the other, so an object is an n -fold product of copies of c if and only if it is the n_V -power of c . \square

Lemma 15. Suppose C is a V -category such that every object is the n -fold V -product c^n of some object c . Then a V -functor $F: C \rightarrow D$ preserves finite V -products if and only if it preserves powers by all objects of N_V .

Proof. Define a “finite V -power” to be a finite V -product of n copies of the same object. The V -functor F preserves finite V -powers if and only if it maps any universal cone

$$p: 1_V \rightarrow C(b, c)^n$$

in C to a universal cone in D . Similarly, F preserves powers by all objects of N_V if and only if it maps any universal cone

$$q: 1_V \rightarrow C(b, c)^{n_V}$$

in C to a universal cone in D . Two kinds of universality are involved here, but since they have the same form, and since Lemma 12 says the functors $(-)^n: V \rightarrow V$ and $(-)^{n_V}: V \rightarrow V$ are naturally isomorphic, it follows that F preserves finite V -powers if and only if it preserves powers by all objects of N_V .

It thus suffices to show that F preserves finite V -products if and only if it preserves finite V -powers. This follows from the assumption that every object is the n -fold V -product c^n of some object c . \square

Lemma 16. Let V be cartesian closed with chosen finite coproducts of the terminal object and let T be a V -category. These conditions for a V -functor $\tau: A_V \rightarrow T$ are equivalent:

1. (T, τ) is a V -theory,
2. τ preserves finite V -products,
3. τ preserves powers by objects of N_V .

Proof. Conditions 1 and 2 are equivalent by definition. Since $A_V = \underline{N}_V^{\text{op}}$, finite V -products in A_V are the same as finite V -coproducts in \underline{N}_V , which are the same as finite coproducts in N_V . Since every object in \underline{N}_V is a finite coproduct of copies of 1_V , Lemma 15 implies that conditions 2 and 3 are equivalent. \square

Lemma 17. Given a V -theory (T, τ) and a V -functor $\mu: T \rightarrow C$, the following conditions are equivalent:

- μ is a model of (T, τ) ,
- μ preserves finite V -products,
- μ preserves powers by objects of N_V .

Proof. Conditions 1 and 2 are equivalent by definition. Since τ is the identity on objects and preserves V -products each object of T is of the form t^n where $t = \tau(1_V)$. Thus, Lemma 15 implies that conditions 2 and 3 are equivalent. \square

References

- [1] J. Adámek & J. Rosický (1994): *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series 189, Cambridge University Press, Cambridge, doi:10.1017/CBO9780511600579.
- [2] J. C. Baez (2019): *Can 1+1 have more than two points?* The n -Category Café. Available at https://golem.ph.utexas.edu/category/2019/04/can_11_have_more_than_two_poin.html.
- [3] J. C. Baez & M. Stay (2011): *Physics, topology, logic and computation: a Rosetta Stone*. In B. Coecke, editor: *New Structures for Physics*, Springer, Berlin, pp. 95–172, doi:10.1007/978-3-642-12821-9. Available at <https://arxiv.org/abs/0903.0340>.
- [4] M. Barr & C. Wells (1984): *Toposes, Triples and Theories*. Grundlehren der mathematischen Wissenschaften 278, Springer, Berlin, doi:10.4204/EPTCS. Available at <https://www.math.mcgill.ca/barr/papers/ttt.pdf>.
- [5] J. Beardsley & L. Z. Wong (2019): *The enriched Grothendieck construction*. Advances in Mathematics 344, pp. 234 – 261, doi:10.1016/j.aim.2018.12.009. Available at <https://arxiv.org/abs/1804.03829>.
- [6] R. Blackwell, G. M. Kelly & A. J. Power (1989): *Two-dimensional monad theory*. Journal of Pure and Applied Algebra 59(1), pp. 1–41, doi:10.1016/0022-4049(89)90160-6.
- [7] F. Borceux (1994): *Handbook of Categorical Algebra*. Cambridge University Press, Cambridge, doi:10.1112/BLMS/28.4.440.
- [8] R. Crole (1994): *Categories for Types*. Cambridge University Press, Cambridge, doi:10.1017/CBO9781139172707.
- [9] B. Day & R. Street (1997): *Monoidal bicategories and Hopf algebroids*. Advances in Mathematics 129(1), pp. 99–157, doi:10.1006/aima.1997.1649.
- [10] E. J. Dubuc (1970): *Kan Extensions in Enriched Category Theory*. Lecture Notes in Mathematics 145, Springer, Berlin, doi:10.1007/bfb0060485.
- [11] M. Fiore, G. Plotkin & D. Turi (1999): *Abstract syntax and variable binding*. In: *Proceedings, 14th Symposium on Logic in Computer Science*, pp. 193–202.

- [12] G. Friedman (2012): *An elementary illustrated introduction to simplicial sets*. *Rocky Mountain Journal of Mathematics* 42(2), pp. 353–423, doi:10.1216/rmj-2012-42-2-353.
- [13] M. Hyland & J. Power (2006): *Discrete Lawvere theories and computational effects*. *Theoretical Computer Science* 366(1-2), pp. 144–162, doi:10.1016/j.tcs.2006.07.007. Available at <https://core.ac.uk/download/pdf/81105779.pdf>.
- [14] M. Hyland & J. Power (2007): *The category theoretic understanding of universal algebra: Lawvere theories and monads*. *Electronic Notes in Theoretical Computer Science* 172, pp. 437–458, doi:10.1016/j.entcs.2007.02.019.
- [15] B. Jacobs (1998): *Categorical Logic and Type Theory*. Elsevier, Amsterdam, doi:10.1016/s0049-237x(98)x8028-6.
- [16] A. Joyal (2008): *The theory of quasicategories and its applications*. Available at <http://citeseervx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.4968>.
- [17] P. Karazeris & G. Protsonis (2012): *Left Kan extensions preserving finite products*. *Journal of Pure and Applied Algebra* 216(8-9), pp. 2014–2028, doi:10.1016/j.jpaa.2012.02.038.
- [18] G. Kelly (2005): *The Basic Concepts of Enriched Category Theory*. Reprints in Theory and Applications of Categories 10, doi:10.1112/blms/15.1.96. Available at <http://www.tac.mta.ca/tac/reprints/articles/10/tr10abs.html>.
- [19] Y. Kinoshita, J. Power & M. Takeyama (1999): *Sketches*. *Journal of Pure and Applied Algebra* 143(1-3), pp. 275–291, doi:10.1016/s0022-4049(98)00114-5.
- [20] F. W. Lawvere (1963): *Functorial semantics of algebraic theories*. *Proceedings of the National Academy of Sciences* 50(5), pp. 869–872, doi:10.1073/pnas.50.5.869. Available at <http://www.tac.mta.ca/tac/reprints/articles/5/tr5abs.html>.
- [21] F. E. J. Linton (1966): *Some aspects of equational categories*. In: *Proceedings of the Conference on Categorical Algebra*, Springer, pp. 84–94, doi:10.1007/978-3-642-99902-4_3.
- [22] R. B. B. Lucyshyn-Wright (2015): *Enriched algebraic theories and monads for a system of arities*. *Theory and Applications of Categories* 31(5). Available at <http://www.tac.mta.ca/tac/volumes/31/5/31-05abs.html>.
- [23] C. Lüth & N. Ghani (1997): *Monads and modular term rewriting*. In: *Category Theory and Computer Science*, Springer, pp. 69–86, doi:10.1007/bfb0026982. Available at <http://www.informatik.uni-bremen.de/~cxl/papers/ctcs97l.pdf>.
- [24] B. Milewski (2017): *Category Theory for Programmers*. Available at <https://bartoszmilewski.com/2017/08/26/lawvere-theories/>.
- [25] K. Nishizawa & J. Power (2009): *Lawvere theories enriched over a general base*. *Journal of Pure and Applied Algebra* 213(3), pp. 377–386, doi:10.1016/j.jpaa.2008.07.009.
- [26] G. D. Plotkin (2004): *A structural approach to operational semantics*. *Journal of Logical and Algebraic Methods in Programming* 60-61, pp. 17–139, doi:10.1016/j.jlap.2004.05.001. Available at http://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf.
- [27] J. Power (1999): *Enriched Lawvere theories*. *Theory and Applications of Categories* 6(7), pp. 83–93. Available at <http://www.tac.mta.ca/tac/volumes/6/n7/6-07abs.html>.
- [28] M. Schönfinkel (1924): *Bausteine zu einer Logik der mathematischen Wissenschaften*. *Mathematische Annalen* 92, pp. 305–316. Available at <http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002270110>.
- [29] R. A. G. Seely (1987): *Modelling computations: a 2-categorical framework*. In: *Proceedings of the Second Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, Ithaca, New York, pp. 22–25.
- [30] M. Stay & L. G. Meredith (2017): *Representing operational semantics with enriched Lawvere theories*. Available at <https://arxiv.org/abs/1704.03080>.

- [31] M. Stay & L. G. Gregory Meredith (2016): *Logic as a distributive law*. Available at <https://arxiv.org/abs/1610.02247>.

SEMANTICS OF HIGHER-ORDER RECURSION SCHEMES

JIRÍ ADÁMEK^a, STEFAN MILIUS^b, AND JIRÍ VELEBIL^c

^{a,b} Institut für Theoretische Informatik, Technische Universität Braunschweig, Germany
e-mail address: adamek@iti.cs.tu-bs.de, mail@stefan-milius.eu

^c Faculty of Electrical Engineering, Czech Technical University of Prague, Prague, Czech Republic
e-mail address: velebil@math.feld.cvut.cz

ABSTRACT. Higher-order recursion schemes are recursive equations defining new operations from given ones called “terminals”. Every such recursion scheme is proved to have a least interpreted semantics in every Scott’s model of λ -calculus in which the terminals are interpreted as continuous operations. For the uninterpreted semantics based on infinite λ -terms we follow the idea of Fiore, Plotkin and Turi and work in the category of sets in context, which are presheaves on the category of finite sets. Fiore *et al* showed how to capture the type of variable binding in λ -calculus by an endofunctor H_λ and they explained simultaneous substitution of λ -terms by proving that the presheaf of λ -terms is an initial H_λ -monoid. Here we work with the presheaf of rational infinite λ -terms and prove that this is an initial iterative H_λ -monoid. We conclude that every guarded higher-order recursion scheme has a unique uninterpreted solution in this monoid.

1. INTRODUCTION

The present paper is a contribution to the study of the semantics of recursive definitions using category-theoretic tools and methods. Our goal is to present a category-theoretic semantics of higher-order recursion schemes in the sense of W. Damm [8]. To reach this goal we apply the theory of rational monads on a category \mathcal{K} , developed in our previous work [2] in order to formalize iteration in algebra, to the category

$$\mathcal{K} = \mathbf{Set}^{\mathcal{F}} \quad (\mathcal{F} = \text{finite sets and functions})$$

of sets in context. We use the approach to λ -calculus based on H -monoids in the category of sets in context due to M. Fiore, G. Plotkin and D. Turi [10]. Our main result is a description of the initial iterative H -monoid as the monoid of rational λ -terms, and the fact that in this monoid every higher-order recursion scheme has a unique uninterpreted solution.

We now explain the motivation of our paper in more detail. In the higher-order semantics we assume a given collection Σ of existing programs of given types (that is, a many-sorted signature of “terminals”). One recursively defines new typed programs p_1, \dots, p_n

Key words and phrases: Higher-order recursion schemes, infinite λ -terms, sets in context, rational tree.

^c Supported by the grant MSM 6840770014 of the Ministry of Education of the Czech Republic.

(forming a many-sorted signature of “nonterminals”) using symbols from Σ and $\{p_1, \dots, p_n\}$. If the recursion only concerns application, we can formalize this as a collection of equations

$$p_i = f_i \quad (i = 1, \dots, n) \quad (1.1)$$

whose right-hand sides f_i are terms in the signature of all terminals and all non-terminals. Such collections are called (first-order) *recursion schemes* and were studied in 1970’s by various authors, e.g. B. Courcelle, M. Nivat and I. Guessarian (see the monograph [12] and references there) or S. J. Garland and D. C. Luckham [11]. Recently, a categorical approach to semantics of first-order recursion schemes was presented by S. Milius and L. Moss [18]. In the present paper we take a first step in an analogous approach to the semantics of *higher-order recursion schemes* in which λ -abstraction is also used as one of the operations. That is, a higher-order recursion scheme, as introduced by W. Damm [8] (see also the recent contributions [7] and [19]) is a collection of equations $p_i = f_i$ where f_i are terms using application and λ -abstraction on symbols from Σ and $\{p_1, \dots, p_n\}$. As in [18], we first study the uninterpreted semantics, where the given system is regarded as a purely syntactic construct. At this stage the operation symbols in Σ as well as λ -abstraction and application have no interpretation on actual data. So the semantics is provided by formal (infinite) terms. These terms can be represented by rational trees, i.e., infinite trees having finitely many subtrees. Thus the uninterpreted solution assigns to each of the recursive variables p_i in (1.1) a rational tree p_i^\dagger such that the formal equations become identities if we substitute p_i^\dagger for p_i ($i = 1, \dots, n$). We assume α -conversion (renaming of bound variables) but no other rules in the uninterpreted semantics. We next turn to an interpreted semantics. Here a recursion scheme is given together with an interpretation of all symbols from Σ as well as λ -abstraction and application. Following D. Scott, we interpret the λ -calculus on a CPO, say D . The symbols of Σ are interpreted as continuous operations on D , and formal λ -abstraction and application are the actual λ -abstraction and application in the model D . An interpreted solution in D then assigns to each p_i in the context Γ of all free variables in (1.1) an element of $\mathbf{CPO}(D^\Gamma, D)$ (continuously giving to each assignment of free variables in D^Γ an element of D) such that the formal equations in the recursion scheme become identities in D when the right-hand sides are interpreted in D .

Example 1.1. The fixed-point operator Y is specified by

$$Y = \lambda f. f(Yf)$$

and the uninterpreted semantics is the rational tree

$$Y^\dagger = \begin{array}{c} \lambda f \\ | \\ @ \\ f \swarrow \searrow \\ @ \\ f \quad f \\ | \\ @ \\ f \quad \dots \end{array} \quad (1.2)$$

(The symbol @ makes application explicit.) The interpreted solution in D is the least fixed point operator (considered as an element of D).

The above example is untyped, and indeed we are only treating the untyped case in the present paper since its uninterpreted semantics is technically simpler than the typed case; however, the basic ideas of uninterpreted semantics are similar. In contrast, the interpreted semantics (based on a specified model of λ -calculus with “terminal” symbols interpreted as operations) is more subtle in the untyped case.

Our main result is that every guarded higher-order recursion scheme has a unique uninterpreted solution, and a least interpreted one. This demonstrates that the methods for iteration in locally finitely presentable categories developed in [2] can serve not only for first-order iteration, when applied to endofunctors of \mathbf{Set} , but also for higher-order iteration: it is possible to apply these methods to other categories, here the category of sets in context.

Related Work. This is an extended and revised version of the conference paper [4]. In addition to the material in that extended abstract we include here the theory of iterative monoids in a monoidal category, see Section 4 below, and we provide detailed proofs.

2. PRESHEAVES AS ALGEBRAS

Notation 2.1.

- (1) Throughout the paper a given countably infinite set Var of variables is assumed. Finite subsets $\Gamma \subseteq \mathsf{Var}$ are called *contexts* and form a full subcategory \mathcal{F} of \mathbf{Set} . We also assume that a (possibly empty) finitary signature Σ is given.

When speaking about formulas in context Γ we mean those that have all free variables in Γ . For example, $\lambda x.yx$ is a formula in context $\Gamma = \{y, y'\}$.

- (2) The category $\mathbf{Set}^{\mathcal{F}}$ of “covariant presheaves” on \mathcal{F} is well known to be equivalent to the category of finitary endofunctors of \mathbf{Set} . Indeed, every endofunctor X yields the presheaf $X \upharpoonright \mathcal{F}$, and conversely, every presheaf X in $\mathbf{Set}^{\mathcal{F}}$ has a left Kan extension to a finitary endofunctor of \mathbf{Set} : for every set M we have

$$X(M) = \bigcup X i_{\Gamma}[X(\Gamma)]$$

where the union ranges over embeddings $i_{\Gamma}: \Gamma \hookrightarrow M$ of contexts Γ into M , and $X i_{\Gamma}[X(\Gamma)]$ denotes the image of $X i_{\Gamma}$.

- (3) From now on we speak about presheaves when objects of $\mathbf{Set}^{\mathcal{F}}$ are meant. The word *endofunctor* is reserved for endofunctors on $\mathbf{Set}^{\mathcal{F}}$ throughout our paper.

Example 2.2.

- (i) The *presheaf of variables*, V , is our name for the embedding $\mathcal{F} \hookrightarrow \mathbf{Set}$: $V(\Gamma) = \Gamma$. As we will see in Section 3, V is the unit of the monoidal operation of substitution.
- (ii) *Free presheaf* on one generator of context Γ is our name for the representable presheaf

$$\mathcal{F}(\Gamma, -).$$

Indeed, the Yoneda lemma states that this presheaf is freely generated by the element id_{Γ} of context Γ : for every presheaf X and every $x \in X(\Gamma)$ there exists a unique morphism $f: \mathcal{F}(\Gamma, -) \rightarrow X$ with $f_{\Gamma}(\text{id}_{\Gamma}) = x$. Observe that $\mathcal{F}(\Gamma, -)$ is naturally

isomorphic to the functor $X \mapsto X^n$, where $n = \text{card } \Gamma$ is the power of Γ . Consequently a free presheaf on k generators in contexts $\Gamma_1, \dots, \Gamma_k$ has the form

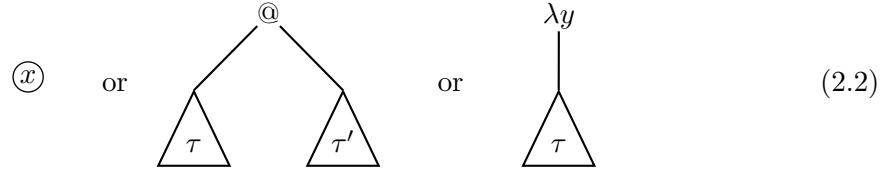
$$\Gamma \mapsto \Gamma^{n_1} + \dots + \Gamma^{n_k}, \quad \text{where } n_i = \text{card } \Gamma_i.$$

This is the “polynomial presheaf” X_Σ of a signature Σ of k operation symbols of the given arities n_i .

- (iii) The *presheaf F_λ of (finite) λ -terms* is defined via a quotient since we want to treat λ -terms always modulo α -conversion. We first consider the set of all λ -trees τ given by the grammar

$$\tau ::= x \mid \tau @ \tau \mid \lambda y. \tau \quad (x, y \in \text{Var}). \quad (2.1)$$

In the graphic form:



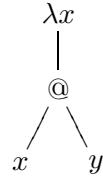
The notions of a free and bound variable of a λ -tree τ are defined as usual.

As explained in [10], the following approach is equivalent to defining λ -terms up to α -equivalence by de Bruijn levels: We first denote by $F'_\lambda(\Gamma)$ the set of all finite λ -trees with free variables in the context $\Gamma = \{x_1, \dots, x_n\}$. We then define the presheaf F_λ in context Γ by

$$F_\lambda(\Gamma) = F'_\lambda(\Gamma)/\sim_\alpha$$

where \sim_α represents the α -conversion: this is the least congruence with $\lambda y. \tau \sim_\alpha \lambda z. \tau[z/y]$, where z is not a free variable of τ . And we define F_λ on morphisms $\gamma : \Gamma \rightarrow \Gamma'$ by choosing a term $t \in F_\lambda(\Gamma)$, relabelling all bound variables so that they do not lie in Γ' , and denoting by $F_\lambda \gamma(t)$ the term obtained by relabelling every free variable $x \in \Gamma$ to $\gamma(x) \in \Gamma'$.

We call the congruence classes of finite λ -trees modulo α -conversion *finite λ -terms*. (Finite λ -trees do not form a presheaf, due to possible clashes of bound and free variables. For example consider the λ -tree



in $F'_\lambda\{y\}$ and the function $j : \{y\} \rightarrow \Gamma$ with $x \in \Gamma$ and $j(y) = x$. Then to define the action of F'_λ on j we must rename the bound variable x to some $z \notin \Gamma$. But in fact, any other renaming to $z' \notin \Gamma$ is fine, too. So trying to define the action of F'_λ on functions naturally forces us to consider equivalence classes modulo α -conversion.)

- (iv) The *presheaf $F_{\lambda, \Sigma}$ of finite λ - Σ -terms* is defined analogously: in (2.1) we just add the term $\sigma(\tau_1, \dots, \tau_n)$ for every n -ary operation symbol $\sigma \in \Sigma$, and in (2.2) the corresponding tree.

- (v) The presheaf T_λ of all (finite and infinite) λ -terms is defined analogously to F_λ . We first denote by $T'_\lambda(\Gamma)$ the set of all trees (2.2) dropping the assumption of finiteness. Then we use α -conversion: for infinite trees t and t' we write

$$t \sim_\alpha t'$$

if their (finite) cuttings at level k (with label \perp for all leaves at level k) are α -equivalent in the above sense for all $k \in \mathbb{N}$. (We can formalize this by using $\Sigma_\perp = \Sigma \cup \{\perp\}$ with \perp a constant symbol outside of $\Sigma \cup \text{Var}$). The presheaf T_λ is defined on objects Γ by $T_\lambda(\Gamma) = T'_\lambda(\Gamma)/\sim_\alpha$ and on morphisms $\gamma : \Gamma \rightarrow \Gamma'$ by relabellings of variables as in (iii). Observe that since $\text{Var} \setminus \Gamma$ is infinite, the relabelling of bound variables needed here causes no problem.

- (vi) The presheaf R_λ of *rational* λ -terms is also defined analogously. Recall that a tree is called *rational* if it has up to isomorphism only finitely many subtrees. We denote by $R'_\lambda(\Gamma)$ the set of all rational trees in $T'_\lambda(\Gamma)$ and define a presheaf R_λ by $R_\lambda(\Gamma) = R'_\lambda(\Gamma)/\sim_\alpha$ on objects, and by relabellings of variables (as in (iii)) on morphisms. Observe that, by definition, every rational λ -term t is represented by a rational λ -tree. However, t can also be represented by non-rational λ -trees—for example, if it contains infinitely many λ 's, the α -conversion can introduce an infinite number of bound variables.
- (vii) The presheaves $T_{\lambda,\Sigma}$ (of all λ - Σ -terms) and $R_{\lambda,\Sigma}$ (of rational λ - Σ -terms) are obvious modifications of (iv) and (v): one adds to (2.1) and (2.2) the case $\sigma(\tau_1, \dots, \tau_n)$ for all n -ary symbols $\sigma \in \Sigma$ and all (rational) λ - Σ -trees τ_1, \dots, τ_n .

Notation 2.3. We denote by $\delta : \mathbf{Set}^{\mathcal{F}} \rightarrow \mathbf{Set}^{\mathcal{F}}$ the endofunctor defined by

$$\delta X(\Gamma) = X(\Gamma + 1).$$

Observe that δ preserves limits and colimits.

Note that an algebra for δ is a presheaf Y together with an operation $Y(\Gamma + 1) \rightarrow Y(\Gamma)$ for all contexts Γ —this is precisely the form of λ -abstraction, where to a formula f in $Y(\Gamma + \{y\})$ we assign $\lambda y.f$ in $Y(\Gamma)$. The other λ -operation, application, is simply a presheaf morphism $X \times X \rightarrow X$, that is, a binary operation on X . We put these two together:

Notation 2.4. Let H_λ denote the endofunctor of $\mathbf{Set}^{\mathcal{F}}$ given by

$$H_\lambda X = X \times X + \delta X.$$

Thus, an algebra for H_λ is a presheaf X together with operations of application $X(\Gamma) \times X(\Gamma) \rightarrow X(\Gamma)$ and abstraction $X(\Gamma + 1) \rightarrow X(\Gamma)$ for all contexts Γ ; these operations are compatible with the renaming of free variables.

Example 2.5. The presheaves F_λ , T_λ and R_λ are algebras for H_λ in the obvious sense.

Remark 2.6.

- (i) The slice category $V/\mathbf{Set}^{\mathcal{F}}$ of presheaves X together with a morphism $i : V \rightarrow X$ is called the category of *pointed presheaves*. For example F_λ is a pointed presheaf in a canonical sense: $i^F : V \rightarrow F_\lambda$ takes a variable x to the term x . Analogously $i^T : V \rightarrow T_\lambda$ and $i^R : V \rightarrow R_\lambda$ are pointed presheaves, and so are $F_{\lambda,\Sigma}$, $R_{\lambda,\Sigma}$ and $T_{\lambda,\Sigma}$.
- (ii) Recall that the category $\mathbf{Alg} H_\lambda$ of algebras for H_λ has as morphisms the usual H_λ -homomorphisms, i.e., a morphism from $a : H_\lambda X \rightarrow X$ to $b : H_\lambda Y \rightarrow Y$ is a natural

transformation $f: X \rightarrow Y$ such that $f \cdot a = b \cdot H_\lambda f$. Then $\mathbf{Alg} H_\lambda$ is a concrete category over $\mathbf{Set}^{\mathcal{F}}$ with the forgetful functor $(H_\lambda X \rightarrow X) \mapsto X$.

Theorem 2.7 (see [10]). *The presheaf F_λ of finite λ -terms is the free H_λ -algebra on V .*

Definition 2.8 (see [2]). Given an endofunctor H , an algebra $a: HA \rightarrow A$ is called

- (1) *completely iterative* (cia for short) if for every object X (of variables) and every (flat equation) morphism $e: X \rightarrow HX + A$ there exists a unique *solution* which means a unique morphism $e^\dagger: X \rightarrow A$ such that the square below commutes

$$\begin{array}{ccc} X & \xrightarrow{e^\dagger} & A \\ e \downarrow & & \uparrow [a, \text{id}] \\ HX + A & \xrightarrow{He^\dagger + \text{id}} & HA + A \end{array} \quad (2.3)$$

- (2) *iterative* if every equation morphism $e: X \rightarrow HX + A$ with X finitely presentable has a unique solution $e^\dagger: X \rightarrow A$.

We are going to characterize finitely presentable presheaves in Theorem 2.16. In practice, we are interested only in equations using free presheaves (on polynomial endofunctors of \mathbf{Set}) as X , but including the more general concept does not “disturb” anything as we explain in Remark 5.2.

Example 2.9. As proved in [17], Corollary 6.3, the free completely iterative algebra for an arbitrary finitary endofunctor H on an object X is precisely the terminal coalgebra for $H(-) + X$. More detailed, suppose TX is the terminal coalgebra for $H(-) + X$, then its structure morphism is an isomorphism by Lambek’s Lemma and the inverse of this morphism has the components

$$\tau_X: HTX \rightarrow TX \quad \text{and} \quad \eta_X^T: X \rightarrow TX$$

making TX a free cia on X .

Conversely, let $\tau_X: HTX \rightarrow TX$ be a cia which is free on X w.r.t. the universal arrow η_X^T . Then $[\tau_X, \eta_X^T]: HTX + X \rightarrow TX$ is an isomorphism, and its inverse is the structure of the terminal coalgebra for $H(-) + X$.

Theorem 2.10. *The presheaf T_λ of infinite λ -terms is the free completely iterative H_λ -algebra on V .*

Proof. As explained in Example 2.9 above, the free completely iterative algebra for H_λ on V is precisely the terminal coalgebra for $H_\lambda(-) + V$. The latter functor clearly preserves limits of ω^{op} -chains. Consequently, its terminal coalgebra is a limit of the chain W with $W_0 = 1$ (the terminal presheaf) and $W_{n+1} = H_\lambda W_n + V$, where the connecting maps are the unique $w_0: W_1 \rightarrow W_0$ and $w_{n+1} = H_\lambda w_n + \text{id}_V$.

Observe first that the limit of W is computed objectwise. So for every context Γ we can identify $W_0(\Gamma)$ with the set $\{\perp\}$ where $\perp \notin \mathbf{Var}$, and we have

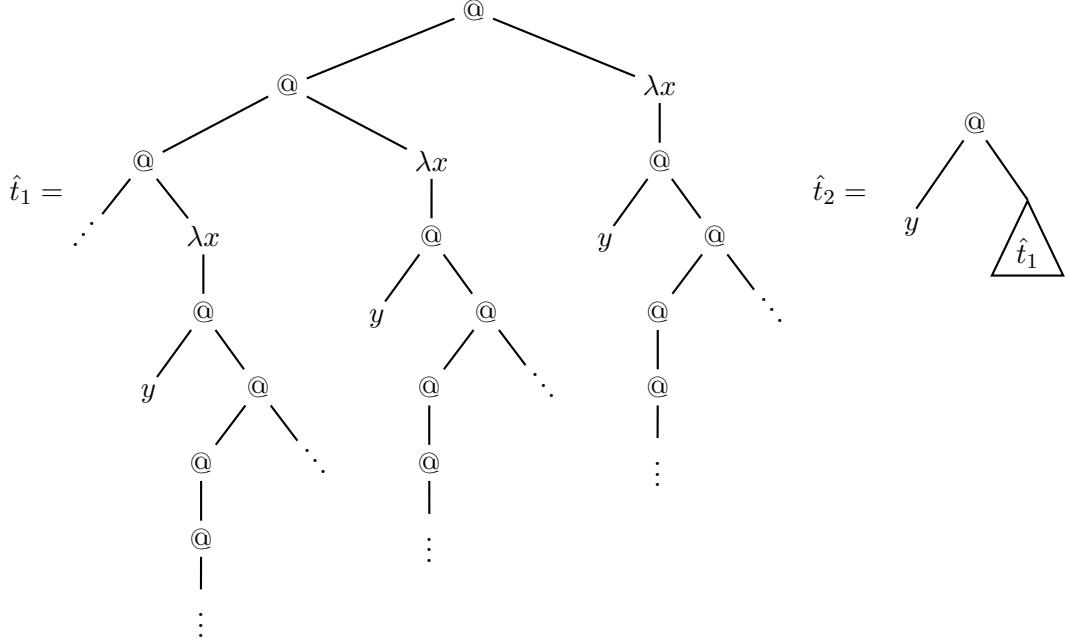
$$W_{n+1}(\Gamma) = W_n(\Gamma) \times W_n(\Gamma) + W_n(\Gamma + 1) + \Gamma.$$

An easy induction proof now shows that $W_n(\Gamma)$ can be identified with the set of all λ -terms in context Γ of depth at most n having all leaves of depth n labelled by \perp . And $w_{n+1}: W_{n+1} \rightarrow W_n$ cuts away the level $n+1$ in the trees of $W_{n+1}(\Gamma)$, relabelling level- n leaves by \perp . With this identification we obtain T_λ as a limit of W_n where the limit maps $T_\lambda \rightarrow W_n$ cut the trees in $T_\lambda(\Gamma)$ at level n and relabel level- n leaves by \perp . \square

Example 2.11. The complete iterativity of the algebra T_λ means that we are able to solve systems of recursive equations such as

$$\begin{array}{lcl} p_1 & = & p_1 @ (\lambda x.p_2) \\ p_2 & = & y @ p_1. \end{array} \quad (2.4)$$

Indeed, the solution in $T_\lambda(\{y\})$ is formed by the λ -terms represented by the following trees \hat{t}_1 and \hat{t}_2 :



How is this related to the above concept of Definition 2.8? Firstly, every system of recursive equations can be flattened: a flat system has in context Γ the right-hand sides of only three types: $p_i @ p_j$ or $\lambda x.p_i$ or a term in $T_\lambda(\Gamma)$. For example, we flatten the system (2.4) to

$$\begin{aligned} p_1 &= p_1 @ p_3 \\ p_2 &= p_4 @ p_1 \\ p_3 &= \lambda x.p_2 \\ p_4 &= y \end{aligned} \tag{2.5}$$

Let $\Gamma = \{y\}$ be the context of all free variables and let X be the free presheaf on generators p_1, \dots, p_4 of context Γ , see Example 2.2. Notice that even though the recursion variables p_1, \dots, p_4 appear as constants in the system (2.5), the associated presheaf X is not a constant presheaf. Using the Yoneda lemma, the above system (2.5) defines an obvious morphism

$$e: X \rightarrow H_\lambda X + T_\lambda$$

viz, the unique one such that $e_T(p_i)$ is the right-hand side of the equation above. The solution

$$e^\dagger: X \rightarrow T_\lambda$$

is the unique morphism such that e_Γ^\dagger takes p_i to the solution in T_λ ; for example $e_\Gamma^\dagger(p_1) = [\hat{t}_1]$ for the above tree \hat{t}_1 . We will see in Theorem 5.7 below that equations such as (2.5) have a unique solution yielding rational trees.

Remark 2.12. Given an equation morphism

$$e: X \rightarrow H_\lambda X + T_\lambda$$

then the solution $e^\dagger: X \rightarrow T_\lambda$ allows us to choose, for every element p of $X(\Gamma)$, a tree \hat{t}_p in $T_\lambda(\Gamma)$ with

$$e_\Gamma^\dagger(p) = [\hat{t}_p].$$

Due to the commutativity of (2.3) for $H_\lambda X = X \times X + \delta X$ we have three possible cases for every p :

- (a) $e_\Gamma(p) = (p_1, p_2)$ in $X(\Gamma) \times X(\Gamma)$, then for the operation $\tau: H_\lambda T_\lambda \rightarrow T_\lambda$ we have

$$[\hat{t}_p] = \tau([\hat{t}_{p_1}], [\hat{t}_{p_2}])$$

in other words,

$$\begin{array}{ccc} & @ \nearrow & \searrow \\ \hat{t}_p \sim_\alpha & & \\ & \hat{t}_{p_1} & \hat{t}_{p_2} \end{array}$$

- (b) $e_\Gamma(p) = q$ in $X(\Gamma + \{x\})$, then

$$[\hat{t}_p] = \tau([\hat{t}_q])$$

in other words

$$\begin{array}{c} \lambda x \\ \hat{t}_q \sim_\alpha \downarrow \\ \hat{t}_p \end{array}$$

or

- (c) $e_\Gamma(p)$ lies in $T_\lambda(\Gamma)$ and is represented by \hat{t}_p :

$$e_\Gamma^\dagger(p) = [\hat{t}_p] = e_\Gamma(p).$$

Remark 2.13. We are going to characterize the presheaf R_λ as a free iterative algebra for H_λ . That is, in equations we admit only presheaves X of variables that are *finitely presentable*. Recall that an object X of a category \mathcal{W} is *finitely presentable* provided that its hom-functor $\mathcal{W}(X, -)$ preserves filtered colimits. We are first going to characterize the finitely presentable presheaves by using the following concept:

Definition 2.14 (see [4]). A presheaf X is called *super-finitary* provided that each $X(\Gamma)$ is finite and there exists a nonempty context Γ_0 *generating* X in the sense that for every nonempty context Γ we have

$$X(\Gamma) = \bigcup_{\gamma: \Gamma_0 \rightarrow \Gamma} X\gamma[X(\Gamma_0)]. \quad (2.6)$$

Example 2.15. A signature Σ defines the polynomial presheaf X_Σ , see Example 2.2(ii), by $X_\Sigma(\Gamma) = \coprod_{\sigma \in \Sigma} \Gamma^{\text{ar}(\sigma)}$. This is a super-finitary presheaf iff Σ is a finite signature. Other super-finitary presheaves are precisely the quotients of X_Σ with Σ finite.

Theorem 2.16. A presheaf in $\mathbf{Set}^{\mathcal{F}}$ is finitely presentable iff it is super-finitary.

Proof. (1) Let X be a super-finitary presheaf and let Γ_0 be a context of n variables generating X . We prove that X is a finite colimit of representables. Since representables are (due to Yoneda lemma) clearly finitely presentable, this proves finite presentability of X .

Form the finite diagram of all presheaves

$$Z_a = \mathcal{F}(\Gamma, -)$$

where $\Gamma \subseteq \Gamma_0 + \Gamma_0$ is a context of at most $2n$ variables¹ and $a \in X(\Gamma)$. The connecting morphisms are the Yoneda transformations

$$Yf: Z_a \rightarrow Z_{a'} \quad \text{for } a \in X(\Gamma) \text{ and } a' \in X(\Gamma')$$

where $f: \Gamma' \rightarrow \Gamma$ is a function that fulfils $Xf(a') = a$. The Yoneda transformations

$$z_a: Z_a \rightarrow X, \quad \text{with the components defined by } f \mapsto Xf(a),$$

clearly form a compatible cocone of this finite diagram. We prove that this is a colimit cocone. In other words, for every context $\bar{\Gamma}$ we must prove that the cocone of all $\bar{\Gamma}$ -components $z_a^{\bar{\Gamma}}$ (sending elements $f: \Gamma \rightarrow \bar{\Gamma}$ of $Z_a = \mathcal{F}(\Gamma, -)$ to $Xf(a)$) is a colimit in **Set**. For that we only need to verify that in every context $\bar{\Gamma}$

- (i) the cocone $z_a^{\bar{\Gamma}}$ is collectively epimorphic, and
- (ii) whenever two elements $f: \Gamma \rightarrow \bar{\Gamma}$ of Z_a and $f': \Gamma' \rightarrow \bar{\Gamma}$ of $Z_{a'}$ fulfil $z_a^{\bar{\Gamma}}(f) = z_{a'}^{\bar{\Gamma}}(f')$, then there exists a zig-zag connecting f and f' in the $\bar{\Gamma}$ -component of our diagram.

The proof of (i) is trivial: given an element $a \in X(\bar{\Gamma})$, either $\bar{\Gamma} = \emptyset$ or by Equation (2.6) there exists $f: \Gamma_0 \rightarrow \bar{\Gamma}$ and an element $b \in X(\Gamma_0)$ with $a = Xf(b)$, in other words,

$$a = z_b^{\bar{\Gamma}}(f).$$

In case $\bar{\Gamma} = \emptyset$ we have $a = z_a^{\bar{\Gamma}}(\text{id}_\emptyset)$.

To prove (ii), observe that the given equation states

$$Xf(a) = Xf'(a').$$

In case $\bar{\Gamma}$ has at most $2n$ variables, we can assume $\bar{\Gamma} \subseteq \Gamma_0 + \Gamma_0$ and the desired zig-zag is

$$Z_a \xleftarrow{Yf} Z_b \xrightarrow{Yf'} Z_{a'},$$

where $b = Xf(a)$. Thus, we can assume that $\bar{\Gamma}$ has more than $2n$ elements.

Case 1: $\Gamma = \emptyset = \Gamma'$. Here $f = f'$ and we have $Xf(a) = Xf(a')$. Choose a monomorphism $m: \Gamma_0 \rightarrow \bar{\Gamma}$ and observe that $f = m \cdot g$ for the unique $g: \emptyset \rightarrow \Gamma_0$. Thus $Xm(Xg(a)) = Xm(Xg(a'))$ and since m is a split monomorphism, we conclude $Xg(a) = Xg(a') = c$.

The desired zig-zag is

$$Z_a \xleftarrow{Yg} Z_c \xrightarrow{Yg'} Z_{a'}.$$

Case 2: $\Gamma = \emptyset \neq \Gamma'$. Factorize f' as an epimorphism e followed by a split monomorphism m :

$$\begin{array}{ccccc} \Gamma' & \xrightarrow{\quad} & \bar{\Gamma} & \xleftarrow{\quad} & \Gamma_1 \\ & \searrow e & & \nearrow m & \\ & & \Gamma_1 & & \end{array}$$

Then, since for the unique $h: \emptyset \rightarrow \Gamma_1$ we have $f = m \cdot h$, we obtain

$$Xm(Xe(a')) = Xm(Xh(a)).$$

¹The reason why we need $2n$ variables will become clear in (2.7) below.

Thus, $Xe(a') = Xh(a) = c$ which yields the zig-zag

$$Z_a \xleftarrow{Yh} Z_c \xrightarrow{Ye} Z_{a'}.$$

Case 3: $\Gamma \neq \emptyset \neq \Gamma'$. Find $g: \Gamma_0 \rightarrow \Gamma$ with $a = Xg(b)$ and $g': \Gamma_0 \rightarrow \Gamma'$ with $a' = Xg'(b')$ for some $b, b' \in X(\Gamma_0)$. Then $X(f \cdot g)(b) = X(f' \cdot g')(b')$. Now factorize $[f \cdot g, f' \cdot g']: \Gamma_0 + \Gamma_0 \rightarrow \bar{\Gamma}$ as an epimorphism followed by a split monomorphism; so we obtain a commutative diagram

$$\begin{array}{ccc} \Gamma_0 + \Gamma_0 & \xrightarrow{[f \cdot g, f' \cdot g']} & \bar{\Gamma} \\ & \searrow [e, e'] & \swarrow m \\ & \Gamma_1 & \end{array} \quad (2.7)$$

Since m is a split monomorphism, conclude that $Xe(b) = Xe'(b') = c$. The desired zig-zag is

$$\begin{array}{ccc} Z_a & & Z_{a'} \\ \downarrow Yg & & \downarrow Yg' \\ Z_b & \xleftarrow{Ye} & Z_{b'} \\ & \nearrow Ye' & \end{array}$$

(2) Let X be a finitely presentable object of $\mathbf{Set}^{\mathcal{F}}$. The empty maps are denoted by $t_{\Gamma}: \emptyset \rightarrow \Gamma$. For every nonempty context Γ_0 let X_{Γ_0} be the subfunctor of X generated by the elements of $X(\Gamma_0) \cup X(\emptyset)$: it assigns to every Γ the subset of $X(\Gamma)$ given by

$$X_{\Gamma_0}(\Gamma) = X t_{\Gamma}[X(\emptyset)] \cup \bigcup_{f: \Gamma_0 \rightarrow \Gamma} X f[X(\Gamma_0)].$$

We obviously have a union

$$X = \bigcup_{\Gamma_0 \in \mathcal{F} \setminus \{\emptyset\}} X_{\Gamma_0}$$

which is directed: given nonempty contexts Γ_0, Γ_1 then $X_{\Gamma_0} \cup X_{\Gamma_1} \subseteq X_{\Gamma_0 \cup \Gamma_1}$. Since X is finitely presentable, the morphism

$$\text{id}_X: X \rightarrow \underset{\Gamma_0 \in \mathcal{F} \setminus \{\emptyset\}}{\text{colim}} X_{\Gamma_0}$$

factorizes through one of the colimit injections $X_{\Gamma_0} \hookrightarrow X$. In other words

$$X = X_{\Gamma_0} \quad \text{for some } \Gamma_0 \neq \emptyset.$$

It remains to prove that the sets $X(\Gamma_0)$ and $X(\emptyset)$ are finite, then every $X(\Gamma)$ is finite.

For every finite set $M \subseteq X(\emptyset)$ we have the subfunctor X^M of X equal to X on nonempty objects and maps, and assigning M to \emptyset . We obviously get X as a directed union of these subfunctors X^M , thus, as above, there exists M with $X = X^M$. Then $X(\emptyset) = M$ is finite.

For every finite set $M \subseteq X(\Gamma_0)$ we have the subfunctor ${}^M X$ of $X = X_{\Gamma_0}$ generated by the elements of $M \cup X(\emptyset)$:

$${}^M X(\Gamma) = X t_{\Gamma}[X(\emptyset)] \cup \bigcup_{f: \Gamma_0 \rightarrow \Gamma} X f[M].$$

Again X is a directed union of these subfunctors ${}^M X$, thus, there exists M with $X = {}^M X$, proving that $X(\Gamma_0)$ is finite. \square

Theorem 2.17. *The presheaf R_λ of rational λ -terms is the free iterative H_λ -algebra on V .*

Proof. (I) R_λ is an iterative algebra for H_λ . Indeed, given an equation morphism

$$e: X \rightarrow H_\lambda X + R_\lambda$$

where Equation (2.6) holds for Γ_0 , we know that its extension

$$\bar{e}: X \xrightarrow{e} H_\lambda X + R_\lambda \hookrightarrow H_\lambda X + T_\lambda$$

has a unique solution $e^\dagger: X \rightarrow T_\lambda$, and we are going to prove that the trees $e_{\Gamma_0}^\dagger(p)$ and $e_\emptyset^\dagger(p)$ are all rational. It then follows that all the trees $e_\Gamma^\dagger(p)$ are rational for all contexts Γ , and this gives us the desired solution $X \rightarrow R_\lambda$. Indeed, for each $x \in X(\Gamma)$ with $\Gamma \neq \emptyset$ we have $x = Xf(p)$ for some $f: \Gamma_0 \rightarrow \Gamma$ and $p \in X(\Gamma_0)$. Then $e_\Gamma^\dagger(x) = e_\Gamma^\dagger(Xf(p)) = T_\lambda f(e_{\Gamma_0}^\dagger(p))$ by the naturality of e^\dagger , and since $e_{\Gamma_0}^\dagger(p)$ is rational, so is $T_\lambda f(e_{\Gamma_0}^\dagger(p))$. (The action of $T_\lambda f$ is just relabelling leaves according to f .)

Now every element of $X(\Gamma_0) = \{p_1, \dots, p_n\}$ yields an element

$$e_{\Gamma_0}(p_i) \in X(\Gamma_0) \times X(\Gamma_0) + X(\Gamma_0 + \{x\}) + R_\lambda(\Gamma_0)$$

which is either (i) a pair (p_j, p_k) or (ii) $q \in X(\Gamma_0 + \{x\})$ or (iii) a rational tree in $R_\lambda(\Gamma_0)$. Put $t_i = e_{\Gamma_0}^\dagger(p_i)$, then in the last case the commutativity of Diagram (2.3) implies that $e_{\Gamma_0}(p_i) = t_i$ (cf. Remark 2.12). From (2.3) we also obtain in cases (i) and (ii)

$$t_i = t_j @ t_k \quad \text{and} \quad t_i = \lambda x. e_{\Gamma_0 + \{x\}}^\dagger(q), \quad \text{respectively.}$$

From Equation (2.6) we see that in case (ii) there exists $f: \Gamma_0 \rightarrow \Gamma_0 + \{x\}$ with $q = Xf(p_j)$ for some j , then $e_{\Gamma_0 + \{x\}}^\dagger(q) = T_\lambda f(e_{\Gamma_0}^\dagger(p_j)) = T_\lambda f(t_j)$. Thus we get equations telling us that for every i either $t_i = t_j @ t_k$ or $t_i = \lambda x. T_\lambda f(t_j)$ or t_i is a rational tree. Using these equations it is now easy, for every $i = 1, \dots, n$, to prove by induction on the depth k of subtrees of t_i that each subtree of t_i is either of the form $s = T_\lambda f(e_{\Gamma_0}^\dagger(r))$ for some $r \in X(\Gamma_0)$ and some $f: \Gamma_0 \rightarrow \Gamma_0 + \{x\}$, or s is a subtree of some rational tree $e_{\Gamma_0}^\dagger(r) = e_{\Gamma_0}(r)$ in case (iii). Since $X(\Gamma_0)$ is a finite set, it follows that every tree t_i has only finitely many subtrees, whence $t_i \in R_\lambda(\Gamma_0)$.

The case $X(\emptyset) = \{p_1, \dots, p_n\}$ is analogous: for $t_i = e_\emptyset^\dagger(p_i)$ we get (i) $t_i = t_j @ t_k$ or (ii) $t_i = \lambda x. e_{\{x\}}^\dagger(q)$ or (iii) $t_i = e_\emptyset(p_i) \in R_\lambda(\emptyset)$. We already know that the trees in case (ii) are rational. Thus, each subtree of $e_\emptyset^\dagger(p_i)$ is either $e_\emptyset^\dagger(r)$ or it is a subtree of some rational tree in cases (ii) or (iii).

The solution of e in R_λ is unique because every solution in R_λ yields a solution of the extended morphism \bar{e} in T_λ .

(II) Let \mathcal{D} be the category of all equation morphisms

$$e: X \rightarrow H_\lambda X + V, \quad X \text{ finitely presentable},$$

whose morphisms are the coalgebra homomorphisms for $H_\lambda(-) + V$. The diagram $D: \mathcal{D} \rightarrow \mathbf{Set}^{\mathcal{F}}$, $D(e) = X$, is filtered and its colimit is the free iterative H_λ -algebra on V , see [2]. We will prove that R_λ is a colimit of D . Recall that R_λ is a pointed presheaf (see Remark 2.6).

For every e as above the equation morphism

$$\tilde{e} \equiv X \xrightarrow{e} H_\lambda X + V \xrightarrow{\text{id} + i^R} H_\lambda X + R_\lambda$$

has a unique solution $\tilde{e}^\dagger: X \rightarrow R_\lambda$. It is easy to verify that these morphisms form a cocone for the diagram D . Since D is a filtered diagram in $\mathbf{Set}^{\mathcal{F}}$ and since colimits in $\mathbf{Set}^{\mathcal{F}}$ are constructed objectwise in \mathbf{Set} , in order to prove that

$$R_\lambda = \text{colim } D \quad \text{with the colimit cocone } (\tilde{e}^\dagger)$$

all we need to prove is that for every context Γ

- (a) the cocone \tilde{e}_Γ^\dagger is collectively epimorphic: $R_\lambda(\Gamma) = \bigcup \tilde{e}_\Gamma^\dagger[X]$, and
- (b) whenever \tilde{e}_Γ^\dagger merges $x, x' \in X(\Gamma)$, there exists a connecting morphism in \mathcal{D} merging x and x' too.

To prove (a), let $t \in R_\lambda(\Gamma)$ be a rational tree and let Γ_0 be the context of variables x_s indexed by the finitely many subtrees s of t (up to isomorphism). Let X be the free presheaf on the set Γ_0 of generators of context $\bar{\Gamma} = \Gamma \cup \Gamma_0$, see Example 2.2(ii). Define

$$e: X \rightarrow H_\lambda X + V$$

by assigning to every variable x_s , for a subtree s of t , the following value: if $s = s' @ s''$ in t , then

$$e_\Gamma(x_s) = x_{s'} @ x_{s''} \quad \text{in } X(\bar{\Gamma}) \times X(\bar{\Gamma}),$$

if $s = \lambda y.s'$ in t , then

$$e_\Gamma(x_s) = \lambda y.x_{s'} \quad \text{in } X(\bar{\Gamma} + \{y\}),$$

and if s is a leaf labelled by $x \in \Gamma$, then

$$e_\Gamma(x_s) = x \quad \text{in } \Gamma = V(\Gamma).$$

This object e of \mathcal{D} yields two equation morphisms: $\tilde{e}: X \rightarrow H_\lambda X + R_\lambda$ above, and analogously $\hat{e} = (\text{id} + i^T) \cdot e: X \rightarrow H_\lambda X + T_\lambda$. The solution of the latter is the unique morphism

$$\hat{e}^\dagger: X \rightarrow T_\lambda \quad \text{with} \quad \hat{e}_{\bar{\Gamma}}^\dagger(x_s) = s \text{ for all } s \in \Gamma_0.$$

Indeed, Diagram (2.3) is easily seen to commute for \hat{e} and \hat{e}^\dagger . In (I) above we saw that the solution $\tilde{e}^\dagger: X \rightarrow R_\lambda$ is a codomain restriction of \hat{e}^\dagger . In particular:

$$t = \hat{e}_{\bar{\Gamma}}^\dagger(x_t).$$

This proves (a).

To prove (b) let $\tau: H_\lambda T_\lambda \rightarrow T_\lambda$ denote the algebra structure of T_λ . By Theorem 2.10 and Example 2.9 we have that

$$[\tau, i^T]: H_\lambda T_\lambda + V \rightarrow T_\lambda \quad \text{is an isomorphism.}$$

From Diagram (2.3) we get

$$\hat{e}^\dagger = [\tau, \text{id}_{T_\lambda}] \cdot [H_\lambda \hat{e}^\dagger + \text{id}_{T_\lambda}] \cdot (\text{id}_{H_\lambda X} + i^T) \cdot e$$

which yields

$$[\tau, i^T]^{-1} \cdot \hat{e}^\dagger = (H_\lambda \hat{e}^\dagger + \text{id}_V) \cdot e.$$

Let us factorize \hat{e}^\dagger as a strong epimorphism $k: X \rightarrow Y$ followed by a monomorphism $m: Y \rightarrow T_\lambda$. Then the last equation makes it possible to apply the diagonal fill in:

$$\begin{array}{ccccc}
X & \xrightarrow{k} & Y & & \\
e \downarrow & & \swarrow f & m \downarrow & \\
H_\lambda X + V & & T_\lambda & & \\
H_\lambda k + \text{id} \downarrow & & \downarrow [\tau, i^T]^{-1} & & \\
H_\lambda Y + V & \xrightarrow[H_\lambda m + \text{id}_V]{} & H_\lambda T_\lambda + V & &
\end{array}$$

Indeed, $H_\lambda = (-)^2 + \delta$ preserves connected limits (because each summand does), thus, monomorphisms; consequently, $H_\lambda m + \text{id}_Y$ is a monomorphism. Since Y is a strong quotient of X , it follows from Theorem 2.16 that Y is finitely presentable. Thus,

$$f: Y \rightarrow H_\lambda Y + V$$

is an object of \mathcal{D} , and clearly k is a connecting morphism from e to f .

From (I) we know that \tilde{e}^\dagger is the domain restriction of \hat{e}^\dagger , thus we see that $\tilde{e}_\Gamma^\dagger(x) = \tilde{e}_\Gamma^\dagger(x')$ implies $\hat{e}_\Gamma^\dagger(x) = \hat{e}_\Gamma^\dagger(x')$, and since m_Γ is a monomorphism with $\hat{e}_\Gamma^\dagger = m_\Gamma \cdot k_\Gamma$, we conclude

$$k_\Gamma(x) = k_\Gamma(x')$$

as requested. \square

Remark 2.18. As mentioned in the Introduction we want to combine application and abstraction with other operations. Suppose $\Sigma = (\Sigma_n)_{n \in \mathbb{N}}$ is a signature (of “terminals”). Then we can form the endofunctor $H_{\lambda, \Sigma}$ of $\mathbf{Set}^{\mathcal{F}}$ on objects by

$$H_{\lambda, \Sigma} X = X \times X + \delta X + \coprod_{n \in \mathbb{N}} \Sigma_n \bullet X^n$$

where $\Sigma_n \bullet X^n$ is the coproduct (that is: disjoint union in every context) of Σ_n copies of the n -th Cartesian power of X . For this endofunctor an algebra is an H_λ -algebra A together with an n -ary operation on $A(\Gamma)$ for every $\sigma \in \Sigma_n$ and every context Γ .

In the following result we use notation of Example 2.2(vii).

Theorem 2.19. *For every signature Σ*

- (i) $F_{\lambda, \Sigma}$ *is the free $H_{\lambda, \Sigma}$ -algebra on V ,*
- (ii) $R_{\lambda, \Sigma}$ *is the free iterative $H_{\lambda, \Sigma}$ -algebra on V , and*
- (iii) $T_{\lambda, \Sigma}$ *is the free completely iterative $H_{\lambda, \Sigma}$ -algebra on V .*

Indeed, (i) was proved in [10], and the proofs of (ii) and (iii) are completely analogous to the proofs of Theorems 2.17 and 2.10.

3. PRESHEAVES AS MONOIDS

So far we have not treated one of the basic features of λ -calculus: substitution of subterms. For the presheaf $F_{\lambda, \Sigma}$ of finite λ - Σ -terms this was elegantly performed by Fiore *et al* [10] based on the monoidal structure of the category $\mathbf{Set}^{\mathcal{F}}$. As mentioned in Notation 2.1(3), we can work with the equivalent category $\text{Fin}(\mathbf{Set}, \mathbf{Set})$ of all finitary endofunctors of \mathbf{Set} . Composition of functors makes this a (strict, non-symmetric) monoidal

category with unit $\text{Id}_{\mathbf{Set}}$. This monoidal structure, as shown in [10], corresponds to simultaneous substitution. Indeed, let X and Y be objects of $\text{Fin}(\mathbf{Set}, \mathbf{Set})$. Then the “formulas of the composite presheaf $X \cdot Y$ ” in context Γ are the elements of

$$X \cdot Y(\Gamma) = X(Y(\Gamma)) = \bigcup_{u: \bar{\Gamma} \hookrightarrow Y(\Gamma)} Xu[\bar{\Gamma}], \quad (3.1)$$

where $u: \bar{\Gamma} \hookrightarrow Y(\Gamma)$ ranges over finite subobjects of $Y(\Gamma)$. Indeed, X preserves the filtered colimit $Y(\Gamma) = \text{colim } \bar{\Gamma}$.

Consequently, in order to specify an $X \cdot Y$ -formula t in context Γ we need (a) an X -formula s in some new context $\bar{\Gamma}$ and (b) for every variable $x \in \bar{\Gamma}$ a Y -formula of context Γ , say, r_x . We can then think of t as the formula $s(r_x/x)$ obtained from s by simultaneous substitution.

Example 3.1. We consider the presheaves $F_{\lambda, \Sigma}$ and $F_{\lambda, \Sigma'}$, where Σ is the signature with a binary operation symbol $*$ and Σ' a signature with a unary operation symbol o . Then for every context Γ , the elements of $F_{\lambda, \Sigma} \cdot F_{\lambda, \Sigma'}(\Gamma)$ are λ - Σ -terms in some context $\bar{\Gamma}$ with free variables replaced by λ - Σ' -terms in context Γ . For a concrete example, let $\bar{\Gamma} = \{y, z\}$ and $\Gamma = \{z'\}$ and consider the λ - Σ -term

$$t = \lambda x. x * (y * z) \quad \text{in } F_{\lambda, \Sigma}\{y, z\}$$

and the function

$$u: \bar{\Gamma} \rightarrow F_{\lambda, \Sigma'}(\Gamma) \quad \text{with} \quad \begin{aligned} u(y) &= \lambda x. o(x) @ z' \\ u(z) &= z' @ o(o(z')) \end{aligned}$$

Then the element of $F_{\lambda, \Sigma} \cdot F_{\lambda, \Sigma'}(\Gamma)$ corresponding to t and u is the term

$$\lambda x. x * ((\lambda x. o(x) @ z') * (z' @ o(o(z')))).$$

Remark 3.2.

- (i) The monoidal structure on $\mathbf{Set}^{\mathcal{F}}$ corresponding to composition in $\text{Fin}(\mathbf{Set}, \mathbf{Set})$ will be denoted by \otimes . Its unit (corresponding to Id) is V , see Notation 2.2(i). Observe that every endofunctor $- \otimes X$ preserves colimits, e.g., $(A + B) \otimes X \cong (A \otimes X) + (B \otimes X)$.
- (ii) Explicitly, the monoidal structure can be described by the coend

$$(X \otimes Y)(\Gamma) = \int^{\bar{\Gamma}} \mathbf{Set}(\bar{\Gamma}, Y(\Gamma)) \bullet X(\bar{\Gamma}). \quad (3.2)$$

- (iii) Recall that monoids in the monoidal category $\text{Fin}(\mathbf{Set}, \mathbf{Set})$ are precisely the finitary monads on \mathbf{Set} .
- (iv) The presheaf $F_{\lambda, \Sigma}$ is endowed with the usual simultaneous substitution of λ -terms which defines a morphism $m^F: F_{\lambda, \Sigma} \otimes F_{\lambda, \Sigma} \rightarrow F_{\lambda, \Sigma}$. Together with the canonical pointing $i^F: V \rightarrow F_{\lambda, \Sigma}$, see Remark 2.6, this constitutes a monoid as proved in [10].

Analogously the simultaneous substitution of infinite λ -terms defines a monoid

$$(T_{\lambda, \Sigma}, m^T, i^T).$$

It is easy to see that given a rational term, every simultaneous substitution of rational terms for variables yields again a rational term. Thus, we have a submonoid $(R_{\lambda, \Sigma}, m^R, i^R)$.

- (v) The monoidal operation of $F_{\lambda,\Sigma}$ is well connected to its structure of an $H_{\lambda,\Sigma}$ -algebra. This was expressed in [10] by the concept of an $H_{\lambda,\Sigma}$ -monoid.

In order to recall this concept, we need the notion of point-strength introduced in [9] under the name (I/\mathcal{W}) -strength; this is a weakening of the classical strength (necessary since $H_{\lambda,\Sigma}$ is unfortunately not strong). Recall that given an object I of a category \mathcal{W} , then objects of the slice category I/\mathcal{W} are morphisms $x: I \rightarrow X$ for $X \in \text{obj } \mathcal{W}$.

Definition 3.3 (see [9]). Let $(\mathcal{W}, \otimes, I)$ be a strict monoidal category and H an endofunctor on \mathcal{W} . A **point-strength** of H is a collection of morphisms

$$s_{(X,x)(Y,y)}: HX \otimes Y \rightarrow H(X \otimes Y)$$

natural in (X, x) and (Y, y) ranging through I/\mathcal{W} such that

- (i) $s_{(X,x)(I,\text{id})} = \text{id}_{HX}$, and
- (ii) the following triangles commute:

$$\begin{array}{ccc} HX \otimes Y \otimes Z & \xrightarrow{s_{(X,x),(Y \otimes Z,y \otimes z)}} & H(X \otimes Y \otimes Z) \\ & \searrow s_{(X,x),(Y,y)} \otimes \text{id}_Z & \nearrow s_{(X \otimes Y,x \otimes y),(Z,z)} \\ & H(X \otimes Y) \otimes Z & \end{array} \quad (3.3)$$

Example 3.4.

- (i) The endofunctor $X \mapsto X \otimes X$ (which usually fails to be strong) has the point-strength $s_{(X,x)(Y,y)} = (X \otimes X) \otimes Y = (X \otimes I \otimes X) \otimes Y \xrightarrow{\text{id}_X \otimes y \otimes \text{id}_{X \otimes Y}} (X \otimes Y) \otimes (X \otimes Y)$.
- (ii) The endofunctor $X \mapsto X^n$ of $\mathbf{Set}^{\mathcal{F}}$ is clearly (point-)strong for every $n \in \mathbb{N}$.
- (iii) The functor δ in Notation 2.3 is point-strong, as observed in [10]. The easiest way to describe its point-strength is by working in $\text{Fin}(\mathbf{Set}, \mathbf{Set})$. Given pointed endofunctors $x: \text{Id} \rightarrow X$ and $y: \text{Id} \rightarrow Y$, then the point-strength $s_{(X,x)(Y,y)}: (\delta X) \cdot Y \rightarrow \delta(X \cdot Y)$ has components

$$X(Y(\Gamma) + 1) \xrightarrow{X(\text{id} + y_1)} X(Y(\Gamma) + Y(1)) \xrightarrow{X \text{ can}} X \cdot Y(\Gamma + 1),$$

where $\text{can}: Y(\Gamma) + Y(1) \rightarrow Y(\Gamma + 1)$ denotes the canonical morphism.

- (iv) A coproduct of point-strong functors is point-strong.

Corollary 3.5. *The endofunctors H_λ and $H_{\lambda,\Sigma}$ are point-strong. Their point-strength is denoted by s^H .* \square

Definition 3.6 (see [10]). Let H be a point-strong endofunctor of a monoidal category. By an **H -monoid** is meant an H -algebra (A, a) which is also a monoid

$$m: A \otimes A \rightarrow A \quad \text{and} \quad i: I \rightarrow A$$

such that the square below commutes:

$$\begin{array}{ccccc} HA \otimes A & \xrightarrow{s_{(A,i)(A,i)}} & H(A \otimes A) & \xrightarrow{Hm} & HA \\ \downarrow a \otimes \text{id} & & & & \downarrow a \\ A \otimes A & \xrightarrow{m} & A & & \end{array} \quad (3.4)$$

Remark 3.7.

- (1) Homomorphisms of H -monoids are those monoid homomorphisms which are also H -algebra homomorphisms.
- (2) An H -monoid is called (*completely*) *iterative* if its underlying H -algebra has this property.

Example 3.8.

- (1) F_λ is an H_λ -monoid. Indeed, we know that substitution yields the monoid structure (Remark 2.6) and tree tupling yields the algebra structure (Example 2.5). Let us consider the square

$$\begin{array}{ccc} H_\lambda F_\lambda \otimes F_\lambda & \xrightarrow{s} & H_\lambda(F_\lambda \otimes F_\lambda) \xrightarrow{Hm^F} HF_\lambda \\ \varphi \otimes \text{id} \downarrow & & \downarrow \varphi \\ F_\lambda \otimes F_\lambda & \xrightarrow{m^F} & F_\lambda \end{array}$$

The elements t of $H_\lambda F \otimes F_\lambda$ in context Γ are those of

$$H_\lambda F_\lambda(\Gamma_0) = F_\lambda(\Gamma_0) \times F_\lambda(\Gamma_0) + F_\lambda(\Gamma_0 + \{x\})$$

for a given context Γ_0 together with a substitution $f: \Gamma_0 \rightarrow F_\lambda(\Gamma)$. In case of the summand $F_\lambda(\Gamma_0) \times F_\lambda(\Gamma_0)$ the lower passage $m_\Gamma^F(\varphi_\Gamma \otimes \text{id})$ assigns to $t = (t_1, t_2)$ the term $t_1 @ t_2$ with variables substituted according to f . And the upper passage first substitutes to t_1 and t_2 according to f separately, and then forms $@$; the result is the same. In case of the summand $F_\lambda(\Gamma_0 + \{x\})$ the lower passage assigns to t the term $\lambda x.t$ with variables substituted according to f ; the upper one first substitutes in t and then forms $\lambda x.-$ yielding the same result again.

- (2) More generally, for every signature Σ we have an $H_{\lambda,\Sigma}$ -monoid $F_{\lambda,\Sigma}$.

Theorem 3.9 (see [10]). *The presheaf $F_{\lambda,\Sigma}$ of finite λ - Σ -terms is the initial $H_{\lambda,\Sigma}$ -monoid.*

Theorem 3.10 (see [16]). *The presheaf $T_{\lambda,\Sigma}$ of λ - Σ -terms is an $H_{\lambda,\Sigma}$ -monoid with simultaneous substitution as monoid structure.*

Although in [16], Example 13, just T_λ is used, the methods of that paper apply to $T_{\lambda,\Sigma}$ immediately. The following theorem proves a stronger property of $T_{\lambda,\Sigma}$, corresponding to Theorem 3.9 above.

Theorem 3.11. *The presheaf $T_{\lambda,\Sigma}$ of λ - Σ -terms is the initial completely iterative $H_{\lambda,\Sigma}$ -monoid.*

An elementary proof of this theorem was presented in [4]. Here we will prove a more general result in Theorem 4.18 below.

4. THE INITIAL ITERATIVE H -MONOID

The aim of this section is to prove that the presheaf $R_{\lambda,\Sigma}$ of rational λ - Σ -terms is the initial iterative $H_{\lambda,\Sigma}$ -monoid in $\mathbf{Set}^{\mathcal{F}}$. We have (in contrast to the characterization of T_λ in the preceding section) no elementary proof. Rather, we need to work with the monad $\mathbb{R}_{\lambda,\Sigma}$ of free iterative $H_{\lambda,\Sigma}$ -algebras on $\mathbf{Set}^{\mathcal{F}}$ (for which $R_{\lambda,\Sigma}$ is $\mathbb{R}_{\lambda,\Sigma}(V)$) and prove that it is point-strong and use this strength further. We will actually work in a more general setting (which can be applied later for the case of typed λ -calculus).

Assumption 4.1. Throughout this section we assume that H is a finitary endofunctor of \mathcal{W} where

- (1) \mathcal{W} is a locally finitely presentable category, i.e., a cocomplete category with a set of finitely presentable objects \mathcal{W}_{fp} whose closure under filtered colimits is all of \mathcal{W} .
- (2) \mathcal{W} is also a strict monoidal category with the unit I finitely presentable and the tensor product preserving finite presentability: if A, B are finitely presentable, then so is $A \otimes B$.
- (3) \mathcal{W} is right distributive, that is, for every object W the endofunctor $- \otimes W$ preserves finite coproducts.
- (4) The tensor product is a finitary functor, i.e., its preserves filtered colimits (in both variables).

We call categories satisfying (1)–(4) *monoidally locally finitely presentable*.

Example 4.2. Set is, as a cartesian closed category, monoidally locally finitely presentable. For every monoidally locally finitely presentable category \mathcal{W} all functor categories $\mathcal{W}^{\mathcal{A}}$, \mathcal{A} small, have the property too; for example, $\mathbf{Set}^{\mathcal{F}}$ with the cartesian product as tensor. However, in our paper we only use the fact that $\mathbf{Set}^{\mathcal{F}}$ is a monoidally locally finitely presentable w.r.t. \otimes in Remark 3.2(i). This follows from the fact that this is equivalent to $\text{Fin}(\mathbf{Set}, \mathbf{Set})$ with the tensor product given by composition. Observe that \otimes is right distributive (since precomposition with a given functor preserves colimits) but not left distributive.

Notation 4.3. For every object Y of \mathcal{W} we denote by

$$\varrho_Y : HRY \rightarrow RY \quad \text{and} \quad \eta_Y : Y \rightarrow RY$$

the structure and universal morphism of the free iterative H -algebra on Y , which exists as proved in [2]. This gives rise to the monad

$$\mathbb{R} = (R, \eta, \mu)$$

where $\mu_X : RRY \rightarrow RY$ is the unique homomorphism extending η_Y :

$$\begin{array}{ccccc}
 & HRRY & \xrightarrow{\varrho_{RY}} & RRY & \xleftarrow{\eta_{RY}} Y \\
 & \downarrow H\mu_Y & & \downarrow \mu_Y & \searrow \eta_Y \\
 HRY & \xrightarrow{\varrho_Y} & RY & &
 \end{array} \tag{4.1}$$

\mathbb{R} is called the *rational monad* of the endofunctor H .

Remark 4.4. In [2] we described the free iterative H -algebra RY as the colimit of the diagram of all “flat equation” morphisms

$$e : W \rightarrow HW + Y, \quad W \in \mathcal{W} \text{ finitely presentable},$$

whose connecting morphisms (“equation morphisms”) are just the coalgebra homomorphisms h for the endofunctor $H(-) + Y$:

$$\begin{array}{ccc} W & \xrightarrow{e} & HW + Y \\ h \downarrow & & \downarrow Hh + \text{id} \\ W' & \xrightarrow{e'} & HW' + Y \end{array} \quad (4.2)$$

More detailed:

- (i) The category EQ_Y of all flat equation morphisms in Y is filtered. The filtered diagram

$$\text{Eq}_Y : \text{EQ}_Y \rightarrow W, \quad \text{Eq}_Y(W \xrightarrow{e} HW + Y) = W$$

has a colimit RY with the colimit injections $e^\# : W \rightarrow RY$.

- (ii) For the flat equation morphism $\text{inl} : Y \rightarrow HY + Y$ put

$$\eta_Y = \text{inr}^\# : Y \rightarrow RY. \quad (4.3)$$

- (iii) There is a unique isomorphism $i : RY \rightarrow HRY + Y$ such that the squares

$$\begin{array}{ccc} W & \xrightarrow{e} & HW + Y \\ e^\# \downarrow & & \downarrow He^\# + Y \\ RY & \xrightarrow{i_Y} & HRY + Y \end{array} \quad (4.4)$$

commute for all flat equations e . Put

$$\varrho \equiv HRY \xrightarrow{\text{inl}} HRY + Y \xrightarrow{i_Y^{-1}} RY.$$

Then RY together with η_Y and ϱ_Y is the free iterative H -algebra on Y . We also have

$$i_Y = [\varrho_Y, \eta_Y]^{-1}. \quad (4.5)$$

Furthermore, $e^\#$ is the unique coalgebra homomorphism from e to i_Y .

- (iv) For every $e : W \rightarrow HW + Y$ the morphism $e^\# : W \rightarrow RY$ is the unique solution (in the iterative algebra RY) of

$$\eta_Y \bullet e \equiv W \xrightarrow{e} HW + Y \xrightarrow{HW + \eta_Y} HW + RY.$$

- (v) Let

$$e : W \rightarrow HW + RY \quad \text{and} \quad e' : W' \rightarrow HW' + RY$$

be two equation morphisms with W and W' finitely presentable, and let h be a coalgebra homomorphism from (W, e) to (W', e') . Then for the unique solutions of e and e' we have

$$e^\dagger = (e')^\dagger \cdot h : W \rightarrow RY.$$

- (v) Suppose we have two morphisms

$$f : V \rightarrow HV + W \quad \text{and} \quad e : W \rightarrow HW + RY$$

where V, W are finitely presentable. Then we can form an equation morphism

$$\begin{array}{ccccc}
 e \square f \equiv V + W & \xrightarrow{[f, \text{inr}]} & HV + W & \xrightarrow{HV + e} & HV + HW + RY \\
 & & & & \downarrow \text{can} + RY \\
 & & & & H(V + W) + RY
 \end{array}$$

and we have

$$(e^\dagger \bullet f)^\dagger = (e \square f)^\dagger \cdot \text{inl}, \quad (4.6)$$

see [3].

- (vi) Finally, every homomorphism $h: A \rightarrow B$ of H -algebras between iterative algebras A and B preserves solutions:

$$h \cdot e^\dagger = (h \bullet e)^\dagger: X \rightarrow B$$

for every equation morphism $e: X \rightarrow HX + A$.

Example 4.5. The rational monad of H_λ is the monad \mathbb{R}_λ of rational λ -terms with constants: to every presheaf Y it assigns the presheaf $R_\lambda(Y)$ defined precisely as R_λ in Example 2.2(v) except that in every context Γ we can also use elements of $Y(\Gamma)$ to label the leaves.

More detailed: we first define the set $R'_\lambda(Y)(\Gamma)$ of rational trees in context Γ with constants from Y . It consists of all rational trees of the form (2.2) such that

a node labelled by an element of $Y(\Gamma)$ is a leaf.

By using the α -conversion precisely as in Example 2.2(iii), we obtain the desired presheaf

$$R_\lambda(Y)(\Gamma) = R'_\lambda(Y)(\Gamma)/\sim_\alpha.$$

It is again pointed; the pointing $i^{R_\lambda(Y)}: V \rightarrow R_\lambda(Y)$ assigns to every variable the corresponding singleton tree. And $R_\lambda(Y)$ is canonically an H_λ -algebra. We define

$$\eta_Y: Y \rightarrow R_\lambda(Y)$$

to assign to every element of $Y(\Gamma)$ the corresponding singleton tree. This is the free iterative H_λ -algebra on Y , the proof is completely analogous to that of Theorem 2.17.

Definition 4.6. A *point-strong monad* is a monad $\mathbb{M} = (M, \eta, \mu)$ on \mathcal{W} together with a point-strength

$$s_{(X,x),(Y,y)}: (MX) \otimes Y \rightarrow M(X \otimes Y)$$

see Definition 3.3, such that s preserves the unit:

$$\begin{array}{ccc}
 MX \otimes Y & \xrightarrow{s_{(X,x),(Y,y)}} & M(X \otimes Y) \\
 \eta_{X \otimes Y} \swarrow & & \searrow \eta_{X \otimes Y} \\
 X \otimes Y & &
 \end{array} \quad (4.7)$$

and the multiplication:

$$\begin{array}{ccccc}
 MMX \otimes Y & \xrightarrow{s_{(MX,\eta_X \cdot x),(Y,y)}} & M(MX \otimes Y) & \xrightarrow{Ms_{(X,x),(Y,y)}} & MM(X \otimes Y) \\
 \mu_{X \otimes Y} \downarrow & & & & \downarrow \mu_Y \\
 MX \otimes Y & \xrightarrow{s_{(X,x),(Y,y)}} & M(X \otimes Y) & &
 \end{array} \quad (4.8)$$

Example 4.7. By our assumption that H be finitary we know that all terminal coalgebras for $H(-) + X$ exist, this follows from [15], see also [5]. Equivalently, all free completely iterative algebras for H exist (cf. Example 2.9), and they yield the object map of a monad $\mathbb{T} = (T, \eta^T, \mu^T)$. This monad is the free completely iterative monad on the endofunctor H , see [17]. The monad multiplication $\mu_X^T: TTX \rightarrow TX$ is the unique algebra homomorphism extending id_{TX} , i.e., such that

$$\mu_X^T \cdot \tau_{TX} = \tau_X \cdot H\mu_X^T \quad \text{and} \quad \mu_X^T \cdot \eta_{TX}^T = \text{id}_{TX}. \quad (4.9)$$

Theorem 4.8. *The free completely iterative monad \mathbb{T} of a point-strong endofunctor H is point-strong.*

Remark. The strength of \mathbb{T} will be proved to be the unique natural transformation s^T for which the diagram

$$\begin{array}{ccccc} HTX \otimes Y & \xrightarrow{s^H} & H(TX \otimes Y) & \xrightarrow{Hs^T} & HT(X \otimes Y) \\ \tau_{X \otimes Y} \downarrow & & & & \downarrow \tau_{X \otimes Y} \\ TX \otimes Y & \xrightarrow{s^T} & & & T(X \otimes Y) \\ \eta_X^T \otimes Y \uparrow & & \nearrow \eta_{X \otimes Y}^T & & \\ X \otimes Y & & & & \end{array} \quad (4.10)$$

commutes. Note that we have dropped the subscripts indicating the components of the natural transformations s^H and s^T above; from now on we shall frequently do this when components of natural transformations are clear from the context.

Proof. (a) Let (X, x) and (Y, y) be pointed objects. For every morphism $f: X \otimes Y \rightarrow TZ$ there exists a unique morphism $f^\flat: TX \otimes Y \rightarrow TZ$ such that that the diagram

$$\begin{array}{ccccc} HTX \otimes Y & \xrightarrow{s^H} & H(TX \otimes Y) & \xrightarrow{Hf^\flat} & HTZ \\ \tau \otimes Y \downarrow & & & & \downarrow \tau \\ TX \otimes Y & \xrightarrow{f^\flat} & & & TZ \\ \eta \otimes Y \uparrow & & \nearrow f & & \\ X \otimes Y & & & & \end{array} \quad (4.11)$$

commutes. Indeed, the algebra TZ is completely iterative. Due to $(HTX + Y) \otimes Y = HTX \otimes Y + X \otimes Y$, see Assumption 4.1(3), we obtain an equation morphism in TZ as follows:

$$TX \otimes Y \xrightarrow{[\tau_X, \eta_X^T]^{-1} \otimes Y} HTX \otimes Y + X \otimes Y \xrightarrow{s_{X,Y}^H + f} H(TX \otimes Y) + TZ.$$

Its unique solution is denoted by f^\flat . It is characterized by the commutative diagram

$$\begin{array}{ccc}
TX \otimes Y & \xrightarrow{f^\flat} & TZ \\
\uparrow [\tau, \eta^T] \otimes Y & & \uparrow [\tau, TZ] \\
HTX \otimes Y + X \otimes Y & & \\
\downarrow s^H + f & & \\
H(TX \otimes Y) + TZ & \xrightarrow{Hf^\flat + TZ} & HTZ + TZ
\end{array}$$

It is easy to verify that this diagram commutes iff (4.11) does.

(b) Put

$$s_{(X,x),(Y,y)}^T = (\eta_{X \otimes Y}^T)^\flat : TX \otimes Y \rightarrow T(X \otimes Y).$$

In other words, we define the components of s^T via (4.11) uniquely.

(b1) s^T is natural: the squares

$$\begin{array}{ccc}
TX \otimes Y & \xrightarrow{s^T} & T(X \otimes Y) \\
\downarrow Tg \otimes h & & \downarrow T(g \otimes h) \\
T'X \otimes Y' & \xrightarrow{s^T} & T(X' \otimes Y')
\end{array}$$

commute for all morphisms g and h of I/\mathcal{W} since both passages form f^\flat for

$$f = \eta_{X' \otimes Y'}^T \cdot (g \otimes h) : X \otimes Y \rightarrow T(X' \otimes Y').$$

Indeed, for the upper passage, $f^\flat = T(g \otimes h) \cdot s^T$, use the following diagram:

$$\begin{array}{ccccccc}
HTX \otimes Y & \xrightarrow{s^H} & HTX \otimes Y & \xrightarrow{Hs^T} & HT(X \otimes Y) & \xrightarrow{HT(h \otimes g)} & HT(X' \otimes Y') \\
\downarrow H\tau_{X \otimes Y} & & & & \downarrow \tau_{X \otimes Y} & & \downarrow \tau_{X' \otimes Y'} \\
TX \otimes Y & \xrightarrow{s^T} & T(X \otimes Y) & \xrightarrow{T(h \otimes g)} & T(X' \otimes Y') & & \\
\uparrow \eta_{X \otimes Y}^T & \nearrow \eta_{X \otimes Y}^T & & & & & \uparrow \eta_{X' \otimes Y'}^T \\
X \otimes Y & \xrightarrow{h \otimes g} & & & & &
\end{array}$$

The two left-hand parts form Diagram (4.11), the remaining two commute by naturality of τ and η .

The lower passage $f^\flat = s^T \cdot (Tg \otimes h)$ follows from the following diagram:

$$\begin{array}{ccccccc}
HTX \otimes Y & \xrightarrow{HTg \otimes h} & HTX' \otimes Y' & \xrightarrow{s^H} & H(T(X') \otimes Y') & \xrightarrow{Hs^T} & HT(X' \otimes Y') \\
\tau_{X \otimes Y} \downarrow & & \downarrow \tau_{X' \otimes Y'} & & & & \downarrow \tau_{X' \otimes Y'} \\
TX \otimes Y & \xrightarrow{Tg \otimes h} & TX' \otimes Y' & \xrightarrow{s^T} & T(X' \otimes Y') & & \\
\eta_{X \otimes Y}^T \uparrow & & \eta_{X' \otimes Y'}^T \uparrow & & \eta_{X' \otimes Y'}^T & & \\
X \otimes Y & \xrightarrow{g \otimes h} & X' \otimes Y' & & & &
\end{array}$$

The right-hand parts form Diagram (4.12), the left-hand ones commute by naturality of τ and η^T .

(b2) s^T is a point-strength of the endofunctor T . Indeed, the axiom

$$s_{(X,x)(V,v)}^T = \text{id}_{T(X)} \quad (4.12)$$

follows from the fact that if $(Y,y) = (V,\text{id})$, then Diagram (4.10) commutes with $\text{id}_{T(X)}$ in lieu of s^T . To verify the Axiom (3.3), apply (a) to $f = \eta_{X \otimes Y \otimes Z}^T$: we prove that the lower passage of (3.3) serves as f^\flat . In detail, the diagram

$$\begin{array}{ccccc}
HT(X \otimes Y) \otimes Z & \xrightarrow{s^H} & H(T(X \otimes Y) \otimes Z) & & \\
\uparrow Hs^T \otimes Z & & \downarrow Hs^T & & \\
HTX \otimes Y \otimes Z & \xrightarrow{s^H \otimes Z} & H(TX \otimes Y) \otimes Z & \xrightarrow{\tau_{X \otimes Y \otimes Z}} & HT(X \otimes Y \otimes Z) \\
\tau_{X \otimes Y \otimes Z} \downarrow & & \downarrow \tau_{X \otimes Y \otimes Z} & & \downarrow \tau_{X \otimes Y \otimes Z} \\
TX \otimes Y \otimes Z & \xrightarrow{s^T \otimes Z} & T(X \otimes Y) \otimes Z & \xrightarrow{s^T} & T(X \otimes Y \otimes Z) \\
\eta_{X \otimes Y \otimes Z}^T \uparrow & \nearrow \eta_{X \otimes Y \otimes Z}^T & & \nearrow \eta_{X \otimes Y \otimes Z}^T & \\
X \otimes Y \otimes Z & & & &
\end{array}$$

commutes. Indeed, all inner parts commute by two applications of (4.10).

(b3) It remains to verify the axioms of Definition 4.6. For (4.7) use the lower triangle of Diagram (4.10). For (4.8) apply (a) to

$$f = s^T : TX \otimes Y \rightarrow T(X \otimes Y).$$

We prove that both passages of (4.8) serve as f^\flat . For the lower passage, $(s^T)^\flat = s^T \cdot \mu^T \otimes Y$, use the following diagram

$$\begin{array}{ccccc}
HTTX \otimes Y & \xrightarrow{s^H} & HT(TX \otimes Y) & \xrightarrow{H(\mu^T \otimes Y)} & HT \\
\downarrow \tau \otimes Y & \searrow H\mu^T \otimes Y & \downarrow \tau \otimes Y & \nearrow s^H & \downarrow \tau \\
TTX \otimes Y & \xrightarrow{\mu^T \otimes Y} & TX \otimes Y & \xrightarrow{s^T} & T(X \otimes Y) \\
\uparrow \eta^T T \otimes Y & \nearrow \text{double} & & \nearrow s^T & \\
TX \otimes Y & & & &
\end{array}$$

The upper left-hand part is Equation (4.9), the lower one commutes by the monad axiom $\mu^T \cdot \eta^T T = \text{id}$, the upper triangle is the naturality of s^H , and the right-hand part follows from (4.10).

For the upper passage, $(s^T)^b = \mu^T \cdot Ts^T \cdot s^T$, use the following diagram

$$\begin{array}{ccccccc}
& & H(TTX \otimes Y) & & & & \\
& \swarrow s^H & \downarrow & \searrow Hs^T & & & \\
HTTX \otimes Y & & HT(TX \otimes Y) & \xrightarrow{HTs^T} & HTT(X \otimes Y) & \xrightarrow{H\mu^T} & HT(X \otimes Y) \\
\downarrow \tau \otimes Y & & \downarrow \tau & & \downarrow \tau & & \downarrow \tau \\
TTX \otimes Y & \xrightarrow{s^T} & T(TX \otimes Y) & \xrightarrow{Ts^T} & TT(X \otimes Y) & \xrightarrow{\mu^T} & T(X \otimes Y) \\
\uparrow \eta^T \otimes Y & \nearrow \eta^T & & & \nearrow s^T & & \\
TX \otimes Y & & & & & &
\end{array}$$

The three upper squares commute due to (4.10), the naturality of τ and (4.9). The lower triangles commute due to (4.10), the naturality of s^T and $\mu^T \cdot \eta^T T = \text{id}$. \square

Remark 4.9. Recall from Example 2.9 that $T = HT + \text{Id}$ with injections τ and η^T . From the Diagram (4.10) we see that the strength s^T then has the form

$$s^T = Hs^T \cdot s^H + X \otimes Y : HTX \otimes Y + X \otimes Y \rightarrow HT(X \otimes Y) + X \otimes Y.$$

Theorem 4.10. *The rational monad of a point-strong endofunctor is point-strong.*

Remark. The strength of \mathbb{R} will be proved to be the unique natural transformation s^R for which the diagram

$$\begin{array}{ccccc}
HRX \otimes Y & \xrightarrow{s^H} & H(RX \otimes Y) & \xrightarrow{Hs^R} & HR(X \otimes Y) \\
\varrho_{X \otimes Y} \downarrow & & & & \downarrow \varrho_{X \otimes Y} \\
RX \otimes Y & \xrightarrow{s^R} & & & R(X \otimes Y) \\
\eta_{X \otimes Y} \uparrow & & \nearrow \eta_{X \otimes Y} & & \\
X \otimes Y & & & &
\end{array} \tag{4.13}$$

commutes.

Proof. (a) Given pointed objects (X, x) and (Y, y) , we prove that for every morphism $f: X \otimes Y \rightarrow RZ$ there exists a unique morphism $f^\flat: RX \otimes Y \rightarrow RZ$ such that the following diagram commutes:

$$\begin{array}{ccccc}
HRX \otimes Y & \xrightarrow{s^H} & H(RX \otimes Y) & \xrightarrow{Hf^\flat} & HRZ \\
\varrho_{X \otimes Y} \downarrow & & & & \downarrow \varrho_Z \\
RX \otimes Y & \xrightarrow{f^\flat} & & & RZ \\
\eta_{X \otimes Y} \uparrow & & \nearrow f & & \\
X \otimes Y & & & &
\end{array} \tag{4.14}$$

(a1) Assume that Y is finitely presentable. Recall $RY = \text{colim } \mathbf{Eq}_Y$ from Remark 4.4. For every object

$$e: W \rightarrow HW + X \quad \text{in } \mathbf{Eq}_X$$

define, using the distributivity $(HW + Y) \otimes Y = HW \otimes Y + X \otimes Y$ (see Assumption 4.1(3)), the equation morphism

$$\hat{e} \equiv W \otimes Y \xrightarrow{e \otimes Y} HW \otimes Y + X \otimes Y \xrightarrow{s^H + f} H(W \otimes Y) + RZ. \tag{4.15}$$

Since $W \otimes Y$ is finitely presentable by Assumption 4.1(2), we obtain the unique solution $\hat{e}^\dagger: W \otimes Y \rightarrow RZ$, and those solutions form a cocone of the diagram $\mathbf{Eq}_X \otimes Y$. Indeed, given a connecting morphism

$$\begin{array}{ccc}
W & \xrightarrow{e} & HW + X \\
h \downarrow & & \downarrow Hh + X \\
W' & \xrightarrow{e'} & HW' + X
\end{array}$$

then $h \otimes Y$ is a coalgebra homomorphism from \hat{e} to \hat{e}' :

$$\begin{array}{ccccc}
W \otimes Y & \xrightarrow{e \otimes Y} & (HW \otimes Y) + (X \otimes Y) & \xrightarrow{s^H + f} & H(W \otimes Y) + RZ \\
h \otimes Y \downarrow & & \downarrow (Hh \otimes Y) + (X \otimes Y) & & \downarrow H(h \otimes Y) + RZ \\
W' \otimes Y & \xrightarrow{e' \otimes Y} & (HW' \otimes Y) + (X \otimes Y) & \xrightarrow{s^H + f} & H(W' \otimes Y) + RZ
\end{array}$$

which implies, by Remark 4.4(v) that

$$\hat{e}^\dagger = \hat{e}^\dagger \cdot (h \otimes Y).$$

Consequently, we can define

$$f^\flat: RX \otimes Y \rightarrow RZ$$

by the commutativity of the triangles

$$\begin{array}{ccc} W \otimes Y & & \\ \downarrow e^\# \otimes Y & \searrow \hat{e}^\dagger & \\ RX \otimes Y & \xrightarrow{f^\flat} & RZ \end{array} \quad \text{for all } e \in \mathbf{Eq}_X. \quad (4.16)$$

Indeed, since $- \otimes Y$ is a finitary functor by Assumption 4.1(4), we see that $RX \otimes Y$ is a colimit of $\mathbf{Eq}_Y \otimes Y$ with the colimit cocone $e^\# \otimes Y$. We now verify that the Diagram (4.14) commutes. Consider the diagram below:

$$\begin{array}{ccccc} W \otimes Y & \xrightarrow{e^\# \otimes Y} & RX \otimes Y & \xrightarrow{f^\flat} & RZ \\ \downarrow e \otimes Y & & \downarrow i_X \otimes Y & \downarrow [\varrho_X \otimes Y, \eta_X \otimes Y] & \uparrow [\varrho_Z, RZ] \\ HW \otimes Y + X \otimes Y & \xrightarrow{He^\# \otimes Y + X \otimes Y} & HRX \otimes Y + X \otimes Y & & \\ \downarrow s^H + f & & \downarrow s^H + f & & \\ H(W \otimes Y) + RZ & \xrightarrow{H(e^\# \otimes Y) + RZ} & H(RX \otimes Y) + RZ & \xrightarrow{Hf^\flat + RZ} & HRZ + RZ \end{array} \quad (4.17)$$

Notice first that the left-hand edge is \hat{e} . The upper left-hand part commutes by (4.4), and the lower one does by naturality of s^H . The outside of the diagram commutes since $f^\flat \cdot (e^\# \otimes Y)$ is the unique solution of \hat{e} in the iterative algebra RZ . Thus, the right-hand part commutes when precomposed by any $e^\# \otimes Y$. So since the latter morphisms are collectively epimorphic (being the injections of $\text{colim } \mathbf{Eq}_X \otimes Y$), we see that the right-hand part commutes. Now we use that i_X is an isomorphism with the inverse $[\varrho_X, \eta_X]$, see Equation (4.5), which implies

$$[\varrho_X \otimes Y, \eta_X \otimes Y] = (i_X \otimes Y)^{-1}.$$

Finally observe that the two coproduct components of the right-hand part of (4.17) yield precisely the upper and lower parts of (4.14)—this proves that (4.14) commutes.

It only remains to prove the uniqueness of f^\flat . So suppose we have some f^\flat such that Diagram (4.14) commutes. Equivalently, the right-hand part of (4.17) commutes, and this implies that $f^\flat \cdot (e^\# \otimes Y)$ is, for every e in \mathbf{Eq}_X , a solution of \hat{e} . This determines f^\flat uniquely.

(a2) Let Y be arbitrary. Then since \mathcal{W} is locally finitely presentable we can express Y as a filtered colimit

$$Y = \underset{q \in Q}{\text{colim}} Y^q \quad \text{with colimit cocone } y^q: Y^q \rightarrow Y$$

of finitely presentable objects Y^q . By Assumption 4.1(2) the unit object I is finitely presentable, thus the given pointing of Y :

$$y: I \rightarrow \operatorname{colim}_{q \in Q} Y^q$$

factorizes through some y^q . The diagram above being filtered, we can assume that this factorization takes place for every $q \in Q$, in other words, that we have a filtered diagram of pointed objects Y^q with colimit Y (and with all the connecting morphisms $Y^q \rightarrow Y^{q'}$ preserving the pointing).

Given $f: X \otimes Y \rightarrow RZ$, for every $q \in Q$ we know from the previous part (a1) that there exists a unique

$$f_q^\flat: RX \otimes Y^q \rightarrow RZ$$

such that Diagram (4.14) commutes when f^\flat is replaced by f_q^\flat and f by

$$f_q \equiv X \otimes Y^q \xrightarrow{X \otimes y^q} X \otimes Y \xrightarrow{f} RZ.$$

This defines a unique $f^\flat: RX \otimes Y \rightarrow RZ$ with

$$f_q^\flat = f^\flat \cdot (RX \otimes y^q) \quad \text{for all } q \in Q. \quad (4.18)$$

Now Diagram (4.14) commutes because $HRX \otimes Y = \operatorname{colim}_{q \in Q} HRX \otimes Y^q$ as well as $X \otimes Y = \operatorname{colim}_{q \in Q} X \otimes Y^q$. And f^\flat is uniquely determined by this commutativity; indeed, for any f^\flat such that (4.14) commutes one easily verifies that (4.18) holds using the uniqueness of f_q^\flat from part (a1).

(b) Analogously to the proof of Theorem 4.8 put

$$s_{(X,x),(Y,y)}^R = \eta_{X \otimes Y}^\flat: RX \otimes Y \rightarrow R(X \otimes Y). \quad (4.19)$$

The verification that s^R is the desired strength is analogous to the above proof: just replace T by R (and τ by ϱ). \square

Remark 4.11. The proofs of Theorems 4.8 and 4.10 have the same structure, and also the proof that the monad $\mathbb{F}_{\lambda,\Sigma}$ is point-strong can proceed analogously:

Let H be a point-strong endofunctor of \mathcal{W} and let $(\hat{M}, \hat{\mu}, \hat{\eta})$ be a monad. Suppose that a natural transformation $\alpha: H\hat{M} \rightarrow \hat{M}$ has the property that for every morphism $f: X \otimes Y \rightarrow MZ$ there exists a unique morphism $f^\flat: MX \otimes Y \rightarrow MZ$ with $f = f^\flat \cdot (\hat{\eta}_X \otimes Y)$ and $f^\flat \cdot (\alpha_X \otimes Y) = \alpha_X \cdot Hf^\flat \cdot s^H$. Then M is a point-strong monad w.r.t. $s^M = \hat{\eta}_{X \otimes Y}^\flat$.

Remark 4.12. The morphisms

$$\varrho_X: HRX \rightarrow RX \quad \text{and} \quad \eta_X: X \rightarrow RX$$

of (4.13) are coproduct injections of

$$RX = HRX + X$$

as proved in [2]. From diagram (4.13) we conclude that the strength of \mathbb{R} ,

$$s^R: RX \otimes Y \rightarrow R(X \otimes Y)$$

whose domain is $HRX \otimes Y + X \otimes Y$ by 4.1(2) and codomain is $HR(X \otimes Y) + X \otimes Y$, has the form

$$s^R = Hs^R \cdot s^H + X \otimes Y.$$

Corollary 4.13. *For a point-strong endofunctor H the free iterative H -algebra*

$$RI$$

on the unit object is an H -monoid w.r.t. the unit $i = \eta_I : I \rightarrow RI$ and the multiplication

$$m \equiv RI \otimes RI \xrightarrow{s_{I,RI}^R} RRI \xrightarrow{\mu_I} RI. \quad (4.20)$$

Proof. Indeed, the unit laws are obvious:

$$\begin{array}{ccccc} I \otimes RI & \xlongequal{\quad} & RI & & \\ \downarrow \eta_I \otimes RI & & \downarrow \eta_{RI} & & \\ RI \otimes RI & \xrightarrow{\quad} & RRI & \xrightarrow{\mu_I} & RI \\ \uparrow RI \otimes \eta_I & & \uparrow R\eta_I & & \\ RI \otimes I & \xrightarrow{s_{I,I}^R} & RI & & \end{array}$$

where the lower square commutes by the naturality of s^R . For the associativity we have the following commutative diagram

$$\begin{array}{ccccccc} & & RI \otimes m & & & & \\ & & \curvearrowright & & & & \\ RI \otimes RI \otimes RI & \xrightarrow{\quad} & RI \otimes RRI & \xrightarrow{\quad} & RI \otimes RI & & \\ \downarrow s_{I,RI}^R \otimes RI & & \downarrow s_{I,RRI}^R & & \downarrow s_{I,RI}^R & & \\ RRI \otimes RI & \xrightarrow{s_{RI,RI}^R} & R(RI \otimes RI) & \xrightarrow{Rs_{I,RI}^R} & RRRI & \xrightarrow{R\mu_I} & RRI \\ \downarrow \mu_I \otimes RI & & \downarrow (4.8) & & \downarrow \mu_{RI} & & \downarrow \mu_I \\ RI \otimes RI & \xrightarrow{s_{I,RI}^R} & RRI & \xrightarrow{\mu_I} & RI & & \end{array}$$

Finally, the Diagram (3.4) commutes due to (4.13):

$$\begin{array}{ccccc} & & Hm & & \\ & & \curvearrowright & & \\ HRI \otimes RI & \xrightarrow{s^H} & H(RI \otimes RI) & \xrightarrow{Hs^R} & HRRI \xrightarrow{H\mu} HRI \\ \downarrow \varrho \otimes RI & & \downarrow Hs^R & & \downarrow \varrho R \\ RI \otimes RI & \xrightarrow{s^R} & RRI & \xrightarrow{\mu} & RI \end{array}$$

□

Corollary 4.14. *The free completely iterative H -algebra*

$$TI$$

on the unit object is an H -monoid w.r.t. η_I^T and $\mu_I^T \cdot s_{I,TI}^T$.

The proof is completely analogous to the previous one.

Notation 4.15. Let

$$e: W \rightarrow HW + I$$

be a flat equation morphism in I , and let a pointing of W be given.

- (i) $e^\# : W \rightarrow RI$ denotes the colimit morphism of $RI = \text{colim } \mathbf{Eq}_I$ and

$$\widehat{e^\#} : RW \rightarrow RI$$

its unique extension to a homomorphism of H -algebras.

- (ii) $e * W$ denotes the following equation morphism in W :

$$e * W \equiv W \otimes W \xrightarrow{e \otimes W} HW \otimes W + W \xrightarrow{s^H + W} H(W \otimes W) + W.$$

- (iii) $\langle e \rangle : W \otimes W + W \rightarrow H(W \otimes W + W) + I$ denotes the flat equation morphism whose left-hand component is

$$\begin{array}{ccc} \langle e \rangle \cdot \text{inl} \equiv W \otimes W & \xrightarrow{e * W} & H(W \otimes W) + W \\ & & \downarrow H(W \otimes W) + e \\ & & H(W \otimes W) + HW + I \xrightarrow{\text{can} + I} H(W \otimes W + W) + I \end{array}$$

and the right-hand one is

$$\langle e \rangle \cdot \text{inr} \equiv W \xrightarrow{e} HW + I \xrightarrow{H \text{inr} + I} H(W \otimes W + W) + I.$$

Lemma 4.16. For every flat equation morphism $e: W \rightarrow HW + I$ with W pointed the square

$$\begin{array}{ccc} W \otimes W & \xrightarrow{(e * W)^\#} & RW \\ \text{inl} \downarrow & & \downarrow \widehat{e^\#} \\ W \otimes W + W & \xrightarrow{\langle e \rangle^\#} & RI \end{array} \quad (4.21)$$

commutes.

Proof. Notice that $\langle e \rangle$ is precisely of the form $e \square f$ from Remark 4.4(vi) for $f = e * W$. Also recall from Remark 4.4(iv) that $\langle e \rangle^\# = (\eta \bullet \langle e \rangle)^\dagger$ and similarly $(e * W)^\# = (\eta \bullet (e * W))^\dagger$. We shall also use that the H -algebra homomorphism $\widehat{e^\#}$ preserves solutions (cf. Remark 4.4(vii)). Thus, we compute

$$\begin{aligned} \widehat{e^\#} \cdot (e * W)^\# &= \widehat{e^\#} \cdot (\eta \bullet (e * W))^\dagger && 4.4(\text{iv}) \\ &= ((\widehat{e^\#} \cdot \eta) \bullet (e * W))^\dagger && 4.4(\text{vii}) \\ &= (e^\# \bullet (e * W))^\dagger && \widehat{e^\#} \text{ extends } e^\# \\ &= ((\eta \cdot e)^\dagger \bullet (e * W))^\dagger && 4.4(\text{iv}) \\ &= (\eta \cdot e \square e * W)^\dagger \cdot \text{inl} && (4.6) \\ &= \left(\eta \bullet (e \square (e * W)) \right)^\dagger \cdot \text{inl} && \text{obvious} \\ &= (e \square (e * W))^\# \cdot \text{inl} && 4.4(\text{iv}) \\ &= \langle e \rangle^\# \cdot \text{inl} && 4.4(\text{vi}) \end{aligned}$$

This completes the proof. \square

Theorem 4.17. *Let H be a finitary, point-strong endofunctor. Then the H -monoid RI above is the initial iterative H -monoid.*

That is, for every H -monoid A there exists precisely one morphism $h: RI \rightarrow A$ which is both a monoid homomorphism and a homomorphism of H -algebras.

Proof. (1) Given an iterative H -monoid

$$\bar{a}: H\bar{A} \rightarrow \bar{A}, \quad \bar{i}: I \rightarrow \bar{A} \quad \text{and} \quad \bar{m}: \bar{A} \otimes \bar{A} \rightarrow \bar{A}$$

we know that there is a unique H -algebra homomorphism

$$h: RI \rightarrow \bar{A} \quad \text{with} \quad h \cdot \eta_I = \bar{i}.$$

It is our task to prove that h preserves multiplication:

$$\begin{array}{ccccc} RI \otimes RI & \xrightarrow{s^R} & RRI & \xrightarrow{\mu_I} & RI \\ h \otimes h \downarrow & & & & \downarrow h \\ \bar{A} \otimes \bar{A} & \xrightarrow{\bar{m}} & & & \bar{A} \end{array} \quad (4.22)$$

(2) Recall $RI = \text{colim } \mathbf{Eq}_I$ from Remark 4.4. We can substitute \mathbf{Eq}_I by the category of all $e: W \rightarrow HW + I$ with W pointed. The argument is as in (a2) of Theorem 4.10. We indicate pointing by writing W_\bullet instead of W (this stands for the notation (W, i^W)). We know that \otimes is finitary, thus

$$RI \otimes RI = \text{colim } \mathbf{Eq}_I \otimes \mathbf{Eq}_I$$

with the colimit cocone

$$e^\# \otimes e^\#: W_\bullet \otimes W_\bullet \rightarrow RI \otimes RI.$$

It is thus sufficient to prove that for every e the square

$$\begin{array}{ccccc} W_\bullet \otimes W_\bullet & \xrightarrow{e^\# \otimes e^\#} & RI \otimes RI & \xrightarrow{s^R} & RRI \xrightarrow{\mu_I} RI \\ (he^\#) \otimes (he^\#) \downarrow & & & & \downarrow h \\ \bar{A} \otimes \bar{A} & \xrightarrow{\bar{m}} & & & \bar{A} \end{array} \quad (4.23)$$

commutes.

For every flat equation morphism $e: W \rightarrow HW + I$ recall $\langle e \rangle$ from Notation 4.15(iii) and put

$$\bar{e} \equiv W \xrightarrow{e} HW + I \xrightarrow{HW + \bar{i}} HW + \bar{A}. \quad (4.24)$$

We prove that (4.23) commutes by verifying that the two sides of the square are both the left-hand part of the solution $\bar{f}^\dagger: W \otimes W + W \rightarrow \bar{A}$ of the equation morphism \bar{f} for

$$f = \langle e \rangle: W \otimes W + W \rightarrow H(W \otimes W + W) + I.$$

(3) Proof of the upper passage of (4.23):

$$h \cdot \mu_I \cdot s^R \cdot (e^\# \otimes e^\#) = \bar{f}^\dagger \cdot \text{inl}: W \otimes W \rightarrow \bar{A}. \quad (4.25)$$

Since $h: RI \rightarrow \bar{A}$ preserves solutions by Remark 4.4(vii), and since $f^\#$ is a solution of $\eta_I \bullet f$ as mentioned in Remark 4.4(iv), the composite $h \cdot f^\#$ is a solution of the equation morphism \bar{f} : indeed, h takes $\eta_I \bullet f$ to \bar{f} due to the diagram

$$\begin{array}{ccc}
W \otimes W + W & \xrightarrow{\eta_I \bullet f} & H(W \otimes W + W) + RI \\
f \searrow & & \downarrow \text{id} + h \\
& H(W \otimes W + W) + I & \\
& \xrightarrow{\text{id} + \eta_I} & \downarrow \text{id} + \bar{I} \\
& \xrightarrow{\bar{f}} & H(W \otimes W + W) + \bar{A}
\end{array}$$

Shortly, the triangle

$$\begin{array}{ccc}
W \otimes W + W & \xrightarrow{f^\#} & RI \\
& \searrow \bar{f}^\dagger & \downarrow h \\
& & \bar{A}
\end{array} \tag{4.26}$$

commutes.

Observe that also the triangle

$$\begin{array}{ccc}
RW & \xrightarrow{\widehat{e}^\#} & RI \\
Re^\# \downarrow & \searrow & \downarrow \mu \\
RRI & \xrightarrow{\mu} & RI
\end{array} \tag{4.27}$$

commutes since RW is the free iterative algebra on $\eta_W: W \rightarrow RW$ and the three morphisms above are homomorphisms of H -algebras which are merged by η_W —indeed:

$$\widehat{e}^\# \cdot \eta_W = e^\#$$

as well as

$$\mu \cdot Re^\# \cdot \eta_W = \mu \cdot \eta_{RI} \cdot e^\# = e^\#.$$

Let us verify that

$$s_{I,W}^R \cdot (e^\# \otimes W) = (e * W)^\#. \tag{4.28}$$

Indeed, for $e: W \rightarrow HW + I$ and $f = \eta_W$ form \hat{e} as in (4.15) and observe that $\hat{e} = \eta_W \bullet (e * W)$ holds. Now recall from Equation (4.19) that $s_{I,W}^R = \eta_W^\flat$. Thus, we have

$$\begin{aligned}
s_{I,W}^R \cdot (e^\# \otimes W) &= \eta_W^\flat \cdot (e^\# \otimes W) \\
&= \hat{e}^\dagger && \text{by (4.16)} \\
&= (\eta_W \bullet (e * W))^\dagger \\
&= (e * W)^\# && \text{by Remark 4.4(iv).}
\end{aligned}$$

We are now in the position to demonstrate (4.25):

$$\begin{array}{ccccc}
W_\bullet \otimes W_\bullet & \xrightarrow{\text{inl}} & (W_\bullet \otimes W_\bullet) + W_\bullet & & \\
\downarrow e^\# \otimes W & \searrow (4.28) & \downarrow (e * W_\bullet)^\# & & \\
RI \otimes W_\bullet & \xrightarrow{s^R} & RW & & \\
\downarrow RI \otimes e^\# & & \downarrow Re^\# & \xrightarrow{(4.21)} & \\
RI \otimes RI & \xrightarrow{s^R} & RRI & \xrightarrow{\widehat{e^\#}} & RI \\
& & \downarrow \mu^I & & \downarrow h \\
& & & & \bar{A}
\end{array}$$

(4) We shall prove for the lower passage of (4.23) that

$$\bar{m} \cdot ((h \cdot \bar{e}^\#) \otimes (h \cdot \bar{e}^\#)) = \bar{f}^\dagger \cdot \text{inl}: W \otimes W \rightarrow \bar{A}.$$

Since the homomorphism $h: RI \rightarrow A$ preserves solutions and, by Remark 4.4(iv), $e^\#$ is the solution of $\eta_I \bullet e$ in RI , it follows that $\bar{e}^\dagger = h \cdot e^\#$ (cf. (4.26)). So we will prove that

$$\bar{f}^\dagger = [\bar{m} \cdot (\bar{e}^\dagger \otimes \bar{e}^\dagger), \bar{e}^\dagger]: W \otimes W + W \rightarrow \bar{A}.$$

To see this it suffices to verify that the following diagram

$$\begin{array}{ccc}
W \otimes W + W & \xrightarrow{[\bar{m} \cdot (\bar{e}^\dagger \otimes \bar{e}^\dagger), \bar{e}^\dagger]} & \bar{A} \\
\downarrow \bar{f} & & \uparrow [\bar{a}, \bar{A}] \\
H(W \otimes W + W) + \bar{A} & \xrightarrow{H[\bar{m} \cdot (\bar{e}^\dagger \otimes \bar{e}^\dagger), \bar{e}^\dagger] + \bar{A}} & H\bar{A} + \bar{A}
\end{array} \tag{4.29}$$

commutes. In order to do so we consider the components of the upper left-hand coproduct separately. For the right-hand component with domain W we obtain

$$\begin{array}{ccccc}
W & \xrightarrow{\bar{e}^\dagger} & \bar{A} & & \\
\downarrow e & \searrow \bar{e} & & & \\
HW + I & \xrightarrow{HW + \bar{i}} & HW + \bar{A} & & \\
\downarrow H \text{inr} + \bar{i} & \nearrow H \text{inr} + \bar{A} & \searrow H\bar{e}^\dagger + \bar{A} & & \\
H(W \otimes W + W) + \bar{A} & \xrightarrow{H[\bar{m} \cdot (\bar{e}^\dagger \otimes \bar{e}^\dagger), \bar{e}^\dagger] + \bar{A}} & H\bar{A} + \bar{A} & & \\
& & & & \uparrow [\bar{a}, \bar{A}]
\end{array}$$

This diagram commutes: the upper right-hand part commutes since \bar{e}^\dagger is a solution of \bar{e} , and all other inner parts are obvious.

For the left-hand component of (4.29) we prove that the following diagram

The diagram consists of several rows of objects and morphisms. The top row has objects $W \otimes W$, $\bar{A} \otimes \bar{A}$, and \bar{A} . Morphisms between them are $e^\dagger \otimes e^\dagger$, m , and $m\cdot([\bar{a}, \bar{A}] \otimes \bar{A})$. The second row has objects $(HW \otimes W) + W$, $(H\bar{A} + \bar{A}) \otimes W$, and $(H\bar{A} \otimes \bar{A}) + (\bar{A} \otimes \bar{A})$. Morphisms are $e \otimes W$, (iii) $[\bar{a}, A] \otimes e^\dagger$, (iv) $\text{id} \otimes e^\dagger$, and $Hm \cdot s^H + m$. The third row has objects $H(W \otimes W) + W$, $H(\bar{A} \otimes \bar{A}) + \bar{A}$, and $H\bar{A} + \bar{A}$. Morphisms are (i) $(s^H + W)$, (v) $Hm + \bar{A}$, and (ii) $[\bar{a}, \bar{A}]$. The fourth row has objects $H(W \otimes W) + HW + \bar{A}$, $H(\bar{A} \otimes \bar{A}) + H\bar{A} + \bar{A}$, and $H\bar{A} + \bar{A}$. Morphisms are $\bar{f} \cdot \text{inl}$, (vi) $H(\bar{e}^\dagger \otimes \bar{e}^\dagger) + \bar{e}^\dagger$, (vii) $Hm + \bar{A}$, and (viii) $H(\bar{e}^\dagger \otimes \bar{e}^\dagger) + \bar{e}^\dagger$. The fifth row has objects $H(W \otimes W + W) + \bar{A}$, $H(\bar{A} \otimes \bar{A} + \bar{A}) + \bar{A}$, and $H\bar{A} + \bar{A}$. Morphisms are $\text{id} + \bar{e}$, (ix) $Hm + [\bar{a}, \bar{A}]$, and (x) $Hm + [\bar{a}, \bar{A}]$. Vertical double arrows indicate commutativity between the rows. Labels (i) through (ix) and (ii) are placed near specific morphisms.

commutes: the left-hand part (i) commutes by the definition of $\bar{f} = \overline{\langle e \rangle}$ and right-hand part (ii) is obvious since \bar{A} is an H -monoid, i.e., $\bar{a} \cdot H\bar{m} \cdot s^H = \bar{m} \cdot (\bar{a} \otimes \bar{A})$. For part (iii) observe that $(HW \otimes W) + W = (HW + I) \otimes W$ and use Diagram (2.3) and Equation (4.24). In part (iv) we use the distributivity for the object in the lower right-hand corner: $(H\bar{A} + \bar{A}) \otimes \bar{A} = H\bar{A} \otimes \bar{A} + \bar{A} \otimes \bar{A}$ —the commutativity is then obvious. We postpone part (v) to the end. Part (vi) commutes by using (2.3). Parts (vii) and (viii) are trivial. We do not claim that part (ix) commutes, but it clearly does when post-composed with $[\bar{a}, \bar{A}]$, which suffices for the commutativity of the outside of the diagram. Finally, it remains to prove that part (v) commutes: we consider the components of the coproduct $(HW \otimes W) + W$ separately. For the right-hand component we obtain the commutative diagram

$$\begin{array}{ccccc}
 W = I \otimes W & \xrightarrow{\bar{i} \otimes W} & \bar{A} \otimes W & \xrightarrow{\bar{A} \otimes \bar{e}^\dagger} & \bar{A} \otimes \bar{A} \\
 \parallel & \searrow I \otimes \bar{e}^\dagger & \nearrow \bar{i} \otimes \bar{A} & \parallel & \downarrow \bar{m} \\
 W & \xrightarrow{\bar{e}^\dagger} & I \otimes \bar{A} & \xrightarrow{\bar{A}} & \bar{A}
 \end{array} \tag{4.30}$$

and for the left-hand one consider the diagram below

$$\begin{array}{ccccc}
 HW \otimes W & \xrightarrow{H\bar{e}^\dagger \otimes \bar{e}^\dagger} & H\bar{A} \otimes \bar{A} & \xrightarrow{s^H} & H(\bar{A} \otimes \bar{A}) \\
 \downarrow s^H & & \nearrow H\bar{m} & & \downarrow H\bar{m} \\
 H(W \otimes W) & \xrightarrow{H(\bar{e}^\dagger \otimes \bar{e}^\dagger)} & H(\bar{A} \otimes \bar{A}) & \xrightarrow{H\bar{m}} & H\bar{A}
 \end{array} \tag{4.31}$$

This completes the proof. \square

Theorem 4.18. *Let H be a finitary, point-strong endofunctor. Then the monoid TI from Corollary 4.14 is the initial completely iterative H -monoid.*

Notice that our proof below uses just the existence of the completely iterative algebras for H (cf. Example 4.7) and not finitariness of H directly.

Proof. Analogously to the preceding proof, for a completely iterative H -monoid $(\bar{A}, \bar{i}, \bar{m}, \bar{a})$ we know that there exists a unique H -algebra homomorphism

$$h: TI \rightarrow \bar{A} \quad \text{with} \quad h \cdot \eta_I^T = \bar{i}$$

and it is our task to prove that it preserves multiplication:

$$\begin{array}{ccccc} TI \otimes TI & \xrightarrow{s^T} & TTI & \xrightarrow{\mu^T} & TI \\ h \otimes h \downarrow & & & & \downarrow h \\ \bar{A} \otimes \bar{A} & \xrightarrow{\bar{m}} & & & \bar{A} \end{array} \quad (4.32)$$

To prove this, we define an equation morphism in \bar{A} :

$$e: TI \otimes TI = HTI \otimes TI + TI \rightarrow H(TI \otimes TI) + \bar{A}$$

such that both passages of (4.32) are solutions of e in \bar{A} : put

$$e = s_{TI, TI}^H + h.$$

For the upper passage of (4.32) we need to verify that the square

$$\begin{array}{ccc} TI \otimes TI & \xrightarrow{h \cdot \mu^T \cdot s^T} & \bar{A} \\ e \downarrow & & \downarrow [\bar{a}, \bar{A}] \\ H(TI \otimes TI) + \bar{A} & \xrightarrow{H(h \cdot \mu^T \cdot s^T) + \bar{A}} & H\bar{A} + \bar{A} \end{array} \quad (4.33)$$

commutes. Consider the components of $HTI \otimes TI + I \otimes TI$ separately. The left-hand component yields

$$\begin{array}{ccccccc} H(TI \otimes TI) & \xrightarrow{Hs^T} & HTTI & \xrightarrow{H\mu^T} & HTI & \xrightarrow{Hh} & H\bar{A} \\ s^H \uparrow & & \tau \downarrow & & \tau \downarrow & & \bar{a} \downarrow \\ HTI \otimes TI & \xrightarrow{\tau \otimes TI} & TTI & \xrightarrow{\mu^T} & TI & \xrightarrow{h} & \bar{A} \\ s^H \left(\begin{array}{c} \nearrow H(TI \otimes TI) \\ \searrow H(TI \otimes TI) \end{array} \right) e & \xrightarrow{s^T} & & & & & \\ \text{inl} \downarrow & & & & & & \uparrow [\bar{a}, \bar{A}] \\ H(TI \otimes TI) + \bar{A} & \xrightarrow{Hs^T + \bar{A}} & HTTI + \bar{A} & \xrightarrow{H\mu^T + \bar{A}} & HTI + \bar{A} & \xrightarrow{Hh + \bar{A}} & H\bar{A} \end{array}$$

The two upper right-hand squares commute since μ^T and h are homomorphisms of algebras, the upper left-hand square is Diagram (4.10) and the lower left-hand part is obvious. Thus, since the outside of the diagram clearly commutes we see that its lower part (4.33) commutes when extended by the left-hand injection $\tau_I \otimes TI$, as desired.

The right-hand component of (4.33) yields

$$\begin{array}{ccccccc}
& I \otimes TI = TI & & & & & \\
& \downarrow \eta^T \otimes TI & \searrow \eta^T T & & & & \\
& TI \otimes TI & \xrightarrow{s^T} & TTI & \xrightarrow{\mu^T} & TI & \xrightarrow{h} \bar{A} \\
& & \searrow e & & & & \\
& \bar{A} & \xrightarrow{\text{inr}} & H(TI \otimes TI) + \bar{A} & \xrightarrow{H(h \cdot \mu^T \cdot s^T) + \bar{A}} & H\bar{A} + \bar{A} & \\
& \swarrow h & & & & \uparrow [\bar{a}, \bar{A}] &
\end{array}$$

Since all other parts clearly commute we see that the right-hand lower part (4.33) commutes when extended by the coproduct injection $\eta_I^T \otimes TI$ as desired. For the lower passage of (4.32) we verify that the square

$$\begin{array}{ccccc}
TI \otimes TI & \xrightarrow{h \otimes h} & \bar{A} \otimes \bar{A} & \xrightarrow{\bar{a}} & \bar{A} \\
\downarrow e & & & & \uparrow [\bar{a}, \bar{A}] \\
H(TI \otimes TI) + \bar{A} & \xrightarrow{H(h \otimes h) + \bar{A}} & H(\bar{A} \otimes \bar{A}) + \bar{A} & \xrightarrow{H\bar{m} + \bar{A}} & H\bar{A} + \bar{A}
\end{array} \tag{4.34}$$

commutes. The left-hand component of $TI \otimes TI = (HTI + I) \otimes TI = (HTI \otimes TI) + TI$ (with coproduct injections $\tau \otimes TI$ and $\eta^T \otimes TI$) yields

$$\begin{array}{ccccccc}
HTI \otimes TI & \xrightarrow{Hh \otimes h} & H\bar{A} \otimes \bar{A} & \xrightarrow{s^H} & H(\bar{A} \otimes \bar{A}) & \xrightarrow{H\bar{m}} & H\bar{A} \\
\downarrow s^H & \searrow \tau \otimes TI & & \searrow \bar{a} \otimes \bar{A} & & & \downarrow \bar{a} \\
H(TI \otimes TI) & & TI \times TI & \xrightarrow{h \otimes h} & \bar{A} \otimes \bar{A} & \xrightarrow{\bar{m}} & \bar{A} \\
& \searrow \text{inl} & \downarrow e & & & & \uparrow [\bar{a}, \bar{A}] \\
& & H(TI \otimes TI) + \bar{A} & \xrightarrow{H(h \otimes h) + \bar{A}} & H(\bar{A} \otimes \bar{A}) + \bar{A} & \xrightarrow{H\bar{m} + \bar{A}} & H\bar{A} + \bar{A}
\end{array}$$

The upper right-hand square is Diagram (3.4) and the middle one commutes since h is an H -algebra homomorphism. The left-hand part is clear since $e = s^H + h$. The lowest passage in the last diagram is

$$\bar{a} \cdot H\bar{m} \cdot H(h \otimes h) \cdot s^H = \bar{a} \cdot H\bar{m} \cdot s^H \cdot (Hh \otimes h)$$

due to the naturality of s^H . Thus, the outside of the diagram commutes, showing that (4.34) commutes when extended by $\tau_I \otimes TI$.

Finally, the right-hand component of (4.34) is obvious:

$$\begin{array}{ccccc}
 I \otimes TI = TI & \xrightarrow{\quad} & & & \\
 h \downarrow & \eta^T \otimes TI & \searrow \bar{i} \otimes h & \searrow h & \\
 \bar{A} & & TI \otimes TI & \xrightarrow{h \otimes h} & \bar{A} \otimes \bar{A} \xrightarrow{\bar{a}} \bar{A} \\
 & \text{inl} \swarrow & e \downarrow & & \uparrow [\bar{a}, \bar{A}] \\
 & & H(TI \otimes TI) + \bar{A} & \xrightarrow{H(h \otimes h) + \bar{A}} & H(\bar{A} \otimes \bar{A}) + \bar{A} \xrightarrow{H\bar{m} + \bar{A}} H\bar{A} + \bar{A}
 \end{array}$$

The lower passage is h and so is the upper one:

$$\bar{a} \cdot (\bar{i} \otimes h) = a \cdot (i \otimes \bar{A}) \cdot (I \otimes h) = I \otimes h.$$

This shows that (4.34) commutes when extended by $\eta_I^T \otimes TI$, which completes the proof. \square

Corollary 4.19. *For every signature Σ the presheaf $T_{\lambda, \Sigma}$ is the initial completely iterative $H_{\lambda, \Sigma}$ -monoid, and the presheaf $R_{\lambda, \Sigma}$ is the initial iterative $H_{\lambda, \Sigma}$ -monoid.* \square

Indeed, $\mathbf{Set}^{\mathcal{F}}$ is monoidally locally finitely presentable category and $H_{\lambda, \Sigma}$ is finitary and point-strong by Corollary 3.5. Moreover, $\mathbb{R}_{\lambda, \Sigma}(V) = R_{\lambda, \Sigma}$ by Theorem 2.19, and $\mathbb{T}_{\lambda, \Sigma}(V) = T_{\lambda, \Sigma}$ by Theorem 4.18.

5. HIGHER-ORDER RECURSION SCHEMES

We can reformulate (and slightly extend) higher-order recursion schemes (1.1) categorically. Throughout this section we put $\mathcal{W} = \mathbf{Set}^{\mathcal{F}}$.

Definition 5.1. A *higher-order recursion scheme* on a signature Σ (of ‘‘terminals’’) is a presheaf morphism

$$e: X \rightarrow F_{\lambda, \Sigma} \otimes (X + V) \tag{5.1}$$

where X is a finitely presentable presheaf.

Remark 5.2.

- (i) The presheaf $F_{\lambda, \Sigma} \otimes (X + V)$ assigns to a context Γ the set $F_{\lambda, \Sigma}(X(\Gamma) + \Gamma)$ of finite λ -terms in contexts $\overline{\Gamma} \subseteq X(\Gamma) + \Gamma$.
- (ii) Although $F_{\lambda, \Sigma}$ is the free $H_{\lambda, \Sigma}$ -algebra on V , see Theorem 2.7, it is not in general true that $F_{\lambda, \Sigma} \otimes Z$ is the free $H_{\lambda, \Sigma}$ -algebra on a presheaf Z . For example, if $Z = V \times V$, then terms in $(F_{\lambda, \Sigma} \otimes Z)(\Gamma)$ are precisely the finite λ - Σ -terms whose free variables are substituted by pairs in $\Gamma \times \Gamma$. In contrast, the free $H_{\lambda, \Sigma}$ -algebra on $V \times V$ contains in context Γ also terms such as $\lambda x.(x, y)$ for $y \in \Gamma$, that is, in variable pairs one member can be bound and one free.
- (iii) In the introduction we considered, for a given context

$$\Gamma_{nt} = \{p_1, \dots, p_n\}$$

of ‘‘nonterminals’’, a system of equations $p_i = f_i$, where f_i is a λ - Σ -term in some context $\Gamma_0 = \{x_1, \dots, x_k\}$. Let X be the free presheaf in n generators p_1, \dots, p_n of

context Γ_0 (a coproduct of n copies of $\mathcal{F}(\Gamma_0, -)$, see Example 2.2(ii)). Then the system of equations defines the unique morphism

$$e: X \rightarrow F_{\lambda, \Sigma} \otimes (X + V)$$

assigning to every p_i the right-hand side f_i lying in

$$F_{\lambda, \Sigma}(\Gamma_{nt} + \Gamma_0) \subseteq F_{\lambda, \Sigma}(X(\Gamma_0) + \Gamma_0).$$

Here we consider $F_{\lambda, \Sigma}$ as an object of $\text{Fin}(\mathbf{Set}, \mathbf{Set})$.

- (iv) Conversely, every morphism (5.1) yields a system of equations $p_i = f_i$ as follows: let Γ_0 fulfill (2.6) in Definition 2.14, and define $\Gamma_{nt} = X(\Gamma_0)$. The element $f_p = e_{\Gamma_0}(p)$ lies, for every nonterminal $p \in \Gamma_{nt}$, in $F_{\lambda, \Sigma}(\Gamma_{nt} + \Gamma_0)$. We obtain a system of equations $p = f_p$ describing the given morphism e .
- (v) We will use the presheaf $R_{\lambda, \Sigma}$ for our uninterpreted solutions of recursion schemes:
A solution of the system of (formal) equations $p_i = f_i$ are rational λ - Σ -terms $p_1^\dagger, \dots, p_n^\dagger$ making those equations identities in $R_{\lambda, \Sigma}(\Gamma_0)$ when we substitute in f_i the λ - Σ -terms p_j^\dagger for the nonterminals p_j ($j = 1, \dots, n$). This is expressed by the Definition 5.3 below.
- (vi) The general case of “equation morphisms” as considered in [2] is (for the endofunctor $H_{\lambda, \Sigma}$) a morphism of type $e: X \rightarrow \mathbb{R}_{\lambda, \Sigma}(X + V)$. We see that every higher-order recursion scheme gives an equation morphism via the inclusion $F_{\lambda, \Sigma} \hookrightarrow R_{\lambda, \Sigma}$ and the strength of the monad $\mathbb{R}_{\lambda, \Sigma}$ (but not necessarily conversely). Our solution theorem below is an application of the general result of [2].

Definition 5.3. A *solution* of a higher-order recursion scheme $e: X \rightarrow F_{\lambda, \Sigma} \otimes (X + V)$ is a morphism $e^\dagger: X \rightarrow R_{\lambda, \Sigma}$ such that the square below, where $j: F_{\lambda, \Sigma} \rightarrow R_{\lambda, \Sigma}$ denotes the embedding, commutes:

$$\begin{array}{ccc} X & \xrightarrow{e^\dagger} & R_{\lambda, \Sigma} \\ e \downarrow & & \uparrow m \\ F_{\lambda, \Sigma} \otimes (X + V) & & \\ j \otimes (X + V) \downarrow & & \uparrow \\ R_{\lambda, \Sigma} \otimes (X + V) & \xrightarrow[R_{\lambda, \Sigma} \otimes [e^\dagger, i]]{} & R_{\lambda, \Sigma} \otimes R_{\lambda, \Sigma} \end{array}$$

Example 5.4. The equation for the fixed-point combinator (see Example 1.1) with $\Sigma = \emptyset$ defines e whose domain is the terminal presheaf 1, that is, $e: 1 \rightarrow F_\lambda \otimes (1 + V)$. The solution $e^\dagger: 1 \rightarrow R_\lambda$ assigns to the unique element of 1 the tree (1.2).

Remark 5.5. Recursion schemes such as $p_1 = p_1$ make no sense—and they certainly fail to have a unique solution. In general, we want to avoid right-hand sides of the form p_i . A recursion scheme is called *guarded* if no right-hand side lies in Γ_{nt} . (Theorem 5.7 below shows that no other restrictions are needed.) Guardedness can be formalized as follows: since

$$R_{\lambda, \Sigma} = H_{\lambda, \Sigma}(R_{\lambda, \Sigma}) + V \quad \text{with injections } \varrho_V \text{ and } i$$

by Remark 4.12, we have (see Remark 3.2(i))

$$R_{\lambda, \Sigma} \otimes (X + V) \cong H_{\lambda, \Sigma}(R_{\lambda, \Sigma}) \otimes (X + V) + X + V$$

with coproduct injections $\varrho_V \otimes \text{id}_{X+V}$ and $i \otimes \text{id}_{X+V}$. Then e is guarded if its extension $(j \otimes (X+V)) \cdot e: X \rightarrow R_{\lambda,\Sigma} \otimes (X+V)$ factorizes through the embedding of the first and third summand of this coproduct:

Definition 5.6. A higher-order recursion scheme $e: X \rightarrow F_{\lambda,\Sigma} \otimes (X+V)$ is called *guarded* if $(j \otimes (X+V)) \cdot e$ factorizes through

$$[\varrho \otimes \text{id}, (i \otimes \text{id}) \cdot \text{inr}]: H_{\lambda}(R_{\lambda,\Sigma}) \otimes (X+V) + V \rightarrow R_{\lambda,\Sigma} \otimes (X+V).$$

Theorem 5.7. Every guarded higher-order recursion scheme has a unique solution. \square

Remark. In Definition 5.1 we restricted higher-order recursion schemes to have $F_{\lambda,\Sigma}$ in their codomain. This corresponds well to the classical notion of recursion schemes as explained in Remark 5.2. Moreover, this leads to a simple presentation of the interpreted semantics in Section 6 below. However, Theorem 5.7 remains valid if we replace $F_{\lambda,\Sigma}$ by $R_{\lambda,\Sigma}$ in Definition 5.1 and define solution by $e^{\dagger} = m \cdot R_{\lambda,\Sigma} \otimes [e^{\dagger}, i] \cdot e$. This extends the notion of a higher-order recursion scheme (1.1) to allow the right-hand sides f_i to be rational λ - Σ -terms. We shall prove Theorem 5.7 working with higher-order schemes of the form $e: X \rightarrow R_{\lambda,\Sigma} \otimes (X+V)$, X finitely presentable. We call e guarded if it factorizes through $[\varrho \otimes \text{id}, (i \otimes \text{id}) \cdot \text{inr}]$.

Proof. Let us apply the monad $\mathbb{R}_{\lambda,\Sigma}$ and its point-strength s^R from Theorem 4.10. We construct for every higher-order recursion scheme $e: X \rightarrow \mathbb{R}_{\lambda,\Sigma}(V) \otimes (X+V)$ a rational equation morphism $\bar{e}: X \rightarrow \mathbb{R}_{\lambda,\Sigma}(X+V)$ in the sense of [2] as follows:

$$\bar{e} \equiv X \xrightarrow{e} \mathbb{R}_{\lambda,\Sigma}(V) \otimes (X+V) \xrightarrow{s^R_{(V,\text{id})(X+V,\text{inr})}} \mathbb{R}_{\lambda,\Sigma}(X+V).$$

From the guardedness of e we conclude that \bar{e} is guarded in the sense of [2], that is, \bar{e} factorizes through the summand $H_{\lambda,\Sigma}\mathbb{R}_{\lambda,\Sigma}(X+V) + V$ of $\mathbb{R}_{\lambda,\Sigma}(X+V)$, see Remark 4.12. Indeed, this follows from the following diagram

$$\begin{array}{ccc} H_{\lambda,\Sigma}(R_{\lambda,\Sigma}) \otimes (X+V) + V & \xrightarrow{H_{\lambda,\Sigma}s^R + \text{id}} & H_{\lambda,\Sigma}(\mathbb{R}_{\lambda,\Sigma}(X+V)) + V \\ \nearrow \text{dashed} \quad \downarrow & & \downarrow \\ X & \xrightarrow[e]{s^R} & \mathbb{R}_{\lambda,\Sigma}(X+V) \end{array}$$

Consequently, by Theorem 4.5 in [2] there exists a unique solution \bar{e}^{\dagger} of \bar{e} with respect to the monad $\mathbb{R}_{\lambda,\Sigma}$ —this means that there exists a unique morphism $\bar{e}^{\dagger}: X \rightarrow \mathbb{R}_{\lambda,\Sigma}(V)$ such that the outside of the diagram below commutes:

$$\begin{array}{ccccc} X & \xrightarrow{\bar{e}^{\dagger}} & \mathbb{R}_{\lambda,\Sigma}(V) & & \\ \downarrow e & & \swarrow m & & \uparrow \mu_V \\ \mathbb{R}_{\lambda,\Sigma}(V) \otimes (X+V) & \xrightarrow{\mathbb{R}_{\lambda,\Sigma}(\text{id}_V) \otimes [\bar{e}^{\dagger}, i]} & \mathbb{R}_{\lambda,\Sigma}(V) \otimes \mathbb{R}_{\lambda,\Sigma}(V) & \xrightarrow{s^R_{(V,\text{id})(R_{\lambda,\Sigma},i)}} & \mathbb{R}_{\lambda,\Sigma}(V)(\mathbb{R}_{\lambda,\Sigma}(V)) \\ \downarrow s^R_{(V,\text{id})(X+V,\text{inr})} & & \searrow s^R_{(V,\text{id})(R_{\lambda,\Sigma},i)} & & \\ \mathbb{R}_{\lambda,\Sigma}(X+V) & \xrightarrow{\mathbb{R}_{\lambda,\Sigma}[\bar{e}^{\dagger}, i]} & & & \end{array}$$

Since the right-hand triangle is Equation (4.20) and the lower square commutes by the naturality of s^R , we see that \bar{e}^\dagger is a solution of e in the sense of Definition 5.3 iff \bar{e}^\dagger is a solution of \bar{e} in the sense of [2]. This proves that e has a unique solution. \square

Remark 5.8. Analogously we could define a solution of a higher-order recursion scheme of the form $e: X \rightarrow T_{\lambda,\Sigma} \otimes (X + V)$ in the initial completely iterative monoid $T_{\lambda,\Sigma}$, see Theorem 3.11. (Here we, moreover, do not need to assume that X is finitely presentable.) And guardedness means here that e factorizes through the summand $H_{\lambda,\Sigma}T_{\lambda,\Sigma} \otimes (X + V) + V$ of $T_{\lambda,\Sigma} \otimes (X + V)$ (cf. Remarks 4.9 and 3.2(i)). Every such guarded scheme has a unique solution in $T_{\lambda,\Sigma}$. The proof is completely analogous to the previous one for $R_{\lambda,\Sigma}$ but using Corollary 3.8 in [1] in lieu of Theorem 4.5 from [2].

Remark 5.9. Notice that the definitions and results in this section generalize to the setting as considered in Section 4 (see Assumption 4.1). Simply replace $H_{\lambda,\Sigma}$ by the finitary functor H , the monoid $F_{\lambda,\Sigma}$ by the initial H -monoid F ; this exists and is given by the free H -algebra on V , see [10]. Further replace the monoid $R_{\lambda,\Sigma}$ by the initial iterative H -monoid RI (cf. Theorem 4.17), and $T_{\lambda,\Sigma}$ by the initial completely iterative H -monoid (cf. Theorem 4.18).

6. INTERPRETED SOLUTIONS

In the present section we prove that every Scott model of λ -calculus as a CPO, D , with fold and unfold operations can be used as a model of higher-order recursion. Following M. Fiore et al [10] we work with the presheaf $\langle D, D \rangle$ which to a context Γ assigns the set of all continuous functions from D^Γ to D . We prove that every higher-order recursion scheme has a least solution in $\langle D, D \rangle$.

We denote by **CPO** the cartesian closed category of posets with directed joins and continuous functions. Thus least elements are not assumed; if they exist we use \perp for them.

Assumption 6.1. *We assume that a Scott model D of λ -calculus is given, i.e., a CPO with \perp and with an embedding-projection pair*

$$\text{fold} : \mathbf{CPO}(D, D) \triangleleft D : \text{unfold}. \quad (6.1)$$

Moreover, for the given signature Σ of terminals we also assume that continuous operations

$$\sigma^D : D^n \rightarrow D \quad \text{for every } n\text{-ary } \sigma \text{ in } \Sigma$$

are given.

Notation 6.2. We define a presheaf $\langle D, D \rangle$ by

$$\langle D, D \rangle \Gamma = \mathbf{CPO}(D^\Gamma, D).$$

Remark 6.3.

- (a) Observe that elements of $\langle D, D \rangle$ can always be interpreted in D : the above function $\text{fold}: \langle D, D \rangle 1 \rightarrow D$ yields obvious functions $\text{fold}_\Gamma: \langle D, D \rangle \Gamma \rightarrow D$ for all contexts Γ via induction: define $\text{fold}_{\Gamma+1}$ by

$$\begin{array}{ccc} \langle D, D \rangle (\Gamma + 1) & \cong & \mathbf{CPO}(D^\Gamma \times D, D) \xrightarrow{\text{fold}_{\Gamma+1}} D \\ & & \downarrow \text{curry} \\ & & \mathbf{CPO}(D^\Gamma, D^D) \xrightarrow{\mathbf{CPO}(D^\Gamma, \text{fold})} \mathbf{CPO}(D^\Gamma, D) \end{array}$$

where $\text{curry}: \mathbf{CPO}(D^\Gamma \times D, D) \rightarrow \mathbf{CPO}(D^\Gamma, D^D)$ is the currfication.

- (b) The projections $D^\Gamma \rightarrow D$ are continuous functions, that is, elements of $\langle D, D \rangle \Gamma$. This defines a natural pointing

$$\iota: V \rightarrow \langle D, D \rangle$$

of the presheaf $\langle D, D \rangle$.

Remark 6.4. The presheaf $\langle D, D \rangle$ is an $H_{\lambda, \Sigma}$ -monoid. Indeed, application and abstraction are naturally obtained from (6.1), see [10]. The monoid structure

$$m: \langle D, D \rangle \otimes \langle D, D \rangle \rightarrow \langle D, D \rangle$$

can be described directly by using the coend formula (3.2) or indirectly:

- (a) For the direct description consider the component of m_Γ corresponding, for an element $f \in \mathbf{Set}(\bar{\Gamma}, \mathbf{CPO}(D^\Gamma, D))$, to the injection

$$\text{in}_f: \mathbf{CPO}(D^{\bar{\Gamma}}, D) \rightarrow \int^{\bar{\Gamma}} \mathbf{Set}(\bar{\Gamma}, \mathbf{CPO}(D^\Gamma, D)) \bullet \mathbf{CPO}(D^{\bar{\Gamma}}, D).$$

Observe that f yields a continuous function $\tilde{f}: D^\Gamma \rightarrow D^{\bar{\Gamma}}$ defined by $\tilde{f}(x) = f(-)(x)$ for all $x \in \Gamma$. We define the component $m_\Gamma \cdot \text{in}_f$ of m_Γ by

$$\begin{array}{c} \langle D, D \rangle \otimes \langle D, D \rangle (\Gamma) = \int^{\bar{\Gamma}} \mathbf{Set}(\bar{\Gamma}, \mathbf{CPO}(D^\Gamma, D)) \bullet \mathbf{CPO}(D^{\bar{\Gamma}}, D) \xrightarrow{m_\Gamma} \mathbf{CPO}(D^\Gamma, D) = \langle D, D \rangle \Gamma \\ \uparrow \text{in}_f \qquad \qquad \qquad \swarrow g \mapsto g \cdot \tilde{f} \\ \mathbf{CPO}(D^{\bar{\Gamma}}, D) \end{array} \quad (6.2)$$

- (b) There is a much more elegant way of obtaining the monoid structure of $\langle D, D \rangle$. From results of Steve Lack [14] we see that the monoidal category $(\mathbf{Set}^{\mathcal{F}}, \otimes, V)$ has the following monoidal action $*$ on \mathbf{CPO} : given X in $\mathbf{Set}^{\mathcal{F}}$ and C in \mathbf{CPO} , we put $X * C = \int^\Gamma X(\Gamma) \bullet C^\Gamma$. Moreover, extending the above notation to pairs C, C' of CPO's and defining $\langle C, C' \rangle \Gamma = \mathbf{CPO}(C^\Gamma, C')$ we obtain a presheaf with a natural isomorphism

$$\mathbf{Set}^{\mathcal{F}}(X, \langle C, C' \rangle) \cong \mathbf{CPO}(X * C, C').$$

As observed by George Janelidze and Max Kelly [13] this yields an enriched category whose hom-objects are $\langle C, C' \rangle$. In particular, $\langle D, D \rangle$ receives a monoid structure. It is tedious but not difficult to prove that this monoid structure is given by (6.2) above and it forms an $H_{\lambda, \Sigma}$ -monoid (cf. Definition 3.6).

Notation 6.5. We denote by

$$\llbracket - \rrbracket: F_{\lambda, \Sigma} \rightarrow \langle D, D \rangle$$

the unique $H_{\lambda, \Sigma}$ -monoid homomorphism (see Theorem 3.9). For every finite term t in context Γ we thus obtain its interpretation as a continuous function $\llbracket t \rrbracket_\Gamma: D^\Gamma \rightarrow D$

Remark 6.6. What is our intuition of an interpreted solution of higher order recursion scheme $e: X \rightarrow F_{\lambda, \Sigma} \otimes (X + V)$ in the presheaf $\langle D, D \rangle$? This should be an interpretation of X -terms in $\langle D, D \rangle$ via a natural transformation

$$e^\dagger: X \rightarrow \langle D, D \rangle$$

with the following property: Given an X -term x in context Γ , then e_Γ assigns to it an element $e_\Gamma(x)$ of $(F_{\lambda,\Sigma} \otimes (X + V))(\Gamma)$, that is, a finite term $t \in F_{\lambda,\Sigma}(\bar{\Gamma})$ for some $\bar{\Gamma} \subseteq X(\Gamma) + \Gamma$. We request that the solution assigns to x the same value $e_\Gamma^\dagger(x) : D^\Gamma \rightarrow D$ that we obtain from the interpretation $\llbracket t \rrbracket_1$ of the given term by substituting the $\bar{\Gamma}$ -variables using $[e^\dagger, \iota] : X + V \rightarrow \langle D, D \rangle$. This substitution is given by composing $\llbracket - \rrbracket \otimes [e^\dagger, \iota]$ with the monoid structure of $\langle D, D \rangle$. This leads to the following

Definition 6.7. Given a higher-order recursion scheme $e : X \rightarrow F_{\lambda,\Sigma} \otimes (X + V)$ by an *interpreted solution* is meant a presheaf morphism $e^\dagger : X \rightarrow \langle D, D \rangle$ such that the square below commutes:

$$\begin{array}{ccc} X & \xrightarrow{e^\dagger} & \langle D, D \rangle \\ e \downarrow & & \uparrow m \\ F_{\lambda,\Sigma} \otimes (X + V) & \xrightarrow{\llbracket - \rrbracket \otimes [e^\dagger, \iota]} & \langle D, D \rangle \otimes \langle D, D \rangle \end{array} \quad (6.3)$$

Theorem 6.8. Every higher-order recursion scheme has a least interpreted solution in $\langle D, D \rangle$ in the pointwise ordering of $\mathbf{Set}^{\mathcal{F}}(X, \langle D, D \rangle)$.

Proof. Observe that $\mathbf{Set}^{\mathcal{F}}(X, \langle D, D \rangle)$ is a CPO with \perp . Therefore it is sufficient to prove that the endomap of $\mathbf{Set}^{\mathcal{F}}(X, \langle D, D \rangle)$ given by

$$s \mapsto m \cdot (\llbracket - \rrbracket \otimes [s, \iota]) \cdot e \quad (6.4)$$

is continuous, then we can use the Kleene Fixed-Point Theorem. Observe that the function (6.4) is a composite

$$\begin{array}{ccc} \mathbf{Set}^{\mathcal{F}}(X, \langle D, D \rangle) & \xrightarrow{s \mapsto m \cdot (\llbracket - \rrbracket \otimes [s, \iota]) \cdot e} & \mathbf{Set}^{\mathcal{F}}(X, \langle D, D \rangle) \\ \downarrow [-, \iota] & & \uparrow - \cdot e \\ \mathbf{Set}^{\mathcal{F}}(X + V, \langle D, D \rangle) & \xrightarrow{z \mapsto m \cdot (\text{id}_{F_{\lambda,\Sigma}} \otimes z)} & \mathbf{Set}^{\mathcal{F}}(F_{\lambda,\Sigma} \otimes (X + V), \langle D, D \rangle) \end{array} \quad (6.5)$$

where the vertical arrows are obviously continuous. It remains to prove the continuity of

$$z \mapsto m \cdot (\text{id} \otimes z) \quad \text{for } z : Z \rightarrow \langle D, D \rangle$$

where $Z = X + V$ (but this structure of Z plays no role).

We use the coend formula (3.2). It is obvious what the components of

$$(\text{id} \otimes z)_\Gamma : \int^{\bar{\Gamma}} \mathbf{Set}((\bar{\Gamma}, Z(\Gamma)) \bullet \langle D, D \rangle \bar{\Gamma}) \rightarrow \int^{\bar{\Gamma}} \mathbf{Set}(\bar{\Gamma}, \langle D, D \rangle \Gamma) \bullet \langle D, D \rangle \bar{\Gamma}$$

are: the $\bar{\Gamma}$ -component composed with the coproduct injection of $u \in \mathbf{Set}(\bar{\Gamma}, Z(\Gamma))$ yields the coproduct injection of

$$f = z_\Gamma \cdot u \in \mathbf{Set}(\bar{\Gamma}, \langle D, D \rangle \Gamma).$$

Combined with the component of m_Γ , see (6.2), this yields the components of $(m \cdot (\text{id} \otimes z))_\Gamma$ as follows: for every context $\bar{\Gamma}$ the component composed with the coproduct injection in_u of u is the map

$$g \mapsto g \cdot \widetilde{z_\Gamma \cdot u} \quad \text{for } g \in \langle D, D \rangle \bar{\Gamma}.$$

We are ready to prove that the lower horizontal arrow of (6.5) is continuous. Suppose

$$z = \bigsqcup_{k \in K} z^k$$

is a directed join in the pointwise ordering of $\mathbf{Set}^{\mathcal{F}}(X + V, \langle D, D \rangle)$. In order to prove the equality

$$m \cdot (\text{id} \otimes z) = \bigsqcup_{k \in K} m \cdot (\text{id} \otimes z^k)$$

in $\mathbf{Set}^{\mathcal{F}}$, we choose an arbitrary context Γ and prove that

$$m_{\Gamma} \cdot (\text{id} \otimes z_{\Gamma}) = \bigsqcup_{k \in K} m_{\Gamma} \cdot (\text{id} \otimes z_{\Gamma}^k)$$

holds in \mathbf{Set} . For that use the fact that the injection maps, for all contexts $\bar{\Gamma}$ and all $u \in \mathbf{Set}(\bar{\Gamma}, Z(\Gamma))$,

$$\langle D, D \rangle \bar{\Gamma} \xrightarrow{\text{in}_u} \mathbf{Set}(\bar{\Gamma}, Z(\Gamma)) \bullet \langle D, D \rangle \bar{\Gamma} \xrightarrow{\text{in}_{\bar{\Gamma}}} \int^{\bar{\Gamma}} \mathbf{Set}(\bar{\Gamma}, Z(\Gamma)) \bullet \langle D, D \rangle \bar{\Gamma}$$

form a collectively epimorphic cocone. Thus, it is sufficient to prove that for every context $\bar{\Gamma}$ and every $g \in \langle D, D \rangle \bar{\Gamma}$ we have

$$g \cdot \widetilde{z_{\Gamma} \cdot u} = \bigsqcup_{k \in K} g \cdot \widetilde{z_{\Gamma}^k \cdot u}.$$

Since g is continuous from $D^{\bar{\Gamma}}$ to D , we just need to verify

$$\widetilde{z_{\Gamma} \cdot u} = \bigsqcup_{k \in K} \widetilde{z_{\Gamma}^k \cdot u}.$$

From the pointwise ordering we clearly get $z_{\Gamma} \cdot u = \bigsqcup_{k \in K} z_{\Gamma}^k \cdot u$, thus, we only need to observe the continuity of the map $f \mapsto \tilde{f}$, and this follows from the coordinate-wise ordering of $D^{\bar{\Gamma}}$.

□

7. CONCLUSIONS

We proved that guarded higher-order recursion schemes have a unique uninterpreted solution, i.e., a solution as a rational λ - Σ -term. And they also have the least interpreted solution for interpretations based on Scott's models of λ -calculus as CPO's with continuous operations for all "terminal" symbols of the recursion scheme.

Following M. Fiore *et al* [10] we worked in the category $\mathbf{Set}^{\mathcal{F}}$ of sets in context, that is, covariant presheaves on the category \mathcal{F} of finite sets and functions. A presheaf is a set dependent on a context (a finite set of variables). For every signature Σ of "terminal" operation symbols it was proved in [10] that the presheaf $F_{\lambda, \Sigma}$ of all finite λ - Σ -terms is the initial $H_{\lambda, \Sigma}$ -monoid. This means that $F_{\lambda, \Sigma}$ has (i) the λ -operations (of abstraction and application) together with the operations given by Σ rendering an $H_{\lambda, \Sigma}$ -algebra, (ii) the operation expressing simultaneous substitution rendering a monoid in the category of presheaves, and (iii) these two structures are canonically related. And $F_{\lambda, \Sigma}$ is the initial presheaf with such structure. R. Matthes and T. Uutalu [16] showed that the presheaf $T_{\lambda, \Sigma}$ of finite and infinite λ - Σ -terms is also an $H_{\lambda, \Sigma}$ -monoid. Here we proved that this is the initial completely

iterative $H_{\lambda,\Sigma}$ -monoid. And its subpresheaf $R_{\lambda,\Sigma}$ of all rational λ - Σ -terms is the initial iterative $H_{\lambda,\Sigma}$ -monoid. We used that last presheaf in our uninterpreted semantics of recursion schemes.

Our approach was based on untyped λ -calculus. The ideas in the typed version are quite analogous. If S is the set of all types, then we form the full subcategory \mathcal{F} of \mathbf{Set}^S of finite S -sorted sets and consider presheaves in $(\mathbf{Set}^S)^{\mathcal{F}}$ —the latter category is equivalent to that of finitary endofunctors of the category \mathbf{Set}^S . The definition of $H_{\lambda,\Sigma}$ is then completely analogous to the untyped case, and one can form the presheaves $F_{\lambda,\Sigma}$ (free algebra on V), $T_{\lambda,\Sigma}$ (free completely iterative algebra) and $R_{\lambda,\Sigma}$ (free iterative algebra). Each of them is a monoid, in fact, an $H_{\lambda,\Sigma}$ -monoid in the sense of [10]. Moreover, every guarded higher-order recursion scheme has a unique solution in $R_{\lambda,\Sigma}$. The interpreted semantics can be built up on a CPO-enriched cartesian closed category (as our model of typed λ -calculus) with additional continuous morphisms for all terminals. The details of the typed version are more involved, and we leave them for future work.

Related results on higher-order substitution can be found e.g. in [16] and [20].

In future work we will, analogously as in [18], investigate the relation of uninterpreted and interpreted solutions.

REFERENCES

- [1] Aczel, P., Adámek, J., Milius, S., Velebil, J.: Infinite trees and completely iterative theories: a coalgebraic view. *Theoret. Comput. Sci.* **300** (2003), 1–45.
- [2] Adámek, J., Milius, S., Velebil, J.: Iterative algebras at work. *Math. Structures Comput. Sci.* **16** (2006), 1085–1131.
- [3] Adámek, J., Milius, S., Velebil, J.: Elgot Algebras. *Log. Methods Comput. Sci.* **2**(5:4), 2006, 32 pp.
- [4] Adámek, J., Milius, S., Velebil, J.: Semantics of higher-order recursion schemes. *Proc. Algebra and Coalgebra in Computer Science (CALCO 2009)*, Lect. Notes Comput. Sci. **5728**, Springer 2009, 49–63.
- [5] Adámek, J., Porst, H. E.: On tree coalgebras and coalgebra presentations. *Theoret. Comput. Sci.* **311** (2004), 257–283.
- [6] Adámek, J., Trnková, V.: Automata and algebras in a category. Kluwer Academic Publishers, Dordrecht, 1990.
- [7] Aehlig, K.: A finite semantics of simply-typed lambda terms for infinite runs of automata. *Proc. Computer Science Logic 2006*, Lect. Notes Comput. Sci. **4207**, Springer 2006, 104–118.
- [8] Damm, W.: Higher-order program schemes and their languages. *Lect. Notes Comput. Sci.* **48**, Springer 1979, 51–72.
- [9] Fiore M.: Second order dependently sorted abstract syntax. *Proc. Logic in Computer Science 2008*, IEEE Press 2008, 57–68.
- [10] Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. *Proc. Logic in Computer Science 1999*, IEEE Press 1999, 193–202.
- [11] Garland, S. J., Luckham, D. C.: Program schemes, recursion schemes and formal languages. *J. Comput. Syst. Sci.* **7** (1973), 119–160.
- [12] Guessarian, I.: Algebraic semantics. *Lect. Notes Comput. Sci.* **99**, Springer, 1981.
- [13] Janelidze, G., Kelly, G. M.: A note on actions of a monoidal category. *Theory Appl. Categ.* **9** (2001), 61–91
- [14] Lack, S.: On the monadicity of finitary monads. *J. Pure Appl. Algebra* **140** (1999), 65–73.
- [15] Makkai, M., Paré R.: Accessible Categories: The Foundations of Categorical Model Theory. Contemporary Mathematics Vol. **104**, American Mathematical Society, 1989.
- [16] Matthes, R., Uustalu, T.: Substitution in non-wellfounded syntax with variable binding. *Theoret. Comput. Sci.* **327** (2004), 155–174.
- [17] Milius, S.: Completely iterative algebras and completely iterative monads. *Inform. and Comput.* **196** (2005), 1–41.

- [18] Milius, S., Moss, L.: The category theoretic solution of recursive program schemes. *Theoret. Comput. Sci.* **366** (2006), 3–59, corrigendum in **403** (2008), 409–415.
- [19] Miranda, G.: Structures generated by higher-order grammars and the safety constraint. Ph.D. Thesis, Merton College, Oxford, 2006.
- [20] Power, J.: A unified category theoretical approach to variable binding. Proc. MERLIN 2003.

A PRACTICAL TYPE THEORY FOR SYMMETRIC MONOIDAL CATEGORIES

MICHAEL SHULMAN

ABSTRACT. We give a natural-deduction-style type theory for symmetric monoidal categories whose judgmental structure directly represents morphisms with tensor products in their codomain as well as their domain. The syntax is inspired by Sweedler notation for coalgebras, with variables associated to types in the domain and terms associated to types in the codomain, allowing types to be treated informally like “sets with elements” subject to global linearity-like restrictions. We illustrate the usefulness of this type theory with various applications to duality, traces, Frobenius monoids, and (weak) Hopf monoids.

Contents

1	Introduction	1
2	Props	9
3	On the admissibility of structural rules	11
4	The type theory for free props	14
5	Constructing free props from type theory	24
6	Presentations of props	29
7	Examples	31

1. Introduction

1.1. TYPE THEORIES FOR MONOIDAL CATEGORIES. Type theories are a powerful tool for reasoning about categorical structures. This is best-known in the case of the internal language of a topos, which is a higher-order intuitionistic logic. But there are also weaker type theories that correspond to less highly-structured categories, such as regular logic for regular categories, simply typed λ -calculus for cartesian closed categories, typed algebraic theories for categories with finite products, and so on (a good overview can be found in [Joh02, Part D]).

This material is based on research sponsored by The United States Air Force Research Laboratory under agreement number FA9550-15-1-0053. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Air Force Research Laboratory, the U.S. Government, or Carnegie Mellon University.

2020 Mathematics Subject Classification: 18M05, 18M85, 03B47.

Key words and phrases: type theory, symmetric monoidal category, prop, coalgebra.

© Michael Shulman, . Permission to copy for private use granted.

However, type theories seem to be only rarely used to reason about *non-cartesian monoidal* categories. Such categories are of course highly important in both pure mathematics and its applications (such as quantum mechanics, network theory, etc.), but usually they are studied using either traditional arrow-theoretic syntax or string diagram calculus [JS91, Sel11].

Type theories corresponding to non-cartesian monoidal categories do certainly exist — intuitionistic linear logic for closed symmetric monoidal categories, ordered logic for non-symmetric monoidal categories, classical linear logic for $*$ -autonomous categories — but they are not widely used for reasoning about monoidal categories. I believe this is largely because their convenience for practical category-theoretic arguments does not approach that of cartesian type theories. The basic problem is that most nontrivial arguments in monoidal category theory involve tensor product objects in both the *domain* and the *codomain* of morphisms.

Existing type theories for non-cartesian monoidal categories fall into two groups, neither of which deals satisfactorily with this issue. The first is exemplified by intuitionistic linear logic, whose judgments have many variables as inputs but only a single term as output:

$$x : A, y : B, z : C \vdash f(x, g(y, z)) : D.$$

This allows the terms (such as $f(x, g(y, z))$) to intuitively “treat types as if they were sets with elements” (one of the big advantages of type-theoretic syntax). But it privileges tensor products in the domain (here the domain is semantically $A \otimes B \otimes C$) over the codomain. The only way to talk about morphisms into tensor products is to use the tensor product type constructor:

$$x : A, y : B \vdash (y, x) : B \otimes A.$$

This is asymmetric, but more importantly its term syntax is heavy and unintuitive: in the cartesian case we can use an element $p : A \times B$ by simply projecting out its components $\pi_1(p) : A$ and $\pi_2(p) : B$, but in the non-cartesian case we have to “match” it and bind both components as new variables:

$$p : A \otimes B \vdash \text{let } (x, y) := p \text{ in } (y, x) : B \otimes A.$$

The second group of non-cartesian type theories is exemplified by classical linear logic, whose judgments have multiple inputs as well as multiple outputs:

$$A, B, C \vdash D, E.$$

This eliminates the asymmetry, but at the expense of no longer having a concise and intuitive term syntax. Most commonly such type theories are presented as sequent calculi without any terms at all. Term calculi do exist (e.g. [Red93, BCST96, Ong96, CPT16]) but usually involve some kind of “covariables”, which complicate the interpretation of types as “sets with elements”. Moreover, in type theories of this sort the tensor product appearing in codomains is usually a *different* one from the one appearing in domains

(the “cotensor product” of a $*$ -autonomous category), whereas in practical applications it happens very frequently that they are the same. The only type theories with the same tensor product appearing in domains and codomains that I know of in the literature are [Shi99, Dun06], which are sequent calculi without a real term syntax, and [Has97], which does have a term syntax but still decomposes tensor products by matching. (The referee has pointed out that [PLC17] is in a similar spirit to our work, though not directly comparable.)

The purpose of this paper is to fill this gap, by describing a type theory for symmetric monoidal categories in which:

- Judgments have multiple inputs as well as multiple outputs.
- The tensor products appearing in domains and codomains are the same.
- There is a convenient and intuitive term syntax without the need for “covariables”.
- Tensor product types, though rarely needed, have a convenient term syntax involving “projections”, as in the cartesian case, rather than “matches”.

This type theory seems to be very convenient in practice for reasoning about symmetric monoidal categories. We will show this in section 7 with a number of examples; here we sketch two of them to whet the reader’s appetite.

For the first, recall that in a compact closed category, the **trace** $\text{tr}(f)$ of an endomorphism $f : A \rightarrow A$ is the composite

$$I \xrightarrow{\eta} A \otimes A^* \xrightarrow{f \otimes \text{id}} A \otimes A^* \xrightarrow{\sim} A^* \otimes A \xrightarrow{\varepsilon} I$$

where I is the unit object, A^* is the dual of A , and η and ε are the unit and counit of the duality (also called the coevaluation and evaluation, respectively). A fundamental property of trace is *cyclicity*, i.e. $\text{tr}(fg) = \text{tr}(gf)$ for any $f : A \rightarrow B$ and $g : B \rightarrow A$. The proof of this is essentially incomprehensible when written with composites of morphisms:

$$\begin{aligned} \text{tr}(fg) &= \varepsilon \circ (fg \otimes \text{id}) \eta = \varepsilon \circ (f \otimes \text{id})(g \otimes \text{id}) \eta = \varepsilon \circ (f \otimes \text{id})(\text{id} \otimes \varepsilon \otimes \text{id})(\eta \otimes \text{id} \otimes \text{id})(g \otimes \text{id}) \eta = \\ &= \varepsilon \circ (\text{id} \otimes \varepsilon \otimes \text{id})(f \otimes \text{id} \otimes \text{id} \otimes \text{id})(\text{id} \otimes \text{id} \otimes g \otimes \text{id})(\eta \otimes \eta) = (\varepsilon \otimes \varepsilon) \circ (\text{id} \otimes \text{id} \otimes g \otimes \text{id})(f \otimes \text{id} \otimes \text{id} \otimes \text{id})(\eta \otimes \eta) = \\ &= (\varepsilon \otimes \varepsilon) \circ (\text{id} \otimes \text{id} \otimes g \otimes \text{id})(\text{id} \otimes \text{id} \otimes \eta)(f \otimes \text{id}) \eta = \varepsilon \circ (\text{id} \otimes \varepsilon \otimes \text{id})(g \otimes \text{id} \otimes \text{id} \otimes \text{id})(\eta \otimes \text{id} \otimes \text{id})(f \otimes \text{id}) \eta = \\ &= \varepsilon \circ (g \otimes \text{id})(\text{id} \otimes \varepsilon \otimes \text{id})(\eta \otimes \text{id} \otimes \text{id})(f \otimes \text{id}) \eta = \varepsilon \circ (g \otimes \text{id})(f \otimes \text{id}) \eta = \varepsilon \circ (gf \otimes \text{id}) \eta = \text{tr}(gf). \end{aligned}$$

It becomes much more understandable when written in string diagram calculus, as in Figure 1. But in our type theory, the proof is one line of algebra:

$$\text{tr}(fg) \stackrel{\text{def}}{=} (| \lambda^B y \triangleleft fgy) = (| \lambda^B y \triangleleft fx, \lambda^A x \triangleleft gy) = (| \lambda^A x \triangleleft gfx) \stackrel{\text{def}}{=} \text{tr}(gf).$$

The reader is not expected to understand this proof yet, but to give some flavor we explain that the two non-definitional equalities are “ β -reductions for duality”. The binary operator \triangleleft denotes the counit ε , while a variable/abstraction pair $(x, \lambda^A x)$ denotes the

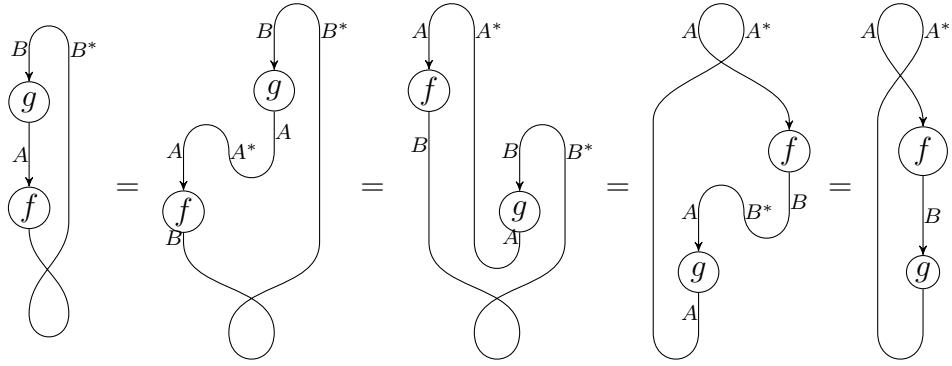


Figure 1: Cyclicity of trace with string diagrams

unit η . A term of the form $\lambda^A x \triangleleft t$, where x does not occur in t , is a β -redex for duality, and in the resulting reduction this term is eliminated and the occurrence of x elsewhere¹ is substituted by t . (Semantically, this means applying a zigzag identity, which in string diagram notation is “straightening a bent string”.) Thus the middle term ($| \lambda^B y \triangleleft fx, \lambda^A x \triangleleft gy$) contains two β -redexes, and the resulting two reductions yield the two terms on either side.

For our second example, recall that if x is an element of a monoid M , and \bar{x} and \hat{x} are both two-sided inverses² of x , then $\bar{x} = \hat{x}$ by the following calculation:

$$\bar{x} = \bar{x}e = \bar{x}(x\hat{x}) = (\bar{x}x)\hat{x} = e\hat{x} = \hat{x}. \quad (1.2)$$

This same proof can be reproduced in cartesian type theory, and therefore holds for monoid objects in any cartesian monoidal category. It follows that if a monoid object M has an “inversion” morphism $i : M \rightarrow M$ such that

$$\begin{array}{ccccc}
 & M \times M & \xrightarrow{i \times 1} & M \times M & \\
 & \Delta \nearrow & & & \searrow m \\
 M & \xrightarrow{!} & 1 & \xrightarrow{e} & M \\
 & \Delta \searrow & & & \nearrow m \\
 & M \times M & \xrightarrow{1 \times i} & M \times M &
 \end{array} \quad (1.3)$$

commutes (thereby making it a group object), then i is the unique such morphism.

An interesting application of this relies on the fact that the category $\text{CComon}(\mathcal{C})$ of *cocommutative comonoids* in any symmetric monoidal category \mathcal{C} is in fact *cartesian* monoidal, with the cartesian product being the monoidal structure of \mathcal{C} . A monoid object

¹This “elsewhere” could be anywhere at all. Thus although $\lambda^A x$ does “bind” the variable x , it does not delimit its scope.

²In fact, of course, it suffices for \bar{x} to be a left inverse and \hat{x} a right inverse.

in $\text{CComon}(\mathcal{C})$ is a *cocommutative bimonoid object* in \mathcal{C} : an object with both a monoid and a cocommutative comonoid structure, such that the monoid structure maps are comonoid morphisms (or equivalently the comonoid structure maps are monoid morphisms). For any bimonoid object, a morphism $i : M \rightarrow M$ satisfying the analogue of (1.3):

$$\begin{array}{ccccc}
& M \otimes M & \xrightarrow{i \otimes 1} & M \otimes M & \\
\Delta \nearrow & & & & \searrow m \\
M & \xrightarrow{\varepsilon} & I & \xrightarrow{e} & M \\
\Delta \searrow & & & & \nearrow m \\
& M \otimes M & \xrightarrow{1 \otimes i} & M \otimes M &
\end{array} \tag{1.4}$$

(where Δ, ε are the comonoid structure) is called an *antipode*, and a bimonoid object equipped with an antipode is called a *Hopf monoid*. Thus, the classical argument for uniqueness of inverses, phrased in cartesian type theory, implies that a cocommutative bimonoid object has at most one antipode. Dualizing, we can conclude that a *commutative* bimonoid object also has at most one antipode.

For bimonoids that are neither commutative nor cocommutative, cartesian type theory cannot help us. We can manually translate the classical proof (1.2) into commutative diagrams and then replace all the \times 's by \otimes 's, but this is tedious and one feels that there should be a better way. And indeed, in our type theory for symmetric monoidal categories, we can write a very close analogue of (1.2) that applies to arbitrary bimonoids:

$$\bar{x} = \bar{x} e = (\overline{x_{(1)}} e \mid \cancel{x_{(2)}}) = \overline{x_{(1)}} x_{(2)} \widehat{x_{(3)}} = (e \widehat{x_{(2)}} \mid \cancel{x_{(1)}}) = e \widehat{x} = \widehat{x}. \tag{1.5}$$

Again, the reader is not expected to understand this completely, but the resemblance to (1.2) should be clear. The main differences are the subscripts on the x 's and the insertion of the steps $(\overline{x_{(1)}} e \mid \cancel{x_{(2)}})$ and $(e \widehat{x_{(2)}} \mid \cancel{x_{(1)}})$. The subscripts are used to track applications of the comultiplication $\Delta : M \rightarrow M \otimes M$. This is necessary because (unlike in the cartesian case) not all morphisms are comonoid morphisms (so that $f(x)_{(1)}$ might differ from $f(x_{(1)})$), and also because in the absence of cocommutativity the resulting “copies” of x must be distinguished (so that $(x_{(1)}, x_{(2)})$ is different from $(x_{(2)}, x_{(1)})$). Similarly, the canceled terms such as $\cancel{x_{(2)}}$ track applications of the counit $\varepsilon : M \rightarrow I$, and the steps involving these are inserted in order to match the middle composite $M \xrightarrow{\varepsilon} I \xrightarrow{e} M$ in (1.4).

With this as preview and motivation, we now move on to a somewhat more detailed description of our type theory.

1.6. GENERALIZED SWEEDLER NOTATION. The idea behind our type theory is to formalize and generalize the informal *Sweedler notation* that is common in coalgebra theory. A coalgebra is a comonoid in the category of vector spaces, i.e. a vector space C with a comultiplication $\Delta : C \rightarrow C \otimes C$ that is coassociative and counital. Since the elements of the tensor product $C \otimes C$ are finite sums of generating tensors $c_{(1)} \otimes c_{(2)}$, we can write

$$\Delta(c) = \sum_i c_{(1)}^i \otimes c_{(2)}^i.$$

Sweedler's notation is to omit the index i , and sometimes also the symbol \sum , obtaining

$$\Delta(c) = c_{(1)} \otimes c_{(2)}.$$

(This is therefore a sort of “dual” to the “Einstein summation convention” for products of tensors, which also has a type-theoretic formalization known to physicists as “abstract index notation” [PR84].) For instance, coassociativity can then be expressed as

$$c_{(1)} \otimes c_{(2)(1)} \otimes c_{(2)(2)} = c_{(1)(1)} \otimes c_{(1)(2)} \otimes c_{(2)}$$

(which is then often written as $c_{(1)} \otimes c_{(2)} \otimes c_{(3)}$), and counitality, for a counit ε , can be expressed as

$$c_{(1)} \cdot \varepsilon(c_{(2)}) = c = \varepsilon(c_{(1)}) \cdot c_{(2)}$$

where \cdot denotes scalar multiplication.

We generalize this notation to apply to any morphism $f : A \rightarrow B \otimes C$ in any monoidal category whose codomain is a tensor product. Of course we now have to note which morphism we are talking about, so instead of $c_{(1)} \otimes c_{(2)}$ we write $f_{(1)}(x) \otimes f_{(2)}(x) : B \otimes C$, where $x : A$ is a formal variable. Note that in Sweedler's original notation for coalgebras *in vector spaces*, the element $\Delta(c)$ really is a sum of generating tensors $c_{(1)} \otimes c_{(2)}$, whereas now we are in an arbitrary monoidal category so that this doesn't even make sense. Nevertheless, we can still use a similar *formal* notation governed by appropriate rules, just as in type theory we use variables and terms assigned to objects of the category as “types” even though the objects of an arbitrary category may not actually have “elements”.

However, as a nod towards this extra formality (and also an extra step in the “types are like sets” direction), we stop using the symbol \otimes and write instead a pair $(f_{(1)}(x), f_{(2)}(x)) : (B, C)$ (although see subsection 1.9). We also have to incorporate an analogue of “scalar multiplication”, which in a general monoidal category means the unit isomorphism $C \otimes I \cong C$; for consistency we collect all the “scalars” in a separate tuple separated by a bar. Thus, for instance, the counitality of a comultiplication $\Delta : C \rightarrow C \otimes C$ with counit $\varepsilon : C \rightarrow I$ in an arbitrary monoidal category will be expressed as

$$(\Delta_{(1)}(c) \mid \varepsilon(\Delta_{(2)}(c))) = c = (\Delta_{(2)}(c) \mid \varepsilon(\Delta_{(1)}(c))).$$

And the composite of $f : A \otimes B \rightarrow C \otimes D$ with $g : E \otimes D \rightarrow F \otimes G$ along the object D is

$$\frac{x : A, y : B \vdash (f_{(1)}(x, y), f_{(2)}(x, y)) : (C, D) \\ z : E, w : D \vdash (g_{(1)}(z, w), g_{(2)}(z, w)) : (F, G)}{x : A, y : B, z : E \vdash (f_{(1)}(x, y), g_{(1)}(z, f_{(2)}(x, y)), g_{(2)}(z, f_{(2)}(x, y))) : (C, F, G)}$$

Note that in contrast to intuitionistic linear logic, the variables in a context are not literally treated “linearly” in our terms, since they can occur multiple times in the multiple “components” of a map f . Instead, the “usages” of a variable are controlled by the codomain arity of the morphisms applied to them.

One further technical device is required to deal with morphisms having nullary domain. Suppose we have $f : I \rightarrow B \otimes C$, written in our type theory as $\vdash (f_{(1)}, f_{(2)}) : (B, C)$, and we compose/tensor it with itself (and apply a symmetry) to get a morphism $I \rightarrow B \otimes B \otimes C \otimes C$. We would naïvely write this as $\vdash (f_{(1)}, f_{(1)}, f_{(2)}, f_{(2)}) : (B, B, C, C)$, but this is ambiguous since we can't tell which $f_{(1)}$ matches which $f_{(2)}$. Thus, we disambiguate the possibilities by annotating the terms in pairs, such as $() \vdash (f_{(1)}, f'_{(1)}, f_{(2)}, f'_{(2)})$ or $() \vdash (f_{(1)}, f'_{(1)}, f'_{(2)}, f_{(2)})$.

The “variable-binding notation” $(x, \lambda^A x)$ for duality that we mentioned earlier is just syntactic sugar for a special case of this: we use variables like x as the “labels” for unit morphisms $\eta_A : () \rightarrow (A, A^*)$ in place of primes, writing $(x, \lambda^A x)$ instead of $((\eta_A)_{(1)}^x, (\eta_A)_{(2)}^x)$. Of course, this requires that the “variables” used as labels of this sort are disjoint from those occurring in the actual context. The operator \triangleleft is just an infix notation for the counit $\varepsilon : (A^*, A) \rightarrow ()$.

Another bit of syntactic sugar is that if each type is equipped with at most one specified comonoid structure, then there is no ambiguity in reverting back to Sweedler's original notation $(x_{(1)}, x_{(2)})$ instead of $(\Delta_{(1)}(x), \Delta_{(2)}(x))$. For instance, the axiom of a Frobenius algebra can be written as

$$(x_{(1)}, x_{(2)}y) = ((xy)_{(1)}, (xy)_{(2)}) = (xy_{(1)}, y_{(2)}) \quad (1.7)$$

while the principal axiom of a bialgebra is

$$((xy)_{(1)}, (xy)_{(2)}) = (x_{(1)}y_{(1)}, x_{(2)}y_{(2)}). \quad (1.8)$$

Note that if all types have such a comonoid structure *and* all morphisms are comonoid morphisms, which in type-theoretic notation means $f(x)_{(i)} = f(x_{(i)})$ and $\varepsilon(f(x)) = \varepsilon(x)$, then the monoidal structure automatically becomes cartesian. In ordinary type theory for cartesian monoidal categories, the diagonal (i.e. the comultiplication) is represented by literally duplicating a variable (x, x) ; thus our subscripting of variables $(x_{(1)}, x_{(2)})$ can be viewed as a minimal modification of this to deal with situations where the comultiplication is not literally a cartesian diagonal. (Indeed, the comultiplication of a coalgebra is often viewed as a non-cartesian substitute for the diagonal, e.g. in the theory of quantum groups.) We can similarly regard the counit ε as a “discarding” operation, in which case I like to notate $\varepsilon(x)$ by \cancel{x} (as was done in (1.5)).

1.9. AN ALTERNATIVE NOTATION FOR COALGEBRAISTS. The particular symbols we choose in the syntax of our type theory are, of course, immaterial. I personally find the pair notation $(f_{(1)}(x), f_{(2)}(x))$ evocative and helpful, since I am more used to thinking of sets than coalgebras. However, a reader who is more used to ordinary coalgebras in vector spaces may prefer to retain \otimes as a formal symbol in place of a comma, and use a symbol like \cdot for “scalar multiplication” in place of the separating bar $|$. For instance, in this notation the axioms (1.7) and (1.8) become

$$x_{(1)} \otimes x_{(2)}y = (xy)_{(1)} \otimes (xy)_{(2)} = xy_{(1)} \otimes y_{(2)} \quad (xy)_{(1)} \otimes (xy)_{(2)} = x_{(1)}y_{(1)} \otimes x_{(2)}y_{(2)}.$$

while the calculation (1.5) becomes

$$\overline{x} = \overline{x} e = \overline{x_{(1)}} e \cdot \varepsilon(x_{(2)}) = \overline{x_{(1)}} x_{(2)} \widehat{x_{(3)}} = e \widehat{x_{(2)}} \cdot \varepsilon(x_{(1)}) = e \widehat{x} = \widehat{x}.$$

A reader who prefers this notation is free to use it instead. (However, one should probably then choose a different notation for the tensor product *types* discussed in Remark 2.6.)

1.10. SOME PLURALISTIC REMARKS. Since many readers will be more familiar with other syntaxes for monoidal categories such as intuitionistic linear logic or string diagrams, it is worth saying a few words about their complementary relationship to our type theory, and in particular how and why they can all coexist.

Intuitionistic linear logic was discussed briefly above; from our present point of view, it has the disadvantages of a heavy and unintuitive “match” syntax for decomposing elements of tensor products. However, it also has definite advantages,³ such as an asymptotically more concise syntax: the present type theory involves much duplication of terms, since the arguments of a function are repeated in every component of its output. This matters little for the examples we consider here, but might become more important if it were ever implemented at scale in a computer proof assistant. Intuitionistic linear logic also cleanly extends to a syntax for *closed* monoidal categories, whereas with our current type theory this seems difficult.

String diagrams (a.k.a. graphical calculus) are another well-known and very powerful tool for working with structures in many kinds of monoidal categories, which also have both strengths and weaknesses. String diagrams are at their best when dealing with structures whose axioms “don’t change the topology”, such as dual pairs and Frobenius algebras, since in this case many proofs are simply topological deformations. For structures whose axioms do change the topology, such as bialgebras and Hopf algebras, string diagrams are still useful, but in such cases only some of the proof steps can be reduced to simple topological deformations: generally those steps involving pure *naturality* conditions.

By contrast, our type theory does not “see” the topological nature of dual pairs and Frobenius algebras. However, it represents most naturality conditions as *syntactic identities*, making them completely invisible; for instance, the two middle steps in Figure 1 disappear entirely in (1.5). Moreover, it leverages a different intuition, allowing us to think of objects of an arbitrary monoidal category as “sets” with “elements” (modulo certain “linearity” restrictions). This seems to be particularly useful for coalgebraic and Hopf-type structures — perhaps unsurprisingly, given the origin of our syntax in Sweedler notation.

Another difference is that string diagrams incarnate categorical duality in an obvious way — by simply rotating or reflecting a string diagram — while our type theory breaks this duality, treating inputs and outputs differently. One might consider this an advantage of string diagrams. But breaking duality in the syntax is part of what gives us the above advantages (notably the view of types as sets), and it also means that duality becomes a

³I am indebted to Matt Oliveri for these points.

nontrivial and useful technique: in a situation that is not self-dual, we can choose which orientation of the type theory is most convenient. For instance, in subsection 7.9 we will use our type theory to show that the antipode of a (weak) Hopf monoid is a monoid anti-homomorphism; it then follows by duality that it is also a comonoid anti-homomorphism, although a direct proof of the latter in our syntax would be less intuitive.

There is no reason to expect any one notation for symmetric monoidal categories to be the best in all situations: each has applications to which it is better-suited and others to which it is not as well suited. The new type theory presented here makes no claim to supplant string diagrams or intuitionistic linear logic; rather it is an alternative that may be more convenient in *some* applications. Only time and experience can render a final verdict on the usefulness of a notation, as well as its generalizations and limitations — e.g. to what extent can the type theory presented in this paper be generalized to other kinds of monoidal categories, such as closed, $*$ -autonomous, braided, ribbon, and planar monoidal categories, indexed monoidal categories, (symmetric) monoidal bicategories, and so on? My primary hope for this paper is to begin a conversation about type theories in this family and their potential uses.

1.11. ACKNOWLEDGEMENTS. I would like to thank Dan Licata, Robin Cockett, Peter LeFanu Lumsdaine, and Matt Oliveri for useful conversations.

2. Props

To simplify and clarify the semantics of our type theory, we will interpret it in a categorical structure that reflects the type-theoretic distinction between judgmental and type operations. To explain what this means, recall that ordinary cartesian type theory is often interpreted in categories with finite products; but in this case both the judgmental comma (in a context $(x : A, y : B, z : C)$) and the product type operation (in a product type $(A \times B) \times C$) are interpreted by the same categorical operation (cartesian product types), which can cause confusion. A more direct semantics of cartesian type theory maintains this distinction by using a “cartesian multicategory”, allowing the judgmental comma to be interpreted by the “categorical comma”, i.e. the concatenation that forms a list of objects to be the domain of a morphism in a multicategory.

For type theories like classical linear logic, which allow multiple types in both domain and codomain but use commas on the left and right of the turnstile to represent different tensor products, the appropriate “multicategorical” structure is called a “polycategory” [Sza75] (the corresponding “monoidal” version being a linearly distributive [CS97, BCST96] or $*$ -autonomous [Bar79, Bar91] category). In our case, where the two tensor products are the same, the appropriate structure is called a *prop*.

2.1. DEFINITION. A **prop** \mathcal{P} consists of

- (i) A set $\text{ob}(\mathcal{P})$ of **objects**, and

- (ii) A symmetric strict monoidal category (that is, a symmetric monoidal category whose associators and unitors are identities) whose underlying monoid of objects is freely generated by $\text{ob}(\mathcal{P})$.

(The original Adams–MacLane [Mac65] definition of prop had only one object; thus our “props” are sometimes called “colored props”.)

We write the objects of the monoidal category in (ii) as finite lists (A, B, \dots, Z) of objects of the prop (i.e. $A, B, \dots, Z \in \text{ob}(\mathcal{P})$). The monoidal structure is given by concatenation of lists; the unit object is the empty list $()$.

We now summarize the relationship between props and symmetric monoidal categories.

2.2. DEFINITION. A **tensor product** $A \otimes B$ of objects in a prop is an object together with an isomorphism

$$(A, B) \xrightarrow{\sim} (A \otimes B)$$

in the monoidal category of Definition 2.1(ii). Similarly, a **unit** is an object I with an isomorphism

$$() \xrightarrow{\sim} (I).$$

A prop is called **representable** if it has a unit and any pair of objects has a tensor product.

A **morphism of props** $\omega : \mathcal{P} \rightarrow \mathcal{Q}$ is a function $\omega_0 : \text{ob}(\mathcal{P}) \rightarrow \text{ob}(\mathcal{Q})$ together with a *strict* symmetric monoidal functor whose action on objects is obtained by applying ω_0 elementwise to lists.

2.3. THEOREM. *The category of symmetric monoidal categories and strong symmetric monoidal functors is equivalent to the subcategory of representable props (and all morphisms between them).*

SKETCH OF PROOF. Every symmetric monoidal category \mathcal{C} has an underlying prop $U\mathcal{C}$ with the same objects, and in which a morphism $(A, \dots, B) \rightarrow (C, \dots, D)$ is a morphism $A \otimes \dots \otimes B \rightarrow C \otimes \dots \otimes D$ in \mathcal{C} . This construction U is functorial on strong symmetric monoidal functors, and the props in its image are representable. (In fact, it is the functorial strictification.) The functor U is faithful since the action of $f : \mathcal{C} \rightarrow \mathcal{D}$ on objects and arrows is preserved in Uf , while the coherence constraints of f are recorded in the action of Uf on the isomorphisms from Definition 2.2. Similarly, the functor U is full, since the action of a morphism $U\mathcal{C} \rightarrow U\mathcal{D}$ on the isomorphisms from Definition 2.2 induces symmetric monoidal constraints on its underlying ordinary functor. Finally, every representable prop induces (by choosing tensor products and a unit) a symmetric monoidal structure on its category of unary and co-unary morphisms, and it is isomorphic to the underlying prop thereof. ■

We can say more: this subcategory is an injectivity-class, and the corresponding category of *algebraic* injectives (objects equipped with chosen lifts against the generating morphisms, and maps that preserve the chosen lifts) is equivalent to the category of symmetric monoidal categories and *strict* symmetric monoidal functors. In particular, the latter is monadic over the category of props.

The fundamental “initiality theorem” for semantics of our type theory, which we prove in section 5, is that the “term model” is the prop freely generated by some input data. Following [BCR18], we will call this input data a *signature*.

2.4. DEFINITION. A **signature** \mathcal{G} is a set of objects together with a set of arrows, each assigned a domain and codomain that are both finite lists of objects.

2.5. THEOREM. *The category of props is monadic over the category of signatures.*

PROOF. Just like the proof in [BCR18, Appendix A.2] for the one-object case. ■

Thus, every prop has a *presentation* as a coequalizer of a pair of maps between free props. In section 6 we will extend our type theory to construct “presented props” as well, allowing equational reasoning as in the examples from subsection 1.1.

2.6. REMARK. The fact that props allow multiple objects in both domains and codomains means that we rarely need to talk about actual tensor product types $A \otimes B$ with semantics in tensor product objects (Definition 2.2). For this reason, we will not include such types in our formal system. However, we note that if necessary, they can be added quite easily: because Definition 2.2 simply asserts objects, morphisms, and equations (rather than a unique factorization property), it can essentially be ensured as a special case of a prop presentation. Thus we need only add some generating terms and axioms to our type theory, for any pair of objects A, B whose tensor product we need to talk about:

$$\begin{array}{ll} x : A, y : B \vdash \langle x, y \rangle : A \otimes B & p : A \otimes B \vdash (\pi_{(1)}(p), \pi_{(2)}(p)) : (A, B) \\ (x, y) = (\pi_{(1)}\langle x, y \rangle, \pi_{(2)}\langle x, y \rangle) & p = \langle \pi_{(1)}(p), \pi_{(2)}(p) \rangle. \end{array}$$

Note how similar this is to the treatment of cartesian products in cartesian type theory with pairing and projection operations.

3. On the admissibility of structural rules

In general, type theories consist of *rules* for deriving *judgments* about *terms*. The most common judgments are *typing judgments* (that a term belongs to a type, or in our case a tuple of terms belong to a tuple of types) and *equality judgments* (that two terms — or tuples of terms — are equal). A tree of rules ending with a judgment is called a *derivation* of that judgment, and the categorical structure presented by a type theory is built out of the *derivable* (or *valid*) judgments.

Now in proving that this categorical structure does in fact have the desired universal property, it is very useful if we arrange the type theory so that every derivable judgment

has a *unique* derivation. The reason for this is that we want a morphism in our categorical semantics to be determined by a *term itself*, not by a choice of derivation of that term; but the natural way to prove the desired universal property (a.k.a. the “initiality theorem” for that type theory) is by induction *over derivations*. Thus, if the same term can arise from multiple derivations, proving this universal property requires an extra step of proving that this choice is immaterial (i.e. that any two derivations of the same term determine the same morphism in the semantics). This step is tricky and often omitted in the literature, leading to incomplete proofs. It becomes even trickier when considering higher-categorical semantics, in which the morphisms determined by two derivations of the same term may be only *isomorphic* rather than equal.

The choice that terms should have unique derivations essentially requires that nearly all structural rules should be admissible rather than primitive. (Recall that an *admissible rule* is one that is *not* asserted as part of the specification of the type theory, but for which we can prove after the fact that whenever we have derivations of its premises we can construct a derivation of its conclusion — usually by inductively traversing and modifying the given derivations of its premises. The *structural rules* are, roughly speaking, those that correspond to the operations of the categorical structure used as the semantics: composition in a category, permutation of domain lists in a symmetric multicategory, and permutation of both domains and codomains in a polycategory or prop.)

The reason this requirement arises is that structural rules generally have to satisfy equations that are “tautological” in their action on terms. For instance, composing $f : A \rightarrow B$ with $g : B \rightarrow C$ and $h : C \rightarrow D$ in the two possible ways (semantically, $h \circ (g \circ f)$ and $(h \circ g) \circ f$) produces the same term:

$$\frac{\begin{array}{c} x : A \vdash f(x) : B \quad y : B \vdash g(y) : C \\ \hline x : A \vdash g(f(x)) : C \end{array}}{x : A \vdash h(g(f(x))) : D}$$

$$\frac{\begin{array}{c} y : B \vdash g(y) : C \quad z : C \vdash h(z) : D \\ \hline y : B \vdash h(g(y)) : D \end{array}}{x : A \vdash h(g(f(x))) : D} .$$

Thus, if composition were a primitive rule, this term would have two distinct derivations. But if composition (i.e. substitution) is an admissible rule, then we can *prove* that the derivations *constructed* by applying it in these two different ways *turn out to be* the same. Similarly, if permutation were primitive, then for any $f : (A, B) \rightarrow C$ we would have two (in fact, infinitely many) derivations of the same term:

$$\frac{}{x : A, y : B \vdash f(x, y) : C} \qquad \frac{\begin{array}{c} x : A, y : B \vdash f(x, y) : C \\ \hline y : B, x : A \vdash f(x, y) : C \end{array}}{x : A, y : B \vdash f(x, y) : C}$$

whereas if permutation (a.k.a. “exchange”) is admissible, then its functoriality as an operation on derivations can be proven.

Type theorists know how to make a rule admissible: we build “just enough” of it into the primitive rules. For instance, if we introduce generating morphisms such as $f : (A, B) \rightarrow C$ and $g : C \rightarrow D$ as simple axioms (i.e. rules with no premises):

$$\overline{x : A, y : B \vdash f(x, y) : C} \quad \overline{z : C \vdash g(z) : D}$$

then there would be no way to construct derivations of composites such as

$$\frac{x : A, y : B \vdash f(x, y) : C \quad z : C \vdash g(z) : D}{x : A, y : B \vdash g(f(x, y)) : D} \quad (3.1)$$

except by using a *primitive* composition/substitution rule. Therefore, we instead introduce each generating morphism in “Yoneda style” by allowing ourselves to *postcompose* any given term(s) with it. For instance, in cartesian type theory we introduce generators with rules such as

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash f(a, b) : C} \quad \frac{\Gamma \vdash c : C}{\Gamma \vdash g(c) : D}$$

for arbitrary contexts Γ and terms a, b, c . Now (3.1) can be obtained without a primitive substitution rule:

$$\frac{\overline{x : A, y : B \vdash x : A} \quad \overline{x : A, y : B \vdash y : B}}{\overline{x : A, y : B \vdash f(x, y) : C}} \quad \frac{}{x : A, y : B \vdash g(f(x, y)) : D} \quad (3.2)$$

In categorical terms, the point is that we can build the free category on a graph either by freely adding all binary (and nullary) composites and then quotienting by the relation of associativity, or we can avoid the need to quotient at all by defining the morphisms as composable lists of generating arrows, where a “list” is defined inductively as either an empty list or a list postcomposed by a generating arrow — that is, we enforce right-associated composites $k \circ (h \circ (g \circ (f \circ \text{id})))$.

This technique is trickier in non-cartesian type theories, since we cannot keep the same context all the way through the derivation. That is, in a cartesian monoidal category we can start (3.2) with the “identity” or “axiom” rules $x : A, y : B \vdash x : A$ and $x : A, y : B \vdash y : B$, corresponding categorically to the projections $A \times B \rightarrow A$ and $A \times B \rightarrow B$; but in a non-cartesian monoidal category such projections do not exist. Thus, we need to concatenate contexts as we go down the derivation tree. For instance, the generator rule for $f : (A, B) \rightarrow C$ must be something like

$$\frac{\Gamma \vdash a : A \quad \Delta \vdash b : B}{\Gamma, \Delta \vdash f(a, b) : C}. \quad (3.3)$$

$$\begin{aligned} x_1 : A_1, \dots, x_m : A_m &\vdash (M_1, \dots, M_n \mid Y_1, \dots, Y_p) : (B_1^{*?}, \dots, B_n^{*?}). \\ x_1 : A_1, \dots, x_m : A_m &\vdash (M_1, \dots, M_n \mid Y_1, \dots, Y_p) = (N_1, \dots, N_n \mid Z_1, \dots, Z_q) : (B_1, \dots, B_n). \end{aligned}$$

Figure 2: Judgments

However, now we have a new problem: the exchange rule (permutations of the domain). In the cartesian case, we can make this admissible by “propagating it up” the entire derivation since the context remains the same. For instance, if we permute the inputs of (3.2) we would simply get

$$\frac{\begin{array}{c} \overline{y : B, x : A \vdash x : A} \quad \overline{y : B, x : A \vdash y : B} \\ \hline y : B, x : A \vdash f(x, y) : C \end{array}}{y : B, x : A \vdash g(f(x, y)) : D} \tag{3.4}$$

But in the non-cartesian case this doesn’t work. We don’t want to assert a primitive exchange rule as this would break the “terms have unique derivations” principle, so instead we build exchange into the generator rule (3.3). Our first try might be something like

$$\frac{\Gamma \vdash a : A \quad \Delta \vdash b : B \quad \sigma : \Gamma, \Delta \cong \Phi}{\Phi \vdash f(a, b) : C}$$

where σ is an arbitrary permutation. But this too breaks the “terms have unique derivations” principle, since a permutation of types *within* Γ or Δ could be obtained either as part of σ or by operating on the input derivations of $\Gamma \vdash a : A$ and $\Delta \vdash b : B$. Instead we have to build in “just enough” exchange but not too much, by requiring σ to be not an arbitrary permutation but a *shuffle*: a permutation of (Γ, Δ) that preserves the *relative* order of the types in Γ and in Δ . We will write $\text{Shuf}(\Gamma; \Delta)$ for the set of such shuffles.

This may seem overly technical, but the presence of such things as shuffles need never be seen by the *user* of the type theory. Indeed, maintaining the “terms have unique derivations” principle is precisely what *allows* the user to work only with terms, ignoring the details of the derivations.

4. The type theory for free props

Let \mathcal{G} be a signature; we will define a type theory that presents the free prop on \mathcal{G} . Our general typing judgment will be of the form shown in Figure 2. For reasons to be explained later, we annotate some of the types in the consequent of each judgment with a superscript star, B^* , and call them *active*; we write $B^{*?}$ to mean that B might be active.

We use vector notation \vec{M}, \vec{A} , etc. to indicate a list of terms or types, so the judgments can be abbreviated as in Figure 3, although this omits the information that \vec{M} and \vec{N}

$$\Gamma \vdash (\vec{M} \mid \vec{Y}) : \vec{B}^{*\?} \quad \Gamma \vdash (\vec{M} \mid \vec{Y}) = (\vec{N} \mid \vec{Z}) : \vec{B}.$$

Figure 3: Judgments, abbreviated

must have the same length as \vec{B} (while the length of \vec{Y} and \vec{Z} is unrestricted). If \vec{Y} is empty, we write $(\vec{M} \mid)$ as simply (\vec{M}) . If furthermore \vec{M} and \vec{B} have length 1, we omit the parentheses, writing simply $M : B$. When all the lists are empty, we have $\vdash () : ()$, which will be valid (it represents the identity morphism of the unit object).

In these judgments A_i, B_j are types, x_i are variables, and M_j, N_j, Y_k, Z_ℓ are terms. Here by a *type* we simply mean an object of our generating signature \mathcal{G} , since there are no type-forming operations in the theory. There is nothing new in our *variables*; the reader who prefers de Bruijn indices is free to think of them in that way, although since our syntax has no variable binding⁴ the usual subtleties of capture-avoidance are irrelevant.

The *terms* are defined inductively by the rules shown in Figure 4. Recall that in general our typing judgments involve a *list* of such terms, one for each type in the codomain, together with a list of scalar terms. The terms defined in Figure 4 do not yet have any types, but they are “well-scoped” by definition: they come with a context and only use variables occurring in that context. We subscript only applications of functions with greater than unary codomain, and as noted in section 1, we annotate each occurrence of a morphism with nullary domain and positive-ary codomain with some element of an infinite alphabet of symbols \mathfrak{A} (such as ‘”, “, . . . , or 1, 2, 3, . . .).

We will write $\vec{f}(\vec{M})$ for the list of all subscriptings of the application of f to the arguments \vec{M} . For instance, if $f : (A, B, C) \rightarrow (D, E)$ then $\vec{f}(\vec{M})$ would be

$$(f_{(1)}(M_1, M_2, M_3), f_{(2)}(M_1, M_2, M_3)).$$

More generally, for $f \in \mathcal{G}(A_1, \dots, A_m; B_1, \dots, B_n)$ we have

$$\vec{f}(\vec{M}) = \begin{cases} (f_{(1)}^{\mathfrak{a}}, \dots, f_{(n)}^{\mathfrak{a}}) & m = 0, n \geq 2 \\ f^{\mathfrak{a}} & m = 0, n = 1 \\ f & m = 0, n = 0 \\ (f_{(1)}(\vec{M}), \dots, f_{(n)}(\vec{M})) & m \geq 1, n \geq 2 \\ f(\vec{M}) & m \geq 1, n \leq 1. \end{cases}$$

In the first and second case, we also write $\vec{f}^{\mathfrak{a}}$ to notate the label \mathfrak{a} . In the third case, we allow ourselves to write $\vec{f}^{\mathfrak{a}}$ to mean simply f (discarding the label). Finally, when $n \leq 1$ we allow ourselves to write $f_{(1)}$ or $f_{(1)}(\vec{M})$ to mean f or $f(\vec{M})$ respectively.

We define the “simultaneous substitution” of a list of terms \vec{M} for a list of variables \vec{x} in the usual way, as shown in Figure 5.

⁴Recall that in the notation $(x, \lambda^A x)$ for the unit of a duality, x is not actually a variable in this sense but rather a “label”; we will introduce labels formally in a moment.

$$\begin{array}{c}
\frac{(x : A) \in \Gamma}{\Gamma \vdash x \text{ term}} \quad \frac{f \in \mathcal{G}(; B_1, \dots, B_n) \quad \mathfrak{a} \in \mathfrak{A} \quad n \geq 2 \quad 1 \leq k \leq n}{\Gamma \vdash f_{(k)}^{\mathfrak{a}} \text{ term}} \\[10pt]
\frac{f \in \mathcal{G}(; B) \quad \mathfrak{a} \in \mathfrak{A}}{\Gamma \vdash f^{\mathfrak{a}} \text{ term}} \quad \frac{f \in \mathcal{G}(;) }{\Gamma \vdash f \text{ term}} \\[10pt]
\frac{\Gamma \vdash M_1 \text{ term} \quad \dots \quad \Gamma \vdash M_m \text{ term} \quad \Gamma \vdash M_1 \text{ term} \quad \dots \quad \Gamma \vdash M_m \text{ term}}{f \in \mathcal{G}(A_1, \dots, A_m; B_1, \dots, B_n) \quad m \geq 1 \quad n \geq 2 \quad 1 \leq k \leq n} \\[10pt]
\frac{\Gamma \vdash M_1 \text{ term} \quad \dots \quad \Gamma \vdash M_m \text{ term}}{f \in \mathcal{G}(A_1, \dots, A_m; B_1, \dots, B_n) \quad m \geq 1 \quad n \leq 1} \\[10pt]
\frac{}{\Gamma \vdash f(M_1, \dots, M_m) \text{ term}}
\end{array}$$

Figure 4: Terms

$$\begin{aligned}
x_k[M_1, \dots, M_n/x_1, \dots, x_n] &= M_k \\
y[\vec{M}/\vec{x}] &= y \quad (y \notin \vec{x}) \\
f_{(k)}^{\mathfrak{a}}[\vec{M}/\vec{x}] &= f_{(k)}^{\mathfrak{a}} \\
f^{\mathfrak{a}}[\vec{M}/\vec{x}] &= f^{\mathfrak{a}} \\
f[\vec{M}/\vec{x}] &= f \\
f_{(k)}(N, \dots, P)[\vec{M}/\vec{x}] &= f_{(k)}(N[\vec{M}/\vec{x}], \dots, P[\vec{M}/\vec{x}]) \\
f(N, \dots, P)[\vec{M}/\vec{x}] &= f(N[\vec{M}/\vec{x}], \dots, P[\vec{M}/\vec{x}])
\end{aligned}$$

Figure 5: Simultaneous substitution into terms

We now move on to the rules governing our typing judgment. In section 3 we argued that by incorporating Yoneda-style generator rules and shuffles, we can make composition and exchange admissible and thereby ensure that any judgment has a unique derivation. In the case of props, we also want to make the monoidal structure admissible (since it satisfies strict associativity and interchange laws that we would otherwise have to assert as judgmental equalities). In particular, for morphisms $f : A \rightarrow B$ and $g : C \rightarrow D$ we would like the judgment

$$x : A, y : C \vdash (f(x), g(y)) : (B, D)$$

to have a unique derivation. Symmetry suggests that this unique derivation cannot apply f first and then g or vice versa. Thus, we replace the generator rule by a “multi-generator” rule allowing only a one-step derivation

$$\frac{x : A, y : C \vdash (x, y) : (A, C)}{x : A, y : C \vdash (f(x), g(y)) : (B, D)}$$

A first approximation to the general form of this rule is

$$\frac{\Gamma \vdash (\vec{M}, \dots, \vec{N}, \vec{P} \mid \vec{Z}) : (\vec{A}, \dots, \vec{B}, \vec{C}) \quad f \in \mathcal{G}(\vec{A}; \vec{D}) \quad \dots \quad g \in \mathcal{G}(\vec{B}; \vec{E})}{\Gamma \vdash (\vec{f}(\vec{M}), \dots, \vec{g}(\vec{N}), \vec{P} \mid \vec{Z}) : (\vec{D}, \dots, \vec{E}, \vec{C})}.$$

However, if there are generators with nullary codomain, we need to collect them into the scalar terms \vec{Z} . Thus a second approximation is

$$\frac{\Gamma \vdash (\vec{M}, \dots, \vec{N}, \vec{P}, \dots, \vec{Q}, \vec{R} \mid \vec{Z}) : (\vec{A}, \dots, \vec{B}, \vec{C}, \dots, \vec{D}, \vec{E}) \quad f \in \mathcal{G}(\vec{A}; \vec{F}_{\geq 1}) \quad \dots \quad g \in \mathcal{G}(\vec{B}; \vec{G}_{\geq 1}) \quad h \in \mathcal{G}(\vec{C}; \cdot) \quad \dots \quad k \in \mathcal{G}(\vec{D}; \cdot)}{\Gamma \vdash (\vec{f}(\vec{M}), \dots, \vec{g}(\vec{N}), \vec{R} \mid h(\vec{P}), \dots, k(\vec{Q}), \vec{Z}) : (\vec{F}, \dots, \vec{G}, \vec{E})}.$$

(Here $\vec{F}_{\geq 1}$ means that \vec{F} contains at least one type.) Eventually we will also incorporate shuffles, but we postpone that for now. Let us consider instead how to prevent duplication of derivations. In addition to our desired term

$$x : A, y : C \vdash (f(x), g(y)) : (B, D) \tag{4.1}$$

we must also be able to write both of the following:

$$x : A, y : C \vdash (f(x), y) : (B, C) \tag{4.2}$$

$$x : A, y : C \vdash (x, g(y)) : (A, D). \tag{4.3}$$

So how do we prevent ourselves from being able to apply the generator rule again to the latter two, obtaining two more derivations of the same morphism as (4.1)?

There are different possible choices that one could make here, potentially leading to different type theories and different “normal forms” for the morphisms in a free prop. The choice we will make is to force ourselves to “apply all functions as soon as possible”. Thus, for instance, we forbid ourselves from applying g to y in (4.2) because we *could have* already applied it to produce (4.1). On the other hand, we will still allow ourselves to apply $h : (B, C) \rightarrow E$ in (4.2) to get

$$x : A, y : C \vdash (h(f(x), y)) : E$$

because h has f as an input, hence could not have been applied at the same time as f .

Making this precise is the purpose of designating some of the types in the consequent as **active**; recall that we denote the active types by A^* . If \vec{A} is a list of types, we write $\vec{A}^{*\geq 1}$ to mean that at least one of the types in \vec{A} is active, \vec{A}^* to mean that they are all active, and $\vec{A}^{*=0}$ to mean that none of them are active. As noted previously, we write $\vec{A}^{*?}$ to avoid specifying whether or not any of the types are active.

The identity rule will make all types active, while the generator rule makes only the outputs of the generators active. We then restrict the generator rule to require that at least one of the *inputs* of each generator being applied must be active in the premise; this means that none of them could have been applied any sooner, since at least one of their arguments was just introduced by the previous rule. Thus, our desired derivation

$$\frac{x : A, y : C \vdash (x, y) : (A^*, C^*)}{x : A, y : C \vdash (f(x), g(x)) : (B^*, D^*)}$$

is allowed, while the undesired one

$$\frac{\begin{array}{c} x : A, y : C \vdash (x, y) : (A^*, C^*) \\ \hline x : A, y : C \vdash (f(x), y) : (B^*, C) \end{array}}{i \quad x : A, y : C \vdash (f(x), g(y)) : (B, D)} \quad ?$$

is not allowed, since in the attempted application of g the input type C is not active. Thus our generator rule now becomes

$$\frac{\begin{array}{c} \Gamma \vdash (\vec{M}, \dots, \vec{N}, \vec{P}, \dots, \vec{Q}, \vec{R} \mid \vec{Z}) : (\vec{A}^{*\geq 1}, \dots, \vec{B}^{*\geq 1}, \vec{C}^{*\geq 1}, \dots, \vec{D}^{*\geq 1}, \vec{E}^{*?}) \\ f \in \mathcal{G}(\vec{A}; \vec{F}_{\geq 1}) \quad \dots \quad g \in \mathcal{G}(\vec{B}; \vec{G}_{\geq 1}) \\ h \in \mathcal{G}(\vec{C};) \quad \dots \quad k \in \mathcal{G}(\vec{D};) \end{array}}{\Gamma \vdash \left(\vec{f}(\vec{M}), \dots, \vec{g}(\vec{N}), \vec{R} \mid \vec{h}(\vec{P}), \dots, \vec{k}(\vec{Q}), \vec{Z} \right) : (\vec{F}^*, \dots, \vec{G}^*, \vec{E}^{*=0})}.$$

Of course, this rule can now never apply to generators with nullary domain. Since these can always be applied at the very beginning, we incorporate them into the identity rule.

Thus the identity rule is now

$$\frac{f \in \mathcal{G}(\vec{B}_{\geq 1}) \quad \dots \quad g \in \mathcal{G}(\vec{C}_{\geq 1}) \\ h \in \mathcal{G}(\cdot) \quad \dots \quad k \in \mathcal{G}(\cdot) \\ \mathbf{a}, \dots, \mathbf{b} \in \mathfrak{A} \text{ and pairwise distinct}}{\vec{x} : \vec{A} \vdash (\vec{x}, \vec{f}^{\mathbf{a}}, \dots, \vec{g}^{\mathbf{b}} \mid h, \dots, k) : (\vec{A}^*, \vec{B}^*, \dots, \vec{C}^*)}.$$

Note the labels on the terms with nullary domain and positive-ary codomain, as promised.

Finally, if we want to make the exchange rule admissible, we have to build permutations into the rules as well. Each rule should add exactly the part of a permutation that can't be "pushed into the premises". Because we've formulated the generator rule so that the premise and conclusion have the same context, any desired permutation in the domain can be pushed all the way up to the identity rule. Thus, for the generator rule it remains to deal with permutation in the codomain.

The freedom we have in the premises of the generator rule is to (inductively) permute the types *within* each list $\vec{A}, \vec{B}, \vec{C}, \vec{D}, \vec{E}$, and also to block-permute the lists \vec{A}, \dots, \vec{B} and separately the lists \vec{C}, \dots, \vec{D} (with a corresponding permutation of the generators f, \dots, g and h, \dots, k). (If we permuted the main premise any more than this, it would no longer have the requisite shape to apply the rule to.) Permutations of \vec{C}, \dots, \vec{D} don't do us any good in terms of permuting the codomain of the conclusion, but we can push permutations of \vec{E} directly into the premise, and also a block-permutation of \vec{F}, \dots, \vec{G} into a block-permutation of \vec{A}, \dots, \vec{B} .

What remains that we have to build into the rule can be described precisely by a permutation of $\vec{F}, \dots, \vec{G}, \vec{E}$ that (1) preserves the relative order of the types in \vec{E} , and (2) preserves the relative order of the *first* types F_1, \dots, G_1 in the lists \vec{F}, \dots, \vec{G} . That is, any permutation of $\vec{F}, \dots, \vec{G}, \vec{E}$ can be factored uniquely as one with these two properties followed by a block sum of a block-permutation of \vec{F}, \dots, \vec{G} with a permutation of \vec{E} . (The choice of the first types is arbitrary; we could just as well use the last types, etc.)

There is no real need to allow ourselves to permute the scalar terms, since semantically their order doesn't matter anyway. But it is convenient to allow ourselves to write the scalar terms in any order, so we incorporate permutations there too. The freedom in the premises allows us to permute the term in \vec{Z} arbitrarily, and also to permute the terms h, \dots, k among themselves; thus what remains is precisely a shuffle. The final generator rule is therefore the first rule shown in Figure 6.

In the identity rule, the only useful freedom in the premises is to block-permute the \vec{B}, \dots, \vec{C} . Thus what remains is a permutation that preserves the relative order of the first types B_1, \dots, C_1 . Any permutation in the scalar terms can be pushed into the premises, so we have the final rule shown second in Figure 6. Note that this also allows us to incorporate an arbitrary permutation in the domain.

Having introduced the auxiliary notion of "active types" to ensure that typing judgments have unique derivations, we now proceed to eliminate it. We start with the following observation. The notion of *subterm* is defined as usual; we write $M \equiv N$ to mean that M and N are syntactically the same term.

$$\begin{array}{c}
\Gamma \vdash (\vec{M}, \dots, \vec{N}, \vec{P}, \dots, \vec{Q}, \vec{R} \mid \vec{Z}) : (\vec{A}^{\star \geq 1}, \dots, \vec{B}^{\star \geq 1}, \vec{C}^{\star \geq 1}, \dots, \vec{D}^{\star \geq 1}, \vec{E}^{\star ?}) \\
f \in \mathcal{G}(\vec{A}; \vec{F}_{\geq 1}) \quad \dots \quad g \in \mathcal{G}(\vec{B}; \vec{G}_{\geq 1}) \\
h \in \mathcal{G}(\vec{C};) \quad \dots \quad k \in \mathcal{G}(\vec{D};) \\
\sigma : (\vec{F}^{\star}, \dots, \vec{G}^{\star}, \vec{E}^{\star=0}) \xrightarrow{\sim} \Delta \text{ preserving activeness} \\
\sigma \text{ preserves the relative order of types in } \vec{E} \\
\sigma \text{ preserves the relative order of } F_1, \dots, G_1 \quad \tau \in \text{Shuf}(h, \dots, k; \vec{Z}) \\
\hline
\Gamma \vdash \left(\sigma \left(\vec{f}(\vec{M}), \dots, \vec{g}(\vec{N}), \vec{R} \right) \mid \tau \left(h(\vec{P}), \dots, k(\vec{Q}), \vec{Z} \right) \right) : \Delta
\end{array}$$

$$\begin{array}{c}
f \in \mathcal{G}(\vec{B}_{\geq 1}) \quad \dots \quad g \in \mathcal{G}(\vec{C}_{\geq 1}) \\
h \in \mathcal{G}(\vec{B}_{\geq 1}) \quad \dots \quad k \in \mathcal{G}(\vec{C}_{\geq 1}) \\
\mathfrak{a}, \dots, \mathfrak{b} \in \mathfrak{A} \text{ and pairwise distinct} \\
\sigma : (\vec{A}^{\star}, \vec{B}^{\star}, \dots, \vec{C}^{\star}) \xrightarrow{\sim} \Delta \text{ preserving activeness} \\
\sigma \text{ preserves the relative order of } B_1, \dots, C_1 \\
\hline
\vec{x} : \vec{A} \vdash \left(\sigma \left(\vec{x}, \vec{f}^{\mathfrak{a}}, \dots, \vec{g}^{\mathfrak{b}} \right) \mid h, \dots, k \right) : \Delta
\end{array}$$

Figure 6: Rules of the typing judgment

4.4. LEMMA. *If $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$ is derivable, and some M_i is a subterm of some M_j , then $i = j$ (hence $M_i \equiv M_j$).*

PROOF. By induction on the derivation. In an application of the identity rule, the non-scalar terms have no proper subterms, so the assumption means that $M_i \equiv M_j$. And since these terms are also uniquely identified by their label and subscript, we have $i = j$.

For an application of the generator rule, all the non-scalar terms either occur verbatim in the main premise, or are of the form $f_{(k)}(\vec{M})$ (including $f(\vec{M})$ when $k = 1$) where each M_i is a non-scalar term in the main premise and $|\vec{M}| \geq 1$. These two possibilities are mutually exclusive, and hence partition the terms into two classes that we call *old* and *new* respectively.

The inductive hypothesis takes care of the case when both terms are old. If a new term $f_{(k)}(\vec{M})$ is a subterm of an old term N , then each M_i is a proper subterm of N , contradicting the inductive hypothesis. This includes the case when an old term equals a new term; while if an old term N is a proper subterm of a new term $f_{(k)}(\vec{M})$, then it must be a subterm of some M_i , also contradicting the inductive hypothesis.

If one new term $f_{(k)}(\vec{M})$ is a proper subterm of another $g_{(\ell)}(\vec{N})$, then it must be a subterm of some N_j . Hence each M_i must be a proper subterm of N_j , contradicting the inductive hypothesis.

Finally, suppose two new terms are syntactically equal, $f_{(k)}(\vec{M}) \equiv g_{(\ell)}(\vec{N})$. Then we must have $f = g$, $k = \ell$, and $\vec{M} \equiv \vec{N}$. Note that $f = g$ means that f and g are the

same function symbol (morphism in \mathcal{G}), but not *a priori* that they arise from the same generator *application* in the rule (a given instance of the generator rule could apply the same generator more than once). However, $\vec{M} \equiv \vec{N}$ and the inductive hypothesis ensure that they *do* arise from the same generator application. Together with $k = \ell$, this implies that they have the same place in the given judgment as well. ■

We are primarily interested in applying Lemma 4.4 in the case of an “improper subterm” $M_i \equiv M_j$, but the stronger hypothesis makes the induction go through more easily.

4.5. REMARK. Semantically, it is not really necessary to label the morphisms in $\mathcal{G}(\cdot; B)$ with a symbol $\alpha \in \mathfrak{A}$, since tensor products of such morphisms are invariant under permutation (because the swap on the unit object is the identity morphism). However, it would be significantly trickier to omit such labels syntactically. In practice, we can leave them off informally and trust the reader to put them back if needed.

4.6. THEOREM. *If there is some assignment of activeness to the types in Δ such that $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$ is derivable, then that assignment is unique, as is the derivation.*

PROOF. By the *height* of an occurrence of a variable or function symbol in a term, we mean its maximum distance to a leaf node in the abstract syntax tree representing the term. Thus the height of a variable is 0, and the height of an occurrence of a function symbol is the least natural number strictly greater than the heights of the head symbols of all its arguments. Note that a nullary function symbol always has height 0, while a function symbol applied to a positive number of variables alone has height 1.

We claim that in any derivable typing judgment, the terms associated to active types are precisely those non-scalar ones whose head symbol has maximum height. The proof is by induction on derivations. In the identity rule, all terms have height 0 and all types are active. Now consider the generator rule, and suppose inductively that the claim is true for the main premise, with maximal height n , say. Then since each of the new function symbols introduced by the rule is applied to at least one term from an active type, which therefore has the maximal height n , it must have height $n + 1$. It follows that the new maximum height is $n + 1$, and that these new symbols are precisely those of maximum height; but they are also precisely those associated to active types. This proves the claim.

It follows immediately that the terms uniquely determine the activeness of the types, since height is a syntactic invariant of the terms. Moreover, we can tell from the terms which rule must have been applied last (if the maximum height is 0, it must come from the identity rule; otherwise it must come from the generator rule) and which function symbols that rule must have introduced (those of maximum height).

In the case of the generator rule, we can also conclude that the new terms must be precisely those whose head generator symbol has maximum height. We divide these new terms into subsets that have the same generator symbol and arguments, so that each subset consists of terms $f_{(i)}(\vec{M})$ for fixed f and \vec{M} but varying i . By Lemma 4.4, there can be no more than one occurrence of a particular term $f_{(i)}(\vec{M})$, so the subset of new

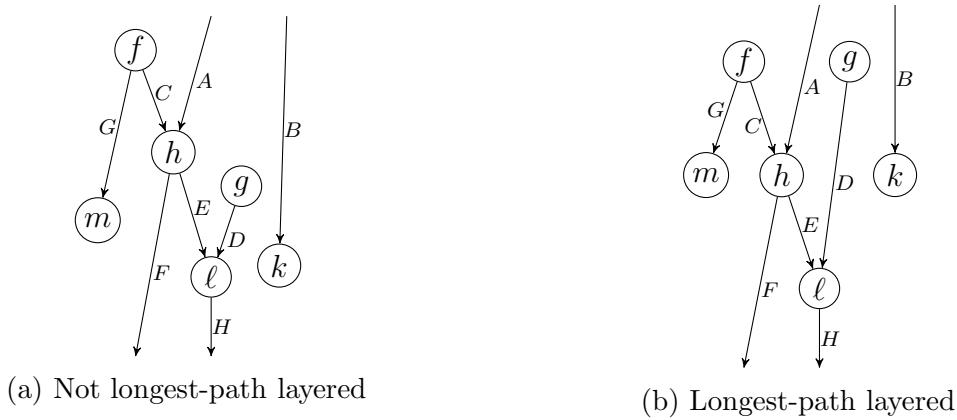


Figure 7: String diagrams for a morphism in a free prop

terms determined by f and \vec{M} must consist of exactly the m terms $f_{(i)}(\vec{M})$ for $1 \leq i \leq m$, where f has m outputs.

Now the ordering of these function symbol applications as f, \dots, g and h, \dots, k must be the order in which the corresponding $f_{(1)}, \dots, g_{(1)}$ and h, \dots, k appear in the term list, since the permutations σ and τ preserve those orders. Then σ^{-1} is uniquely determined by the fact that it must place all the outputs of f first, and so on until all the outputs of g , then all the terms of non-maximum height in the same order that they were given in the conclusion. Similarly, τ^{-1} is uniquely determined by the fact that it has to place h, \dots, k first and the scalar terms of non-maximum height last, preserving internal order in each group. Finally, this determines the main premise uniquely as well.

The argument for the identity rule is similar, with no τ and with σ^{-1} placing all the variables first in the order of the context. Inductively, therefore, the entire derivation is uniquely determined. ■

Note that we can regard this proof as an algorithm for “type-checking” a judgment without activeness annotations: first we use the recursive height function on terms to calculate the activeness, then we proceed as usual to recursively match against the generator or identity rules. Because of this theorem, in the future we will omit the activeness labels.

Theorem 4.6 can be seen as giving a “normal form” for morphisms in a free prop. Intuitively, a composite of generating morphisms and permutations is in normal form if each generator or permutation is applied “as soon as possible”; the inductive definition of judgments with activeness makes precise exactly what that means.

Another perspective on this involves string diagrams. From the latter perspective, a morphism in the free prop generated by \mathcal{G} can be represented by an *acyclic directed graph* whose vertices are labeled by generators and whose edges are labeled by objects. We allow “free edges” one or both of whose ends do not connect to any vertex, corresponding to the input and output objects of the morphism. For instance, such a graph based on generators $f \in \mathcal{G}(; G, C)$, $g \in \mathcal{G}(; D)$, $h \in \mathcal{G}(C, A; F, E)$, $k \in \mathcal{G}(B;)$, $\ell \in \mathcal{G}(E, D; H)$, and $m(G;)$ is shown in Figure 7a. The corresponding term in our type theory is

$$x : A, y : B \vdash (h_{(1)}(f_{(2)}, x), \ell(h_{(2)}(f_{(2)}, x), g) \mid m(f_{(1)}), k(y)) : (F, H)$$

Applying the algorithm of Theorem 4.6, we first calculate the heights of each generator:

$$f : 0 \quad g : 0 \quad h : 1 \quad k : 1 \quad \ell : 2 \quad m : 1$$

Therefore, H is the only active type, and the final rule application must have been a generator rule applying only ℓ :

$$\frac{x : A, y : B \vdash (h_{(2)}(f_{(2)}, x), g, h_{(1)}(f_{(2)}, x) \mid m(f_{(1)}), k(y)) : (E^*, D, F^*) \\ \ell \in \mathcal{G}(E, D; H)}{x : A, y : B \vdash (h_{(1)}(f_{(2)}, x), \ell(h_{(2)}(f_{(2)}, x), g) \mid m(f_{(1)}), k(y)) : (F, H^*)}. \quad (4.7)$$

In the premise of this rule, E and F are the active types since h has height 1. But k and m also have height 1, so they must also be applied by the generator rule leading to this judgment:

$$\frac{x : A, y : B \vdash (f_{(2)}, x, f_{(1)}, y, g) : (C^*, A^*, G^*, B^*, D^*) \\ h \in \mathcal{G}(C, A; F, E) \\ m \in \mathcal{G}(G;) \quad k \in \mathcal{G}(B;)}{x : A, y : B \vdash (h_{(2)}(f_{(2)}, x), g, h_{(1)}(f_{(2)}, x) \mid m(f_{(1)}), k(y)) : (E^*, D, F^*)}. \quad (4.8)$$

Now in the premise all the types are active and all terms have height 0, so it must have arisen from the identity rule:

$$\frac{f \in \mathcal{G}(; G, C) \quad g \in \mathcal{G}(; D)}{x : A, y : B \vdash (f_{(2)}, x, f_{(1)}, y, g) : (C^*, A^*, G^*, B^*, D^*)}. \quad (4.9)$$

Thus we have a complete typing derivation:

$$\frac{\frac{f \in \mathcal{G}(; G, C) \quad g \in \mathcal{G}(; D)}{x : A, y : B \vdash (f_{(2)}, x, f_{(1)}, y, g) : (C^*, A^*, G^*, B^*, D^*)} \quad h \in \mathcal{G}(C, A; F, E) \quad m \in \mathcal{G}(G;) \quad k \in \mathcal{G}(B;)}{\frac{x : A, y : B \vdash (h_{(2)}(f_{(2)}, x), g, h_{(1)}(f_{(2)}, x) \mid m(f_{(1)}), k(y)) : (E^*, D, F^*)}{x : A, y : B \vdash (h_{(1)}(f_{(2)}, x), \ell(h_{(2)}(f_{(2)}, x), g) \mid m(f_{(1)}), k(y)) : (F, H^*)}}.$$

This derivation corresponds to the redrawing of the string diagram shown in Figure 7b, in which the vertices appear in horizontal layers, with the edges always pointing down the page, and each vertex placed in the highest layer possible. The topmost layer with f and g corresponds to the identity rule application (4.9), the next layer with h, m, k corresponds to the generator rule application (4.8), and the final layer with ℓ corresponds to (4.7).

The referee has pointed out that this is essentially the “longest path layering” [EL89] of an acyclic directed graph (modulo a suitable correction to deal with vertexless half-edges). The proof of Theorem 4.6 can thus equivalently be interpreted as computing a normal form for a morphism in a free prop by expressing it as an acyclic directed graph and rearranging it according to its longest path layering, and also making certain canonical

$$\frac{\Gamma \vdash (\vec{M} \mid Z_1, \dots, Z_n) : \Delta \quad \rho \in S_n \quad \sigma : \mathfrak{A} \cong \mathfrak{A}}{\Gamma \vdash (\vec{M} \mid Z_1, \dots, Z_n) = (\vec{M}^\sigma \mid Z_{\rho 1}^\sigma, \dots, Z_{\rho n}^\sigma) : \Delta}$$

Figure 8: Rule of the axiom-free equality judgment

choices regarding the ordering of vertices along each layer. It is natural to expect that other canonical ways of drawing a directed graph might correspond to other normal forms and hence other type theories for free props, but we will not investigate this here.

It remains to consider the equality judgment. We don't have a traditional form of α -conversion since we have no bound variables as such, but as remarked in section 1 the labels $a \in \mathfrak{A}$ can sometimes be regarded as playing a similar role, and in particular the choice of concrete labels must not matter. We also need to impose invariance under permutation of the scalar terms. It may seem silly to have incorporated permutations in the scalar terms earlier and yet quotient out by that freedom now, but such an equality rule would be necessary even if we hadn't incorporated any permutations to start with. The paradigmatic case is when we have two nullary scalar generators $f : () \rightarrow ()$ and $g : () \rightarrow ()$, leading unavoidably to two distinct valid terms

$$\vdash (\mid f, g) : () \qquad \vdash (\mid g, f) : ()$$

that must be equal in a monoidal category (since the monoid of endomorphisms of the unit object is commutative). It is probably no coincidence that this is also the case where interesting things happen upon categorification.

Combining these two permutation rules, we obtain the rule shown in Figure 8. When we consider *presented* props in section 6, with equational theories, we will need the usual reflexivity, symmetry, transitivity, and congruence rules for equality, but with only this rule we can omit them since permutations are already a group. And since we have no type forming operations, there are no β or η rules.

This completes our **type theory for the free prop generated by \mathcal{G}** .

5. Constructing free props from type theory

We now proceed to show that our type theory has the structure of a prop, beginning with the admissibility of exchange on the right.

5.1. PROPOSITION. *If $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$ is derivable and ρ is a permutation of Δ , then $\Gamma \vdash (\rho \vec{M} \mid \vec{Z}) : \rho \Delta$ is also derivable. Moreover, this action is functorial.*

PROOF. This essentially follows from how we built the rules. If the derivation ends with the identity rule, then we can compose ρ with the specified permutation σ from that rule, and reorder the generators f, \dots, g in the rule according to the order that $\rho\sigma$ puts

them in. If the derivation ends with the generator rule, then we similarly compose ρ with σ , reorder the generators f, \dots, g , and inductively push the remaining part of the permutation (that acting on the non-active terms) into the main premise. Functoriality follows immediately. ■

For admissibility of composition/substitution, it seems helpful to first prove the admissibility of a single-generator rule. Note that we formulate it with the domain of the generator at the *end* of the given codomain context.

5.2. PROPOSITION. *If $\Gamma \vdash (\vec{M}, \vec{N} \mid \vec{Z}) : \Delta, \vec{A}$ is derivable and $f \in \mathcal{G}(\vec{A}; \vec{B})$, then $\Gamma \vdash (\vec{M}, f(\vec{N}) \mid \vec{Z}) : \Delta, \vec{B}$ is derivable. Moreover, if none of the types in \vec{A} are active in the given derivation, then all of the types in Δ that are active in the given derivation are still active in the result.*

PROOF. If any of the types in \vec{A} are active, we can simply apply the generator rule with f as the only generator. Otherwise, none of them were introduced by the final rule in the given derivation. If that final rule was the identity rule, then \vec{A} must be empty (since all types in the conclusion of the identity rule are active), so f has nullary domain and we can just add it to that application of the identity rule.

If the final rule in the given derivation was the generator rule, then \vec{A} must also appear at the end of its main premise. If none of the types in \vec{A} are active therein, then we can inductively apply f to that premise; by the second clause of the inductive hypothesis, this does not alter the activeness of the other types in the premise, so we can re-apply the generator rule. Finally, if at least one of the types in \vec{A} is active in the main premise, then we can add f to the generator rule, applying it alongside all the other generators, since it satisfies the condition that at least one of its arguments be active. (Technically, this may require us to first permute the consequent of the main premise so that \vec{A} appears before all the other non-inputs to the generator rule. This is not a problem for the induction since in this case we are not actually using the inductive hypothesis at all.) In all cases, the second claim of the lemma is obvious. ■

Now by combining Proposition 5.1 and Proposition 5.2, we can postcompose with a generator $f \in \mathcal{G}(\vec{A}; \vec{B})$ whose domain types \vec{A} appear anywhere in the consequent of a judgment $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$, in any order.

5.3. PROPOSITION. *Substitution is admissible: if $\Gamma \vdash (\vec{M} \mid \vec{Y}) : \Delta$ and $\Delta \vdash (\vec{N} \mid \vec{Z}) : \Phi$ are derivable, then so is $\Gamma \vdash (\vec{N}[\vec{M}/\Delta] \mid \vec{Z}[\vec{M}/\Delta], \vec{Y}) : \Phi$.*

PROOF. We induct on the derivation of $\Delta \vdash (\vec{N} \mid \vec{Z}) : \Phi$. If it comes from the identity rule, then we just have to compose $\Gamma \vdash (\vec{M} \mid \vec{Y}) : \Delta$ with some number of nullary-domain generators and permute its codomain; we do this one by one using Proposition 5.2 and then Proposition 5.1. Similarly, if it comes from the generator rule, we inductively compose with its main premise, then apply all of the new generators one by one using Proposition 5.2. ■

As an example, suppose we want to compose the following terms:

$$x : A, y : B \vdash (f_{(1)}(y), k(g, f_{(3)}(y)), f_{(2)}(y) \mid h(x)) : (C, D, E) \quad (5.4)$$

$$u : C, v : D, w : E \vdash (m(u, \ell_{(2)}(w)), s, \ell_{(1)}(w) \mid n(v)) : (F, G, H) \quad (5.5)$$

Here the generators are

$$\begin{array}{lllll} f : B \rightarrow (C, E, P) & g : () \rightarrow Q & h : A \rightarrow () & k : (Q, P) \rightarrow D & \ell : E \rightarrow (H, R) \\ m : (C, R) \rightarrow F & & n : D \rightarrow () & & s : () \rightarrow G \end{array}$$

The heights are

$$f = 1 \quad g = 0 \quad h = 1 \quad k = 2 \quad \ell = 1 \quad m = 2 \quad n = 1 \quad s = 0$$

Thus, the final rule of the second derivation must apply m only, so our inductive job is to compose (5.4) with

$$u : C, v : D, w : E \vdash (u, \ell_{(2)}(w), s, \ell_{(1)}(w) \mid n(v)) : (C, R, G, H) \quad (5.6)$$

Now the final rule of the second derivation must apply ℓ and n together, so our inductive job is to compose (5.4) with

$$u : C, v : D, w : E \vdash (w, v, u, s \mid) : (E, D, C, G) \quad (5.7)$$

The latter is obtained from the identity rule, so our task is now to apply Proposition 5.2 to the former and the single generator $s : () \rightarrow G$. Peeling down the derivation of the former, we obtain

$$x : A, y : B \vdash (g, f_{(3)}(y), f_{(1)}(y), f_{(2)}(y) \mid h(x)) : (Q, P, C, E)$$

and then

$$x : A, y : B \vdash (y, x, g \mid) : (B, A, Q)$$

which is also obtained from the identity rule. The identity rule can therefore also give us

$$x : A, y : B \vdash (y, x, g, s \mid) : (B, A, Q, G).$$

Re-applying f, h and then k , we obtain

$$x : A, y : B \vdash (g, f_{(3)}(y), f_{(1)}(y), f_{(2)}(y), s \mid h(x)) : (Q, P, C, E, G)$$

and then

$$x : A, y : B \vdash (f_{(1)}(y), k(g, f_{(3)}(y)), f_{(2)}(y), s \mid h(x)) : (C, D, E, G).$$

Permuting this, we obtain

$$x : A, y : B \vdash (f_{(2)}(y), f_{(1)}(y), k(g, f_{(3)}(y)), s \mid h(x)) : (E, C, D, G).$$

as the result of composing (5.4) and (5.7).

Backing out the induction one more step, we must apply ℓ and n to this using Proposition 5.2. We cannot apply ℓ directly since its domain E is not active (its term $f_{(2)}(y)$ has height 1 while the maximum height is 2). Thus, we back up to the main premise

$$x : A, y : B \vdash (g, f_{(3)}(y), f_{(1)}(y), f_{(2)}(y), s \mid h(x)) : (Q, P, C, E, G)$$

in which E is active. Thus, we can apply ℓ in the same generator rule as k , obtaining

$$x : A, y : B \vdash (\ell_{(1)}(f_{(2)}(y)), \ell_{(2)}(f_{(2)}(y)), f_{(1)}(y), k(g, f_{(3)}(y)), s \mid h(x)) : (H, R, C, D, G). \quad (5.8)$$

Now the domain D of the generator n is active, so we can directly apply it with another generator rule, obtaining (after permutation)

$$x : A, y : B \vdash (f_{(1)}(y), \ell_{(2)}(f_{(2)}(y)), s, \ell_{(1)}(f_{(2)}(y)) \mid n(k(g, f_{(3)}(y))), h(x)) : (C, R, G, H). \quad (5.9)$$

as the result of composing (5.4) and (5.6).

Finally, we must compose this with m using Proposition 5.2. Neither of the domain types C and R is active in (5.9) (in fact, no types are active in (5.9), since the last rule applied was a generator rule whose only generator has nullary codomain), so we have to inductively peel back to (5.8) in which R is active (though not C). Thus, we can then apply m in the same generator rule as n , obtaining

$$x : A, y : B \vdash (m(f_{(1)}(y), \ell_{(2)}(f_{(2)}(y))), s, \ell_{(1)}(f_{(2)}(y)) \mid n(k(g, f_{(3)}(y))), h(x)) : (F, G, H) \quad (5.10)$$

as our end result.

Note that the terms in (5.10) are indeed the result of substituting $f_{(1)}(y)$ for u , $k(g, f_{(3)}(y))$ for v , and $f_{(2)}(y)$ for w (the terms appearing in (5.4)) in the terms of (5.5), and appending the scalar term $h(x)$ of (5.4) to the scalar terms of (5.5):

$$\begin{aligned} m(u, \ell_{(2)}(w))[f_{(1)}(y)/u, k(g(f_{(3)}(y)))/v, f_{(2)}(y)/w] &= m(f_{(1)}(y), \ell_{(2)}(f_{(2)}(y))) \\ s[f_{(1)}(y)/u, k(g(f_{(3)}(y)))/v, f_{(2)}(y)/w] &= s \\ \ell_{(1)}(w)[f_{(1)}(y)/u, k(g(f_{(3)}(y)))/v, f_{(2)}(y)/w] &= \ell_{(1)}(f_{(2)}(y)) \\ n(v)[f_{(1)}(y)/u, k(g(f_{(3)}(y)))/v, f_{(2)}(y)/w] &= n(k(g(f_{(3)}(y)))) \end{aligned}$$

The only choice involved in the proof of Proposition 5.2 is in how to order the scalar terms in the result. We adopted the convention that those associated to the terms being substituted into come first, followed by those associated to the terms being substituted. The opposite convention would do as well for the following theorem:

5.11. PROPOSITION. *Composition is associative and unital.*

PROOF. Since derivations are determined uniquely by their terms by Theorem 4.6, this follows from the evident associativity and unitality of substitution into terms, and the associativity and unitality of concatenation of lists of scalar terms. ■

Thus we have a category whose objects are contexts and whose morphisms are derivable judgments $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$. However, this is not quite the underlying category of our prop: we must quotient it by the equality rule from Figure 8:

$$\frac{\Gamma \vdash (\vec{M} \mid Z_1, \dots, Z_n) : \Delta \quad \rho \in S_n \quad \sigma : \mathfrak{A} \cong \mathfrak{A}}{\Gamma \vdash (\vec{M} \mid Z_1, \dots, Z_n) = (\vec{M}^\sigma \mid Z_{\rho 1}^\sigma, \dots, Z_{\rho n}^\sigma) : \Delta} \quad (5.12)$$

(which also ensures that the choice of ordering in Proposition 5.3 is irrelevant). For this we need the evident observation:

5.13. PROPOSITION. *The equality rule (5.12) is a congruence on the category of contexts and derivable typing judgments. That is, it is an equivalence relation on the morphisms that is preserved by composition on both sides.* ■

Therefore, the quotient by this equality judgment is again a category whose objects are the contexts (i.e. finite lists of types).

5.14. THEOREM. *The contexts and derivable term judgments in the type theory for the free prop generated by \mathcal{G} , modulo the equality rule (5.12), form a symmetric strict monoidal category.*

PROOF. The monoidal structure on contexts is concatenation, with the empty context as unit. To tensor morphisms, it is easiest to first tensor with identities: given $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$, we construct $\Gamma, \vec{x} : \vec{A} \vdash (\vec{M}, \vec{x} \mid \vec{Z}) : \Delta, \vec{A}$ by inducting until we get down to the identity rule and then adding the variables $\vec{x} : \vec{A}$ to the context. Now we obtain the tensor product of $\Gamma \vdash (\vec{M} \mid \vec{Y}) : \Delta$ and $\Phi \vdash (\vec{N} \mid \vec{Z}) : \Psi$ by first tensoring with identities to get $\Gamma, \Phi \vdash (\vec{M}, \Gamma \mid \vec{Y}) : \Delta, \Phi$ and $\Delta, \Phi \vdash (\Delta, \vec{N} \mid \vec{Z}) : \Delta, \Psi$ and then composing to get $\Gamma, \Phi \vdash (\vec{M}, \vec{N} \mid \vec{Z}, \vec{Y}) : \Delta, \Psi$. If we did this in the other order, we would get $(\vec{M}, \vec{N} \mid \vec{Y}, \vec{Z})$ instead, which is equal by (5.12). In particular, this implies functoriality of the tensor product; associativity and unitality follow similarly. Finally, the symmetry isomorphism is $\vec{x} : \vec{A}, \vec{y} : \vec{B} \vdash (\vec{y}, \vec{x}) : \vec{B}, \vec{A}$; it is easy to verify the axioms. ■

Thus we have a prop, which we denote $\mathfrak{F}\mathcal{G}$.

5.15. THEOREM. *$\mathfrak{F}\mathcal{G}$ is the free prop generated by \mathcal{G} .*

PROOF. Let \mathcal{P} be a prop and $\omega : \mathcal{G} \rightarrow \mathcal{P}$ a morphism of signatures. We extend it to a morphism of props $\mathfrak{F}\mathcal{G} \rightarrow \mathcal{P}$ by induction on derivations. By the coherence theorem for symmetric monoidal categories (e.g. [ML98, Chapter XI]), there is a unique choice at each step if we are to have a (symmetric strict monoidal) functor, and likewise both equality rules corresponds to actual equalities that must hold in \mathcal{P} . Then we prove that this actually is a symmetric strict monoidal functor, using the definition of composition and the tensor product in $\mathfrak{F}\mathcal{G}$. ■

5.16. REMARK. Since the free prop generated by a signature is unique up to isomorphism, it follows that our $\mathfrak{F}\mathcal{G}$ is isomorphic to the free prop on \mathcal{G} presented in any other way, such as by using string diagrams whose edges and vertices are labeled by objects and morphisms of \mathcal{G} respectively. Unsurprisingly, this correspondence can be made more explicit: from any derivable $\Gamma \vdash \vec{M} : \Delta$ we can construct a labeled string diagrams from Γ to Δ , whose vertices are the generator applications and whose edges are the disjoint union of the variables in Γ and the terms $f_{(i)}(\vec{N})$ appearing as subterms of \vec{M} .

6. Presentations of props

Since the category of props is monadic over the category of signatures by Theorem 2.5, every prop \mathcal{P} has a *presentation* in terms of signatures, i.e. a coequalizer diagram

$$\mathfrak{F}\mathcal{R} \rightrightarrows \mathfrak{F}\mathcal{G} \rightarrow \mathcal{P}.$$

Moreover, since \mathfrak{F} and its right adjoint are the identity on the set of objects, we may assume that \mathcal{R} and \mathcal{G} both have the same set of objects as \mathcal{P} and all the morphisms are the identity on objects. Now by the universal property of $\mathfrak{F}\mathcal{R}$, the two morphisms of props $\mathfrak{F}\mathcal{R} \rightrightarrows \mathfrak{F}\mathcal{G}$ are equivalently morphisms of signatures $\mathcal{R} \rightrightarrows \mathfrak{F}\mathcal{G}$. Thus, once \mathcal{G} is given, the additional data of \mathcal{R} consists of a set of pairs of parallel morphisms in $\mathfrak{F}\mathcal{G}$, which is to say a set of **equality axioms** of the form

$$\Gamma \vdash (\vec{M} \mid \vec{Y}) = (\vec{N} \mid \vec{Z}) : \Delta$$

where both $\Gamma \vdash (\vec{M} \mid \vec{Y}) : \Delta$ and $\Gamma \vdash (\vec{N} \mid \vec{Z}) : \Delta$ are derivable in the type theory for the free prop generated by \mathcal{G} . We obtain the **type theory for the prop presented by $(\mathcal{G}, \mathcal{R})$** by augmenting the type theory for the free prop on \mathcal{G} by these axioms for the equality judgment, together with the additional rules shown in Figure 9 (which are no longer automatic in the presence of such axioms).

The first three rules in Figure 9 are the usual reflexivity,⁵ symmetry, and transitivity. The next three are congruence rules for precomposition, postcomposition, and the concatenation product. The final one is the label-renaming and scalar-permutation rule from Figure 8. Note that congruence for pre- and post-composition includes congruence under permutations of the domain and codomain.

6.1. PROPOSITION. *For any prop presentation $(\mathcal{G}, \mathcal{R})$, the equality judgment generated by the axioms of \mathcal{R} together with the rules of Figure 9 is a congruence on the prop $\mathfrak{F}\mathcal{G}$. That is, it is an equivalence relation on morphisms preserved by composition and tensor product.*

PROOF. The rules of Figure 9 essentially force this to be true. ■

⁵Actually, it is not necessary to assert reflexivity explicitly, since it is a special case of the permutation rule.

$$\begin{array}{c}
\frac{\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta}{\Gamma \vdash (\vec{M} \mid \vec{Z}) = (\vec{M} \mid \vec{Z}) : \Delta} \quad \frac{\Gamma \vdash (\vec{M} \mid \vec{Y}) = (\vec{N} \mid \vec{Z}) : \Delta}{\Gamma \vdash (\vec{N} \mid \vec{Z}) = (\vec{M} \mid \vec{Y}) : \Delta} \\
\\
\frac{\Gamma \vdash (\vec{M} \mid \vec{X}) = (\vec{N} \mid \vec{Y}) : \Delta \quad \Gamma \vdash (\vec{N} \mid \vec{Y}) = (\vec{P} \mid \vec{Z}) : \Delta}{\Gamma \vdash (\vec{M} \mid \vec{X}) = (\vec{P} \mid \vec{Z}) : \Delta} \\
\\
\frac{\Gamma \vdash (\vec{M} \mid \vec{X}) = (\vec{N} \mid \vec{Y}) : \Delta \quad \Delta \vdash (\vec{P} \mid \vec{Z}) : \Phi}{\Gamma \vdash (\vec{P}[\vec{M}/\Delta] \mid \vec{Z}[\vec{M}/\Delta], \vec{X}) = (\vec{P}[\vec{N}/\Delta] \mid \vec{Z}[\vec{N}/\Delta], \vec{Y}) : \Phi} \\
\\
\frac{\Gamma \vdash (\vec{M} \mid \vec{X}) : \Delta \quad \Delta \vdash (\vec{N} \mid \vec{Y}) = (\vec{P} \mid \vec{Z}) : \Phi}{\Gamma \vdash (\vec{N}[\vec{M}/\Delta] \mid \vec{Y}[\vec{M}/\Delta], \vec{X}) = (\vec{P}[\vec{M}/\Delta] \mid \vec{Z}[\vec{M}/\Delta], \vec{X}) : \Phi} \\
\\
\frac{\Gamma \vdash (\vec{M} \mid \vec{X}) = (\vec{N} \mid \vec{Y}) : \Delta \quad \Phi \vdash (\vec{P} \mid \vec{Z}) = (\vec{Q} \mid \vec{W}) : \Psi}{\Gamma, \Phi \vdash (\vec{M}, \vec{P} \mid \vec{X}, \vec{Z}) = (\vec{N}, \vec{Q} \mid \vec{Y}, \vec{W}) : \Delta, \Psi} \\
\\
\frac{\Gamma \vdash (\vec{M} \mid Z_1, \dots, Z_n) : \Delta \quad \rho \in S_n \quad \sigma : \mathfrak{A} \cong \mathfrak{A}}{\Gamma \vdash (\vec{M} \mid Z_1, \dots, Z_n) = (\vec{M}^\sigma \mid Z_{\rho 1}^\sigma, \dots, Z_{\rho n}^\sigma) : \Delta}
\end{array}$$

Figure 9: Rules of the equality judgment in the presence of axioms

Thus we obtain a prop $\mathfrak{F}\langle\mathcal{G}|\mathcal{R}\rangle$ as the quotient of $\mathfrak{F}\mathcal{G}$ by this congruence.

6.2. THEOREM. *$\mathfrak{F}\langle\mathcal{G}|\mathcal{R}\rangle$ is the prop presented by $(\mathcal{G}, \mathcal{R})$. That is, we have a coequalizer diagram in the category of props:*

$$\mathfrak{F}\mathcal{R} \rightrightarrows \mathfrak{F}\mathcal{G} \rightarrow \mathfrak{F}\langle\mathcal{G}|\mathcal{R}\rangle.$$

PROOF. Since the quotient map $\mathfrak{F}\mathcal{G} \rightarrow \mathfrak{F}\langle\mathcal{G}|\mathcal{R}\rangle$ is surjective, a prop morphism $\omega : \mathfrak{F}\mathcal{G} \rightarrow \mathcal{P}$ factors through $\mathfrak{F}\langle\mathcal{G}|\mathcal{R}\rangle$ in at most one way. Moreover, it does so precisely when it identifies all pairs of tuples of terms that are identified by the equality judgment. However, the equality rules in Figure 9 are satisfied in any prop, so this happens precisely when ω respects the axioms of \mathcal{R} , i.e. when it coequalizes the two maps $\mathfrak{F}\mathcal{R} \rightrightarrows \mathfrak{F}\mathcal{G}$. ■

Thus, we can use our type theory to reason about structures in arbitrary props defined by generators and axioms, and hence also in symmetric monoidal categories (by Theorem 2.3).

7. Examples

7.1. DUALS AND TRACES. We begin by repeating the first example from the introduction more carefully. The **free prop generated by a dual pair** has a generating signature \mathcal{G} with two objects A and A^* and two morphisms $\eta : () \rightarrow (A, A^*)$ and $\varepsilon : (A^*, A) \rightarrow ()$, and a signature of relations \mathcal{R} that imposes two axioms

$$x : A \vdash (\eta_{(1)} \mid \varepsilon(\eta_{(2)}, x)) = x : A \quad y : A^* \vdash (\eta_{(2)} \mid \varepsilon(y, \eta_{(1)})) = y : A^*.$$

A map from this prop to a symmetric monoidal category then reduces to the usual notion of dual.

As suggested in the introduction, we write $M \triangleleft N$ for $\varepsilon(M, N)$, and $(u, \lambda^A u) : (A, A^*)$ for $(\eta_{(1)}, \eta_{(2)})$, where u is a label (i.e. an element of \mathfrak{A}) rather than a variable (appearing in the context). In this notation, the axioms are

$$x : A \vdash (u \mid \lambda^A u \triangleleft x) = x : A \quad y : A^* \vdash (\lambda^A u \mid y \triangleleft u) = y : A^*.$$

Recall that $=$ is a congruence for substitution on both sides. Thus the first axiom means that if $\lambda^A u \triangleleft M$ appears in the scalars, for *any term* $M : A$ not involving u , then it can be removed by replacing u (wherever it appears, even as a subterm of some other term) with M . This justifies regarding it as a sort of “ β -reduction for duality” with u playing the role of a “bound variable”, although as we noted in the introduction the “binder” $\lambda^A u$ does not delimit the “scope” of u at all. Running this rule in reverse, we see that any term $M : A$ (appearing even as a sub-term of some other term) can be replaced by u , for a fresh label u , if we simultaneously add $\lambda^A u \triangleleft M$ to the scalars. The other axiom is similar; we may regard it as an “ η -reduction for duality”.

If A has a dual A^* , and $f : A \rightarrow A$, the **trace** of f is the composite

$$() \xrightarrow{\eta} (A, A^*) \xrightarrow{(f, \text{id})} (A, A^*) \xrightarrow{\cong} (A^*, A) \xrightarrow{\varepsilon} ()$$

In our type theory this is

$$() \vdash (|\lambda^A u \triangleleft f(u)) : () .$$

As advertised in the introduction, we can now prove cyclicity of the trace with a β -expansion followed by a β -reduction: for morphisms $f : A \rightarrow B$ and $g : B \rightarrow A$, with A and B dualizable, we have

$$\begin{aligned} \text{tr}(fg) &\stackrel{\text{def}}{=} (|\lambda^B y \triangleleft f(g(y))) \\ &= (|\lambda^B y \triangleleft f(x), \lambda^A x \triangleleft g(y)) \\ &= (|\lambda^A x \triangleleft g(f(x))) \\ &\stackrel{\text{def}}{=} \text{tr}(gf). \end{aligned}$$

More generally, any $f : (Y, A) \rightarrow (Z, A)$ has a “partial” or “twisted” trace

$$y : Y \vdash (f_{(1)}(y, u) \mid \lambda^A u \triangleleft f_{(2)}(y, u)) : Z.$$

This satisfies a version of cyclicity [PS14, Lemma 4.4]: for any morphisms $f : (Y, A) \rightarrow (Z, B)$ and $g : (W, B) \rightarrow (X, A)$, with A and B dualizable, we have

$$\begin{aligned} y : Y, w : W \vdash (g_{(1)}(w, v), f_{(1)}(y, g_{(2)}(w, v)) \mid \lambda^B v \triangleleft f_{(2)}(y, g_{(2)}(w, v))) \\ &= (g_{(1)}(w, v), f_{(1)}(y, u) \mid \lambda^B v \triangleleft f_{(2)}(y, u), \lambda^A u \triangleleft g_{(2)}(w, v)) \\ &= (g_{(1)}(w, f_{(2)}(y, u)), f_{(1)}(y, u) \mid \lambda^A u \triangleleft g_{(2)}(w, f_{(2)}(y, u))) \\ &\quad : (X, Z). \end{aligned}$$

This general cyclicity includes in particular the *sliding* axiom for traces from [JSV96]. Their other axioms become simply syntactic identities in our type theory. For instance, *tightening* is the statement that if we compose $f : (Y, A) \rightarrow (Z, A)$ with $u : X \rightarrow Y$ and $v : Z \rightarrow W$ and then take its trace:

$$\frac{\begin{array}{c} x : X \vdash u(x) : Y \quad y : Y, a : A \vdash (f_{(1)}(y, a), f_{(2)}(y, a)) : (Z, A) \quad z : Z \vdash v(z) : W \\ \hline x : X, a : A \vdash (v(f_{(1)}(u(x), a)), f_{(2)}(u(x), a)) : (W, A) \end{array}}{x : A \vdash (v(f_{(1)}(u(x), a)) \mid \lambda^A a \triangleleft f_{(2)}(u(x), a)) : W}$$

we get the same result as if we first take the trace of f and then compose with u and v :

$$\frac{\begin{array}{c} y : Y, a : A \vdash (f_{(1)}(y, a), f_{(2)}(y, a)) : (Z, A) \\ \hline y : Y \vdash (f_{(1)}(y, a) \mid \lambda^A a \triangleleft f_{(2)}(y, a)) : Z \quad z : Z \vdash v(z) : W \\ \hline x : A \vdash (v(f_{(1)}(u(x), a)) \mid \lambda^A a \triangleleft f_{(2)}(u(x), a)) : W. \end{array}}{x : X \vdash u(x) : Y}$$

Since the terms concluding both derivations are the same, they represent the same morphism. Of course, these are not actually derivations in our type theory, since they use the admissible rule of substitution. The uniqueness of typing derivations means that if the substitutions are eliminated according to Proposition 5.3 we obtain the same result in both cases, which in this case is:

$$\frac{\overline{x : A \vdash (u(x), a, \lambda^A a) : (Y, A, A^*)}}{x : A \vdash (f_{(1)}(u(x), a), \lambda^A a, f_{(2)}(u(x), a)) : (Z, A^*, A)} \frac{}{x : A \vdash (v(f_{(1)}(u(x), a)) \mid \lambda^A a \triangleleft f_{(2)}(u(x), a)) : W.}$$

7.2. REMARK. Given any signature \mathcal{G} , we can augment it by adding a specified dual A^* for each object $A \in \mathcal{G}$ along with duality data as above. This yields a new signature \mathcal{G}^* such that $\mathfrak{F}\mathcal{G}^*$ is freely generated by \mathcal{G} together with a specified dual for each of its objects. Thus $\mathfrak{F}\mathcal{G}^*$ is the free **compact closed prop** (i.e. prop in which every object has a dual) generated by \mathcal{G} . Any signature \mathcal{R} of relations for \mathcal{G} carries over to \mathcal{G}^* as well, so we can construct presented compact closed props as well, and thereby presented compact closed monoidal categories.

In general, a prop presentation may not have any decision procedure for equality or normal forms for morphisms. However, the view of the zigzag equalities as “ β and η reductions” suggests that this should be the case for some classes of presented compact closed props (whenever the equalities other than the zigzag identities can be controlled). If true, this should include in particular the explicit description of the free compact closed monoidal category on an ordinary category from [KL80].

Note also that by the “Int-construction” [JSV96], any *traced* symmetric monoidal category embeds fully-faithfully and trace-preservingly in a compact closed one. Thus, we can also use presented compact closed props to reason about traced symmetric monoidal categories.

7.3. WELL-IDEMPOTENT DUALIZABLE OBJECTS ARE SELF-DUAL. Tim Campion asked in [Cam17] whether an idempotent dualizable object in a symmetric monoidal category must be self-dual. A proof using string diagrams that this holds assuming “well-idempotence” was given by the user “MTyson”; here we recast this proof in our type theory.

We assume given one type X with a dual X^* , expressed as before, and a morphism $i : () \rightarrow X$ such that $1_X \otimes i : X \rightarrow X \otimes X$ is an isomorphism (such an i is what makes X *well-idempotent*). In our type theory, the latter can be expressed by a term $\vdash i : X$ and a morphism $x : X, y : X \vdash f(x, y) : X$ (the inverse to $1_X \otimes i$) such that $f(x, i) = x$ and $(x, y) = (f(x, y), i)$. More precisely, the latter two equalities are

$$\begin{aligned} x : X &\vdash f(x, i) = x : X \\ x : X, y : X &\vdash (x, y) = (f(x, y), i) : (X, X) \end{aligned}$$

but we tend to omit the contexts and types in equality axioms and calculations when they are obvious.

Note that the equation $f(x, i) = x$ means that i is a “right unit” for the “binary operation” f . We now observe that it is also a left unit:

$$\begin{aligned} f(i, y) &= (f(u, y) \mid \lambda^X u \triangleleft i) \\ &= (u \mid \lambda^X u \triangleleft y) \\ &= y. \end{aligned}$$

Here the first line is a β -expansion and the third is a β -reduction. The second line uses the equality $(x, y) = (f(x, y), i)$ in the following way: first we introduce an extra variable

to get

$$x : X, w : X^*, y : X \vdash (x, w, y) = (f(x, y), w, i) : (X, X^*, X)$$

then we precompose with

$$y : X \vdash (u, \lambda^X u, y) : (X, X^*, X)$$

to get

$$y : X \vdash (u, \lambda^X u, y) = (f(u, y), \lambda^X u, i) : (X, X^*, X)$$

and then we post-compose with

$$x : X, w : X^*, y : X \vdash (x \mid w \triangleleft y) : X$$

to get

$$y : X \vdash (u \mid \lambda^X u \triangleleft y) = (f(u, y) \mid \lambda^X u \triangleleft i) : X.$$

Note that although the type-theoretic justification is a bit complicated, at the level of terms the operation is intuitive: we simply simultaneously substitute u for $f(u, y)$ and y for i , wherever the latter appear as subterms. From now on we will perform such substitutions at term-level without further comment.

Now we define $x : X \vdash \phi(x) : X^*$ and $w : X^* \vdash \psi(w) : X$ by

$$\begin{aligned} \phi(x) &\stackrel{\text{def}}{=} (\lambda^X v \mid \lambda^X u \triangleleft f(v, f(x, u))) \\ \psi(w) &\stackrel{\text{def}}{=} (i \mid w \triangleleft i). \end{aligned}$$

Finally, we can show that ϕ and ψ are inverse isomorphisms (so that X is isomorphic to its dual) with the following computations:

$$\begin{aligned} \psi(\phi(x)) &= (i \mid \lambda^X v \triangleleft i, \lambda^X u \triangleleft f(v, f(x, u))) \\ &= (i \mid \lambda^X u \triangleleft f(i, f(x, u))) && (\beta\text{-reduction for } v) \\ &= (i \mid \lambda^X u \triangleleft f(x, u)) && (i \text{ is a left unit for } f) \\ &= (u \mid \lambda^X u \triangleleft x) && ((x, u) = (f(x, u), i)) \\ &= x && (\beta\text{-reduction for } u) \\ \phi(\psi(w)) &= (\lambda^X v \mid \lambda^X u \triangleleft f(v, f(i, u)), w \triangleleft i) \\ &= (\lambda^X v \mid \lambda^X u \triangleleft f(v, u), w \triangleleft i) && (i \text{ is a left unit for } f) \\ &= (\lambda^X v \mid \lambda^X u \triangleleft v, w \triangleleft u) && ((v, u) = (f(v, u), i)) \\ &= (\lambda^X v \mid w \triangleleft v) && (\beta\text{-reduction for } u) \\ &= w && (\eta\text{-reduction for } v). \end{aligned}$$

We do not reproduce MTyson's string diagram proof here, but the reader is encouraged to compare it to our type-theoretic version. Note that this situation is partly "topological" in the sense of subsection 1.10 (the zigzag axioms for duality, and arguably the unit properties $f(x, i) = x$ and $f(i, x) = x$) and partly non-topological (the axiom $(x, y) = (f(x, y), i)$).

7.4. COMONOIDS AND SWEEDLER NOTATION. As in ordinary cartesian (or even linear) type theory, it is easy to use our type theory to define **monoid objects** in a symmetric monoidal category. The signature has one object M and two morphisms $m : (M, M) \rightarrow M$ and $e : () \rightarrow M$; we usually write $m(x, y)$ infix as $x \cdot y$ or just xy . The axioms are the expected

$$(xy)z = x(yz) \quad xe = x \quad ex = x.$$

However, with our type theory we can also define **comonoid objects**, which have instead two morphisms $\Delta : M \rightarrow (M, M)$ and $\varepsilon : M \rightarrow ()$, and axioms

$$\begin{aligned} (\Delta_{(1)}(\Delta_{(1)}(x)), \Delta_{(2)}(\Delta_{(1)}(x)), \Delta_{(2)}(x)) &= (\Delta_{(1)}(x), \Delta_{(1)}(\Delta_{(2)}(x)), \Delta_{(2)}(\Delta_{(2)}(x))) \\ (\Delta_{(1)}(x) \mid \varepsilon(\Delta_{(2)}(x))) &\qquad\qquad\qquad (\Delta_{(2)}(x) \mid \varepsilon(\Delta_{(1)}(x))). \end{aligned}$$

As suggested in the introduction, this becomes much more manageable if we adopt the convention of traditional *Sweedler notation* for comonoids and comodules:

$$(x_{(1)}, x_{(2)}) \stackrel{\text{def}}{=} (\Delta_{(1)}(x), \Delta_{(2)}(x)).$$

Since there is no other meaning of $Z_{(i)}$ when Z is itself already a term, it is unambiguous to regard it as meaning $\Delta_{(i)}(Z)$ as long as no type has more than one relevant comultiplication. (We could formalize this with a more complicated type-checking algorithm, but we will be content to regard it as an informal abuse of notation.) We may regard this as a sort of “dual” to the shorthand notation “ xy ” for multiplication in a monoid, which omits the name or symbol for the product $(M, M) \rightarrow M$. With this notation, the axioms of a comonoid become

$$\begin{aligned} (x_{(1)(1)}, x_{(1)(2)}, x_{(2)}) &= (x_{(1)}, x_{(2)(1)}, x_{(2)(2)}) \\ (x_{(1)} \mid \varepsilon(x_{(2)})) &= x \qquad\qquad\qquad (x_{(2)} \mid \varepsilon(x_{(1)})) = x. \end{aligned}$$

Traditional Sweedler notation also goes one step further: in view of the coassociativity axiom, it is unambiguous to write $(x_{(1)}, x_{(2)}, x_{(3)})$ for either $(x_{(1)(1)}, x_{(1)(2)}, x_{(2)})$ or $(x_{(1)}, x_{(2)(1)}, x_{(2)(2)})$. In general, if subscripts are applied to a variable or a term that is already subscripted,

with the maximum such subscript being n , then the subscript ${}_{(k)}$ means $\overbrace{{}_{(2)} {}_{(2)} \cdots {}_{(2)}}^{k-1} {}_{(1)}$ if $k < n$ and $\overbrace{{}_{(2)} {}_{(2)} \cdots {}_{(2)}}^{n-1}$ if $k = n$.

Intuitively, in cartesian type theory (i.e. in a cartesian monoidal category), everything can be duplicated and discarded with impunity; whereas a comonoid in a non-cartesian monoidal category is equipped with *specified ways* in which to duplicate and discard elements. (Indeed, a cartesian monoidal category is precisely a symmetric monoidal category in which every object is equipped with a commutative comonoid structure in a natural way.) We thus view $x_{(1)}$ and $x_{(2)}$ as “duplicated copies” of x , the subscripts tracking the

order of duplication. Similarly, we can regard $\varepsilon(x)$ as “discarding” the element x , which inspires us to introduce a sort of “nullary Sweedler notation”

$$\mathcal{X} \stackrel{\text{def}}{=} \varepsilon(x).$$

Thus, for instance, the counit axioms of a comonoid become $(x_{(1)} \mid \underline{x_{(2)}}) = x$ and $(x_{(2)} \mid \underline{x_{(1)}}) = x$. Note that by coassociativity, we also have equalities such as $(x_{(1)}, x_{(2)} \mid \underline{x_{(3)}}) = (x_{(1)}, x_{(2)})$ and $(x_{(1)}, x_{(3)} \mid \underline{x_{(2)}}) = (x_{(1)}, x_{(2)})$ and so on.

Such shorthands need not be restricted to comonoids either. For instance, traditional Sweedler notation is also used for *comodules*, which have a coaction $D \rightarrow (C, D)$ by a coalgebra C . As an example with even greater generality, suppose M is dualizable and has a coaction $\Delta : M \rightarrow (A, M)$, satisfying no axioms at all, and that $f : M \rightarrow M$ is an endomorphism that respects Δ in that $(f(x_{(1)}), f(x_{(2)})) = (f(x)_{(1)}, f(x)_{(2)})$. Then we can use this notation to verify the *fixed-point property* of traces from [PS14, Corollary 5.3]:

$$\begin{aligned} (f(f(u)_{(1)}) \mid \lambda^M u \triangleleft f(u)_{(2)}) &= (f(v_{(1)}) \mid \lambda^M u \triangleleft v_{(2)}, \lambda^M v \triangleleft f(u)) \\ &= (f(v_{(1)}) \mid \lambda^M v \triangleleft f(v_{(2)})) \\ &= (f(v)_{(1)} \mid \lambda^M v \triangleleft f(v)_{(2)}) : A. \end{aligned}$$

7.5. FROBENIUS MONOIDS. A **Frobenius monoid** is an object that is both a monoid and a comonoid and satisfies the additional axiom

$$x : M, y : M \vdash (x_{(1)}, x_{(2)}y) = (xy_{(1)}, y_{(2)}). \quad (7.6)$$

Usually this is stated as two axioms saying that both sides of the above equation equal $((xy)_{(1)}, (xy)_{(2)})$. But this follows from the above axiom by the following argument, which I learned from [PS09]:

$$\begin{aligned} (x_{(1)}, x_{(2)}y) &= (x_{(1)}, x_{(2)}y_{(1)} \mid \underline{y_{(2)}}) && \text{(by counitality)} \\ &= (x_{(1)}, x_{(2)(1)} \mid \underline{x_{(2)(2)}y}) && \text{(by (7.6))} \\ &= (x_{(1)(1)}, x_{(1)(2)} \mid \underline{x_{(2)}y}) && \text{(by coassociativity)} \\ &= ((xy_{(1)})_{(1)}, (xy_{(1)})_{(2)} \mid \underline{y_{(2)}}) && \text{(by (7.6))} \\ &= ((xy)_{(1)}, (xy)_{(2)}) && \text{(by counitality).} \end{aligned}$$

The Frobenius axiom(s) are “topological”, so their string diagrams get a good deal of leverage from topological intuition. Thus, Frobenius monoids are not a very good example for the relative usefulness of type theory. However, for purposes of comparison, we include a proof of one of the basic facts about Frobenius monoids; namely that they are self-dual, with unit and counit:

$$\begin{aligned} (w \triangleleft x) &\stackrel{\text{def}}{=} (\mid \underline{wx}) \\ (u, \lambda^M u) &\stackrel{\text{def}}{=} (e_{(1)}, e_{(2)}). \end{aligned}$$

The axioms of a dual pair follow quite easily:

$$(u \mid \lambda^M u \triangleleft x) \stackrel{\text{def}}{=} (e_{(1)} \mid \underline{e_{(2)}x}) = (ex_{(1)} \mid \underline{x_{(2)}}) = ex = x$$

$$(\lambda^M u \mid w \triangleleft u) \stackrel{\text{def}}{=} (e_{(2)} \mid \underline{we_{(1)}}) = (w_{(2)}e \mid \underline{w_{(1)}}) = we = w.$$

7.7. REMARK. A **hypergraph category** [Kis14] is a symmetric monoidal category in which every object is equipped with a Frobenius monoid structure that is commutative ($xy = yx$), cocommutative ($(x_{(1)}, x_{(2)}) = (x_{(2)}, x_{(1)})$), and special a.k.a. separable ($x_{(1)}x_{(2)} = x$), and such that the Frobenius monoid structure on any tensor product $X \otimes Y$ is induced from those on X and Y in the standard way. The definition of **hypergraph prop** is a bit simpler: it is just a prop in which every object is equipped with a commutative, cocommutative, special Frobenius monoid structure. Since tensor products in a prop are only formal, the final condition is essentially automatic. More specifically, the final condition on a hypergraph category \mathcal{C} is not needed to show that it has an underlying hypergraph prop UC , but it is precisely what is needed to show that \mathcal{C} is equivalent, as a hypergraph category, to the free symmetric monoidal category generated by UC . (One might even argue that for this reason, hypergraph props are a more natural structure than hypergraph categories.)

Now given any signature \mathcal{G} , we can augment it by adding a commutative, cocommutative, special Frobenius monoid structure on every object. This yields a new signature \mathcal{G}^{hy} such that $\mathfrak{F}\mathcal{G}^{\text{hy}}$ is the free hypergraph prop generated by \mathcal{G} . Any signature \mathcal{R} of relations for \mathcal{G} carries over to \mathcal{G}^{hy} as well, so we can construct presented hypergraph props as well.

7.8. HOPF MONOIDS AND ANTIPODES. A monoid object in a cartesian monoidal category is also (like every object) a comonoid, but the monoid and comonoid structures do not satisfy the Frobenius axiom. Instead they satisfy the **bimonoid axioms**:

$$\begin{aligned} x : M, y : M &\vdash (x_{(1)}y_{(1)}, x_{(2)}y_{(2)}) = ((xy)_{(1)}, (xy)_{(2)}) : (M, M) \\ &\vdash (e_{(1)}, e_{(2)}) = (e, e) : (M, M) \\ x : M, y : M &\vdash (| xy) = (| x, y) : () \\ &\vdash (| \emptyset) = () : (). \end{aligned}$$

Thus, a bimonoid object in an arbitrary monoidal category can be regarded as a “non-cartesian” version of a monoid object in a cartesian monoidal category. Indeed, if a bimonoid is cocommutative ($x_{(1)}, x_{(2)} = (x_{(2)}, x_{(1)})$) then it is a monoid object in the cartesian monoidal category of cocommutative comonoids, and dually if it is commutative ($xy = yx$) then it is a monoid object in the opposite of the cocartesian monoidal category of commutative monoids. But in the non-commutative, non-cocommutative case we obtain something truly new.

The analogue for a bimonoid of the inversion operation, making a monoid into a group, is called an **antipode**: an operation $x : M \vdash \bar{x} : M$ such that

$$\begin{aligned} x : M &\vdash x_{(1)} \overline{x_{(2)}} = (e | x) : M \\ x : M &\vdash \overline{x_{(1)}} x_{(2)} = (e | x) : M. \end{aligned}$$

A bimonoid equipped with an antipode (a non-cartesian analogue of a group object) is called a **Hopf monoid**. Note that the comonoid structure is necessary in order to even formulate the antipode axioms: we need to duplicate x in order to invert one copy of it,

and on the other side of the equation we need to discard x in order to write simply “ e ”. In a cartesian monoidal category, Hopf monoids are precisely group objects in the usual sense. However, note also that bimonoids and Hopf monoids in a symmetric monoidal category are self-dual: such a structure on $M \in \mathcal{C}$ is equivalent to such a structure on $M \in \mathcal{C}^{\text{op}}$.

As mentioned in the introduction, cartesian type theory can internalize the basic fact of group theory that inverses in any monoid are unique: if \bar{x} and \hat{x} are both inverses of x the

$$\bar{x} = \bar{x}e = \bar{x}(x\hat{x}) = (\bar{x}x)\hat{x} = e\hat{x} = \hat{x}.$$

Therefore, a monoid object in any cartesian monoidal category admits at most one inverse, and hence both *cocommutative* Hopf monoids and *commutative* ones have unique antipodes. Cartesian type theory has nothing to say about Hopf monoids that are neither commutative nor cocommutative, but in our type theory we can reproduce essentially the same argument: if $x : M \vdash \bar{x} : M$ and $x : M \vdash \hat{x} : M$ are both antipodes, we compute

$$\bar{x} = \bar{x}e = (\overline{x_{(1)}}e \mid \cancel{x_{(2)}}) = \overline{x_{(1)}}x_{(21)}\widehat{x_{(22)}} = (e\widehat{x_{(2)}} \mid \cancel{x_{(1)}}) = e\widehat{x} = \hat{x}.$$

Thus, even in a non-cartesian situation we can use a very similar set-like argument, as long as we keep track of where elements get “duplicated and discarded”. I encourage the reader to write out a proof of this fact using traditional arrow notation or string diagrams for comparison.

As another example, if H and K are bimonoids, a **bimonoid homomorphism** is a morphism $x : H \vdash f(x) : K$ such that

$$\begin{aligned} f(xy) &= f(x)f(y) & (f(x)_{(1)}, f(x)_{(2)}) &= (f(x_{(1)}), f(x_{(2)})) \\ f(e) &= e & (| \cancel{f(x)}) &= (| \cancel{x}). \end{aligned}$$

Now we can show that when H and K are Hopf monoids, any such bimonoid homomorphism preserves antipodes.

$$\begin{aligned} f(\bar{x}) &= (f(\overline{x_{(1)}}) \mid \cancel{x_{(2)}}) \\ &= (f(\overline{x_{(1)}}) \mid \cancel{f(x_{(2)})}) \\ &= (f(\overline{x_{(1)}})e \mid \cancel{f(x_{(2)})}) \\ &= f(\overline{x_{(1)}})f(x_{(2)})_{(1)}\overline{f(x_{(2)})_{(2)}} \\ &= f(\overline{x_{(1)}})f(x_{(2)(1)})\overline{f(x_{(2)(2)})} \\ &= f(\overline{x_{(1)(1)}})f(x_{(1)(2)})\overline{f(x_{(2)})} \\ &= f(\overline{x_{(1)(1)}}x_{(1)(2)})\overline{f(x_{(2)})} \\ &= (f(e)\overline{f(x_{(2)})} \mid \cancel{x_{(1)}}) \\ &= (e\overline{f(x_{(2)})} \mid \cancel{x_{(1)}}) \\ &= (\overline{f(x_{(2)})} \mid \cancel{x_{(1)}}) \\ &= \overline{f(x)}. \end{aligned}$$

There is a more general approach to results of this sort, which we will return to in the next section.

7.9. WEAK BIMONOIDS. A **weak bimonoid** [PS09]⁶ is a monoid and comonoid that satisfies, instead of the bimonoid axioms, the following weakened ones:

$$\begin{aligned} ((xy)_{(1)}, (xy)_{(2)}) &= (x_{(1)}y_{(1)}, x_{(2)}y_{(2)}) \\ (\mid \cancel{xyz}) &= (\mid \cancel{xy_{(1)}}y_{(2)}z) \\ &= (\mid \cancel{xy_{(2)}}y_{(1)}z) \\ (e_{(1)}, e_{(2)}, e_{(3)}) &= (e_{(1)}, e_{(2)}e'_{(1)}, e'_{(2)}) \\ &= (e_{(1)}, e'_{(1)}e_{(2)}, e'_{(2)}). \end{aligned}$$

For a weak bimonoid we define

$$\begin{aligned} s(x) &\stackrel{\text{def}}{=} (e_{(1)} \mid \cancel{e_{(2)}x}) \\ t(x) &\stackrel{\text{def}}{=} (e_{(1)} \mid \cancel{xe_{(2)}}) \\ r(x) &\stackrel{\text{def}}{=} (e_{(2)} \mid \cancel{e_{(1)}x}). \end{aligned}$$

Many equations relating the weak bimonoid structure and the operations s, t, r are proven in [PS09] using string diagrams. All of them can also be proven in our type theory. For instance, here is a version of [PS09, Appendix B, eq. (1)]:

$$\begin{aligned} (s(x)_{(1)}, s(x)_{(2)}) &= (e_{(1)(1)}, e_{(1)(2)} \mid \cancel{e_{(2)}x}) \\ &= (e_{(1)}, e_{(2)} \mid \cancel{e_{(3)}x}) \\ &= (e_{(1)}, e_{(2)}e'_{(1)} \mid \cancel{e'_{(2)}x}) \\ &= (e_{(1)}, e_{(2)}s(x)). \end{aligned}$$

Here is a version of [PS09, Appendix B, eq. (4)]:

$$\begin{aligned} ((x s(y))_{(1)}, (x s(y))_{(2)}) &= (x_{(1)}s(y)_{(1)}, x_{(2)}s(y)_{(2)}) \\ &= (x_{(1)}e_{(1)}, x_{(2)}e_{(2)}s(y)) \quad (\text{using (1)}) \\ &= ((xe)_{(1)}, (xe)_{(2)}s(y)) \\ &= (x_{(1)}, x_{(2)}s(y)). \end{aligned}$$

And here is a version of [PS09, Appendix B, eq. (13)].

$$\begin{aligned} t(x)r(y) &= (e_{(1)}e'_{(2)} \mid \cancel{xe_{(2)}}, \cancel{e'_{(1)}}y) \\ &= (e_{(2)} \mid \cancel{xe_{(3)}}, \cancel{e_{(1)}y}) \\ &= (e_{(2)}e'_{(1)} \mid \cancel{xe'_{(2)}}, \cancel{e_{(1)}y}) \\ &= r(y)t(x). \end{aligned}$$

⁶In [PS09] these definitions are given in the additional generality of a *braided* monoidal category, but our type theory only applies to symmetric monoidal categories. This is sufficient to include a number of examples, however, such as the category algebra of a category with finitely many objects.

As an example of a somewhat longer proof, here is a version of [PS09, Appendix B, eq. (11)]. Unlike the previous examples, this is not a verbatim translation of their proof; theirs builds on previous lemmas, whereas the below proof is direct.

$$\begin{aligned}
(t(x_{(1)}), x_{(2)}y) &= (e_{(1)}, x_{(2)}y \mid \underline{x_{(1)}e_{(2)}}) \\
&= (e_{(1)}, (xe')_{(2)}y \mid \underline{(xe')_{(1)}e_{(2)}}) \\
&= (e_{(1)}, x_{(2)}e'_{(2)}y \mid \underline{x_{(1)}e'_{(1)}e_{(2)}}) \\
&= (e_{(1)}, x_{(2)}e_{(3)}y \mid \underline{x_{(1)}e_{(2)}}) \\
&= (e_{(1)}, x_{(2)}e_{(2)(2)}y \mid \underline{x_{(1)}e_{(2)(1)}}) \\
&= (e_{(1)}, (xe_{(2)})_{(2)}y \mid \underline{(xe_{(2)})_{(1)}}) \\
&= (e_{(1)}, xe_{(2)}y) \\
&= (e_{(1)}, x(e_{(2)}y)_{(2)} \mid \underline{(e_{(2)}y)_{(1)}}) \\
&= (e_{(1)}, xe_{(2)(2)}y_{(2)} \mid \underline{e_{(2)(1)}y_{(1)}}) \\
&= (e_{(1)}, xe_{(3)}y_{(2)} \mid \underline{e_{(2)}y_{(1)}}) \\
&= (e_{(1)}, xe'_{(2)}y_{(2)} \mid \underline{e_{(2)}e'_{(1)}y_{(1)}}) \\
&= (e_{(1)}, x(e'y)_{(2)} \mid \underline{e_{(2)}(e'y)_{(1)}}) \\
&= (e_{(1)}, xy_{(2)} \mid \underline{e_{(2)}y_{(1)}}) \\
&= (s(y_{(1)}), xy_{(2)}).
\end{aligned}$$

If H and K are weak bimonoids, a **weak bimonoid homomorphism** $f : H \rightarrow K$ must commute with both monoid and comonoid structures as in subsection 7.8. It follows that it also commutes with s, t, r ; this is shown in [PS09, Lemma 1.2], and rendered below for s in type theory:

$$\begin{aligned}
f(s(x)) &= (f(e_{(1)}) \mid \underline{e_{(2)}x}) \\
&= (f(e_{(1)}) \mid \underline{f(e_{(2)}x)}) \\
&= (f(e_{(1)}) \mid \underline{f(e_{(2)})f(x)}) \\
&= (f(e)_{(1)} \mid \underline{f(e)_{(2)}f(x)}) \\
&= (e_{(1)} \mid \underline{e_{(2)}f(x)}) \\
&= s(f(x)).
\end{aligned}$$

A weak bimonoid is a **weak Hopf monoid** if it has an **antipode**: a morphism $x : H \vdash \bar{x} : H$ satisfying

$$\begin{aligned}
\overline{x_{(1)}} x_{(2)} &= t(x) \\
x_{(1)} \overline{x_{(2)}} &= r(x) \\
\overline{x_{(1)}} x_{(2)} \overline{x_{(3)}} &= \bar{x}.
\end{aligned}$$

This implies immediately that

$$\bar{x} = \overline{x_{(1)}} x_{(2)} \overline{x_{(3)}} = \overline{x_{(1)}} r(x_{(2)}) \quad \bar{x} = \overline{x_{(1)}} x_{(2)} \overline{x_{(3)}} = t(x_{(1)}) \overline{x_{(2)}}.$$

As for ordinary bimonoids in subsection 7.8, antipodes for weak bimonoids are unique: if $x : H \vdash \bar{x} : H$ and $x : H \vdash \hat{x} : H$ are both antipodes, we have:

$$\begin{aligned}\bar{x} &= \overline{x_{(1)}} r(x_{(2)}) \\ &= \overline{x_{(1)}} x_{(2)(1)} \widehat{x_{(2)(2)}} \\ &= \overline{x_{(1)(1)}} x_{(1)(2)} \widehat{x_{(2)}} \\ &= t(x_{(1)}) \widehat{x_{(2)}} \\ &= \hat{x}.\end{aligned}$$

Similarly, we can translate the argument of [PS09, Proposition 2.2] that homomorphisms $f : H \rightarrow K$ of weak bimonoids preserve antipodes:

$$\begin{aligned}f(\bar{x}) &= f(\overline{x_{(1)}} r(x_{(2)})) \\ &= f(\overline{x_{(1)}}) f(r(x_{(2)})) \\ &= f(\overline{x_{(1)}}) r(f(x_{(2)})) \\ &= f(\overline{x_{(1)}}) f(x_{(2)(1)}) \overline{f(x_{(2)(2)})} \\ &= f(\overline{x_{(1)}}) f(x_{(2)(2)}) \overline{f(x_{(2)(2)})} \\ &= f(\overline{x_{(1)(1)}}) f(x_{(1)(2)}) \overline{f(x_{(2)})} \\ &= f(\overline{x_{(1)(1)}} x_{(1)(2)}) \overline{f(x_{(2)})} \\ &= f(t(x_{(1)})) \overline{f(x_{(2)})} \\ &= t(f(x_{(1)})) \overline{f(x_{(2)})} \\ &= t(f(x_{(1)})) \overline{f(x_{(2)})} \\ &= \overline{f(x)}.\end{aligned}$$

We end by sketching another approach to these sorts of uniqueness results that is inspired by the “types are like sets” perspective of our type theory. For monoids in the category of sets, the uniqueness of inverses is a *pointwise* property: for any element x , if y and z are two elements that are both inverses of x , then $y = z$. However, the uniqueness of antipodes as we have proven it above, for both bimonoids and weak bimonoids, is instead a statement about *inversion operators* that apply to all elements at once.

The pointwise statement is stronger, and this makes for simpler proofs. For instance, we can simply show that a monoid homomorphism $f : H \rightarrow K$ preserves inverses of elements in the sense that if y is an inverse of x then $f(y)$ is an inverse of $f(x)$, and conclude directly from pointwise uniqueness of inverses in K that f of “the” inverse of x coincides with “the” inverse of $f(x)$.

We can also formulate a “pointwise” sort of uniqueness of inverses in the general case. Suppose H is a weak Hopf monoid, with given antipode $x : H \vdash \bar{x} : H$. Suppose also we have a comonoid X and a comonoid morphism $x : X \vdash g(x) : H$, and also a morphism

$x : X \vdash i(x) : H$ such that

$$\begin{aligned} i(x_{(1)}) g(x_{(2)}) &= t(g(x)) \\ g(x_{(1)}) i(x_{(2)}) &= r(g(x)) \\ i(x_{(1)}) g(x_{(2)}) i(x_{(3)}) &= i(x). \end{aligned} \tag{7.10}$$

We think of g as an X -indexed family of elements of H , and i as a similarly indexed family of “inverses”. Then we can calculate

$$\begin{aligned} t(g(x_{(1)})) i(x_{(2)}) &= i(x_{(1)}) g(x_{(2)}) i(x_{(3)}) = i(x) \\ i(x_{(1)}) r(g(x_{(2)})) &= i(x_{(1)}) g(x_{(2)}) i(x_{(3)}) = i(x) \end{aligned}$$

and hence

$$\begin{aligned} i(x) &= i(x_{(1)}) r(g(x_{(2)})) \\ &= i(x_{(1)}) g(x_{(2)}) \overline{g(x_{(3)})} \\ &= t(g(x_{(1)})) \overline{g(x_{(2)})} \\ &= t(g(x)_{(1)}) \overline{g(x)_{(2)}} \\ &= \overline{g(x)}. \end{aligned}$$

Note that in fact it suffices for X to be a co-semigroup and g to be a co-semigroup morphism. Moreover, instead of a single co-semigroup, X could be a context of co-semigroups.

Applying this with $X \stackrel{\text{def}}{=} H$ and $g(x) \stackrel{\text{def}}{=} x$, we obtain uniqueness of the antipode itself. But we can also derive preservation of the antipode by a weak bimonoid homomorphism $f : H \rightarrow K$, by taking $X \stackrel{\text{def}}{=} H$ and $g \stackrel{\text{def}}{=} f$ with $i(x) \stackrel{\text{def}}{=} f(\overline{x})$. We simply check that this satisfies the three properties (7.10):

$$\begin{aligned} f(\overline{x_{(1)}}) f(x_{(2)}) &= f(\overline{x_{(1)}} x_{(2)}) \\ &= f(t(x)) \\ &= t(f(x)). \\ f(x_{(1)}) f(\overline{x_{(2)}}) &= f(x_{(1)} \overline{x_{(2)}}) \\ &= f(r(x)) \\ &= r(f(x)). \\ f(\overline{x_{(1)}}) f(x_{(2)}) f(\overline{x_{(3)}}) &= f(\overline{x_{(1)}} x_{(2)} \overline{x_{(3)}}) \\ &= f(\overline{x}). \end{aligned}$$

Then the above argument shows immediately that $f(\overline{x}) = \overline{f(x)}$, as desired.

As a final example, we show that for a weak Hopf monoid H , the antipode is a weak monoid anti-homomorphism. (Since Hopf monoids are self-dual, this implies that the antipode is also a comonoid anti-homomorphism, which was proven by a long string diagram

calculation in [PS09, Proposition 2.3]). Consider how we prove the analogous statement in the category of sets, that inversion in a group is a monoid anti-homomorphism. Arguably the most natural way is to simply check that $y^{-1}x^{-1}$ is an inverse of xy :

$$\begin{aligned}(y^{-1}x^{-1})(xy) &= y^{-1}(x^{-1}x)y = y^{-1}ey = y^{-1}y = e \\ (xy)(y^{-1}x^{-1}) &= x(yy^{-1})x^{-1} = xex^{-1} = xx^{-1} = e\end{aligned}$$

and conclude by uniqueness of inverses that $y^{-1}x^{-1} = (xy)^{-1}$.

Using our notion of pointwise uniqueness, we can reproduce this inside our type theory to apply to weak Hopf monoids. We take $X \stackrel{\text{def}}{=} (H, H)$, with its induced comonoid structure $x : H, y : H \vdash (x_{(1)}, y_{(1)}, x_{(2)}, y_{(2)}) : (H, H, H, H)$. We define $g(x, y) \stackrel{\text{def}}{=} xy$, which is a co-semigroup morphism (though not a comonoid morphism) by the first axiom of a weak bimonoid, and $i(x, y) \stackrel{\text{def}}{=} \overline{y}\overline{x}$, and check the three properties (7.10):

$$\begin{aligned}i((x, y)_{(1)})g((x, y)_{(2)}) &= \overline{y_{(1)}}\overline{x_{(1)}}x_{(2)}y_{(2)} \\ &= \overline{y_{(1)}}t(x)y_{(2)} \\ &= \overline{(t(x)y)_{(1)}}(t(x)y)_{(2)} \quad (4) \\ &= t(t(x)y) \\ &= t(xy) \quad (3) \\ &= t(g(x, y)).\end{aligned}$$

Here the line labeled (4) uses the so-numbered equation $(y_{(1)}, t(x)y_{(2)}) = ((t(x)y)_{(1)}, (t(x)y)_{(2)})$ from [PS09], and similarly the line (3) is an instance of their equation (3). The next calculation is dual.

$$\begin{aligned}g((x, y)_{(1)})i((x, y)_{(2)}) &= x_{(1)}y_{(1)}\overline{y_{(1)}}\overline{x_{(2)}} \\ &= x_{(1)}r(y)\overline{x_{(2)}} \\ &= (x r(y))_{(1)}\overline{(x r(y))_{(2)}} \\ &= r(x r(y)) \\ &= r(xy) \\ &= r(g(x, y)).\end{aligned}$$

The final calculation uses the commutativity of t with r , equation (13) from [PS09] which we also proved above.

$$\begin{aligned}i((x, y)_{(1)})g((x, y)_{(2)})i((x, y)_{(3)}) &= \overline{y_{(1)}}\overline{x_{(1)}}x_{(2)}y_{(2)}\overline{y_{(3)}}\overline{x_{(3)}} \\ &= \overline{y_{(1)}}t(x_{(1)})r(y_{(2)})\overline{x_{(2)}} \\ &= \overline{y_{(1)}}r(y_{(2)})t(x_{(1)})\overline{x_{(2)}} \quad (13) \\ &= \overline{y}\overline{x} \\ &= i(x, y).\end{aligned}$$

We leave further applications to the interested reader.

References

- [Bar79] Michael Barr. **-autonomous categories*, volume 752 of *Lecture Notes in Mathematics*. Springer, 1979.
- [Bar91] Michael Barr. *-autonomous categories and linear logic. *Mathematical Structures in Computer Science*, 1(2):159–178, 1991.
- [BCR18] John C. Baez, Brandon Coya, and Franciscus Rebro. Props in network theory. *Theory and Applications of Categories*, 33(25):727–783, 2018. arXiv:1707.08321.
- [BCST96] R. Blute, R. Cockett, R. Seely, and T. Trimble. Natural deduction and coherence for weakly distributive categories. *J. Pure and Appl. Algebra*, 113:229–296, 1996.
- [Cam17] Tim Campion. If a \otimes -idempotent object has a dual, must it be self-dual? MathOverflow, 2017. <https://mathoverflow.net/q/277991> (version: 2017-08-09).
- [CPT16] Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *Mathematical Structures in Computer Science*, 26(3):367–423, 2016.
- [CS97] Robin Cockett and Robert Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133–173, 1997. Corrected version available at <https://www.math.mcgill.ca/rags/linear/wdc-fix.pdf>.
- [Dun06] Ross Duncan. *Types for quantum computing*. PhD thesis, Oxford University Computing Laboratory, 2006.
- [EL89] P. Eades and Xuemin Lin. How to draw a directed graph. *[Proceedings] 1989 IEEE Workshop on Visual Languages*, pages 13–17, 1989.
- [Has97] Masahito Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. PhD thesis, University of Edinburgh, 1997. <http://www.kurims.kyoto-u.ac.jp/~hassei/papers/ECS-LFCS-97-360.pdf>.
- [Joh02] Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium: Volumes 1 and 2*. Number 43 in Oxford Logic Guides. Oxford Science Publications, 2002.
- [JS91] André Joyal and Ross Street. The geometry of tensor calculus. I. *Adv. Math.*, 88(1):55–112, 1991.
- [JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Math. Proc. Cambridge Philos. Soc.*, 119(3):447–468, 1996.
- [Kis14] Aleks Kissinger. Finite matrices are complete for (dagger-)hypergraph categories. arXiv:1406.5942, 2014.
- [KL80] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *J. Pure Appl. Algebra*, 19:193–213, 1980.
- [Mac65] Saunders MacLane. Categorical algebra. *Bull. Amer. Math. Soc.*, 71:40–106, 1965.
- [ML98] Saunders Mac Lane. *Categories For the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, second edition, 1998.
- [Ong96] C.-H. L. Ong. A semantic view of classical proofs – type-theoretic, categorical, and denotational characterizations (extended abstract). In *Proceedings of LICS ’96*, pages 230–241. IEEE Press, 1996.
- [PLC17] Jovana Obradović Pierre-Louis Curien. A formal language for cyclic operads. *Higher Structures*, 1(1), 2017. arXiv:1602.07502.

- [PR84] Roger Penrose and Wolfgang Rindler. *Spinors and space-time. Vol. 1.* Cambridge Monographs on Mathematical Physics. Cambridge University Press, Cambridge, 1984. Two-spinor calculus and relativistic fields.
- [PS09] Craig Pastro and Ross Street. Weak Hopf monoids in braided monoidal categories. *Algebra Number Theory*, 3(2):149–207, 2009.
- [PS14] Kate Ponto and Michael Shulman. Traces in symmetric monoidal categories. *Expositiones Mathematicae*, 32(2):248–273, 2014. arXiv:1107.6032.
- [Red93] Uday S. Reddy. A typed foundation for directional logic programming. In E. Lamma and P. Mello, editors, *Extensions of Logic Programming*, pages 282–318, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
- [Sel11] Peter Selinger. A survey of graphical languages for monoidal categories. In Bob Coecke, editor, *New Structures for Physics*, chapter 4. Springer, 2011. arXiv:0908.3347.
- [Shi99] Masaru Shirahata. A sequent calculus for compact closed categories, 1999. <http://www.fbc.keio.ac.jp/~sirahata/Research/cml1.ps>.
- [Sza75] M.E. Szabo. Polycategories. *Communications in Algebra*, 3(8):663–689, 1975.

*Department of Mathematics
 University of San Diego
 5998 Alcala Park
 San Diego, CA, 92110
 USA*
 Email: shulman@sandiego.edu

Iteration theories as monadic algebras

Jiří Adámek *

Iteration theories, as introduced by Stephen Bloom and Zoltán Ésik, are Lawvere theories equipped with a function assigning to every morphism $e : n \rightarrow n + k$ (representing a system of n recursive equations whose left-hand sides are variables and right-hand sides are terms in those n variables and k parameters) a morphism $e^\dagger : n \rightarrow k$ (representing a solution of those equations). This function is required to satisfy a number of equational properties, for example the equation $e^\dagger = [e^\dagger, \text{id}] \cdot e$ telling us that e^\dagger is indeed a solution of e . The motivating idea is “to axiomatize the equational properties of all the computationally interesting structures of this kind” (citation from [2]). We claim that this statement is correct, in fact, it can be sharpened to state that iteration theories precisely axiomatize the equational properties of

$$e^\dagger = \text{the least solution of } e$$

as used in Domain Theory (and being, no doubt, computationally interesting).

In fact, Bloom and Ésik proved in [1] that for every finitary signature Σ a free iteration theory on Σ can be described as the theory R_Σ of rational Σ -trees. More precisely, the obvious forgetful functor U from the category ITh of iteration theories to the category $\text{Sgn} = \text{Set}/\mathbb{N}$ of signatures has a left adjoint given by $\Sigma \mapsto R_\Sigma$, where the theory R_Σ has as morphisms from n to 1 all rational Σ -trees in n variables. We prove more:

Theorem. *The forgetful functor $U : \text{ITh} \rightarrow \text{Sgn}$ is monadic.*

This tells us that iteration theories can be understood as the monadic algebras for the monad $\mathbb{R}\text{at}$ on the category Sgn given by the above adjunction. And using the rational algebraic theories of the ADJ group [3], which essentially are ordered algebraic theories with least solutions of iterative equation morphisms, one obtains the same monad $\mathbb{R}\text{at}$ as the monad of free rational theories. It is easy to see that every rational theory has a free completion to a continuous theory as used in Domain Theory. Therefore, every equational property satisfied by the “least solution” function can be derived from the axioms of iteration theories. (And vice versa — this direction is contained in [1].)

REFERENCES

- [1] S. L. Bloom and Z. Ésik, *Iteration Theories*, Springer Verlag, 1993.

*Joint work with Stefan Milius and Jiří Velebil.

- [2] S. L. Bloom and Z. Ésik, Fixed-point operations on ccc's, Part I, *Theoret. Comput. Sci.* 155 (1996), 1–38.
- [3] J. B. Wright, J. W. Thatcher, E. G. Wagner and J. A. Goguen, Rational algebraic theories and fixed-point solutions, *Proc. 17th IEEE Symposium on Foundations of Computing*, Houston, Texas, 1976, 147–158.

Algebraic examples of differential categories

Rick Blute ^{*}

Differential categories were introduced by Blute, Cockett and Seely in part as a categorical axiomatization of the differential lambda-calculus of Ehrhard and Regnier. The key idea is that one has a monoidal category with a comonad; one thinks of the base category as the linear maps and the co-Kleisli category as the differential maps. A differential category then has a differential combinator satisfying abstract analogues of the usual differentiation axioms.

In this talk, we consider other algebraic approaches to differentiation. We consider Ritt's notion of differential algebra, and show that every differential algebra generates a differential category, enriched over the base field. We also establish connections to homological algebra using the bar resolution as an endofunctor on the category of chain complexes.

^{*}Joint work with Robin Cockett, Kim Flak, and Robert Seely.

Prolocalisations of homological categories

Francis Borceux *

A prolocalisation of an homological (resp. semi-abelian) category is a regular full reflective subcategory, whose reflection preserves short exact sequences. We study the closure operator and the torsion theory associated with such a situation. We pay special attention to the fibered, the epireflective and the monoreflective cases. We give examples in algebra, topos theory, functional analysis.

*Joint work with Maria Manuel Clementino, Marino Gran and Lurdes Sousa.

New aspects of the Schreier-Mac Lane's extension theorem

Dominique Bourn

It is classical that any extension of groups:

$$1 \longrightarrow K \xrightarrow{k} X \xrightarrow{f} Y \longrightarrow 1$$

determines, via conjugation in the group X , a group homomorphism $\phi : Y \rightarrow \text{Aut } K / \text{Int } K$, called the *abstract kernel* of this extension, which allows to recover the set $\text{Ext}_\phi(Y, K)$ of all isomorphic classes of extensions with abstract kernel ϕ . This comes from the fact that, with the combinatorial notion of *obstruction*, it was possible to show that there is on $\text{Ext}_\phi(Y, K)$ an action of the abelian group $\text{Ext}_{\bar{\phi}}(Y, ZK)$ of extensions with abelian kernel the centre ZK of the group K and with module structure given by the restriction $\bar{\phi}(y)$ to ZK of the automorphisms $\phi(y)$; $y \in Y$.

The recent introduction of the notion of *split extension classifier* (spec) (Borceux, Janelidze, Kelly) and *action groupoid* (Borceux, Bourn) in the context of protomodular categories will allow us to show that the previous result on extensions fully holds in any pointed protomodular Barr exact category \mathbb{C} with specs. It is not only a simple generalization of the result in Gp , but also an enlightenment about the structure of extensions, since they will appear as morphisms in a certain groupoid $Tors\mathbb{C}$, which will give another meaning to the group action we recalled above. The main point is that the *action groupoid* $\underline{D}_1 X$ of an object X retains all what remains of the abelianity of X and that the kernel of its normalization $j_X : X \rightarrow DX$ is the centre ZX of the object X . On the other hand, with any abstract kernel $\phi : Y \rightarrow Q_K$, where Q_K is the π_0 of the groupoid $\underline{D}_1 K$, is associated a groupoid $\underline{D}_1 \phi$ which will appear to be the *categorical expression of the obstruction*.

Branched coverings of quasi locally connected toposes

Marta Bunge

Just as in topology, the notion of a complete spread in topos theory [1] is a useful tool in describing branched coverings of an \mathcal{S} -bounded locally connected topos \mathcal{E} . Inspired once again by topology, the locally connected assumption on the domains of complete spreads was removed in [2], and replaced in [3] by an assumption of quasi local connectedness, satisfied by all Grothendieck toposes. In this context, the Lawvere distributions [4] (suitable in the locally connected case) are substituted by distributions with values in the category of zero-dimensional (rather than of discrete) locales in \mathcal{S} .

The purposes of this talk are as follows. The first is to examine the theory presented in [2, 3] with respect to the validity of the complete spread analysis of branched coverings [1] when \mathcal{E} is assumed to be just quasi locally connected. In particular, this gives new topological applications to branched coverings over an arbitrary topological space E . The second is to exploit the comprehensive factorization of a geometric morphism [2, 3] in order to begin, in the manner of [5], a study of the ‘fundamental groupoid topos’ of a quasi locally connected topos \mathcal{E} over an arbitrary base \mathcal{S} . A particular instance is the case of a Grothendieck topos \mathcal{E} .

REFERENCES

- [1] M. Bunge, *Galois Groupoids and Covering Morphisms in Topos Theory*, in: G. Janelidze et al. (eds), Galois Theory, Hopf Algebras, and Semiabelian Categories, Fields Institute Communications, American Mathematical Society (2004) 131-162.
- [2] M. Bunge and J. Funk, *Singular Coverings of Toposes*, Lecture Notes in Mathematics 1890, Springer-Verlag, 2006.
- [3] M. Bunge and J. Funk, Quasicomponents in topos theory: the hyperpure, complete spread factorization, *Math. Proc. Cambridge Phil. Soc.* 142 (2007) 47-62.
- [4] M. Bunge and J. Funk, Quasi locally connected toposes, *Theory and Applications of Categories* 18:08 (2007) 209-239.
- [5] F. W. Lawvere. Intensive and extensive quantities, *Notes for the lectures given at the workshop on Categorical Methods in Geometry*, Aarhus, 1983.

Comparing operadic theories of n -categories

Eugenia Cheng

We give a framework for comparing on the one hand theories of n -categories that are weakly enriched operadically, and on the other hand theories given using a globular operad. Examples of the former are the definition by Trimble and variants (May/Cheng/Gurski) and examples of the latter are the definition by Batanin and variants (Leinster). We will show how to take a theory of n -categories of the former kind and produce a globular operad whose algebras are the n -categories we started with. Our main aim is to show how the globular operad associated with Trimble's topological definition in this way, is related to the globular operad used by Batanin to define fundamental n -groupoids of spaces. This has the potential to provide an easier method for proving Batanin's Homotopy Hypothesis than the partial proofs currently in the literature; it also has the potential to provide a useful class of semi-strict n -categories.

Join restriction categories: the importance of being adhesive

Robin Cockett ^{*}

The completeness theorem for restriction categories [1] states that every restriction category is a full subcategory of a partial map category. A join restriction category is a restriction category in which every set of compatible maps has a join. A natural question is: for what sort of partial map category are join restriction categories complete in the above sense.

The answer, somewhat to our surprise, is that the partial map category $\text{Par}(\mathbf{X}, \mathcal{M})$ must be \mathcal{M} -adhesive, [3], and, furthermore, the stable system of monics \mathcal{M} must be closed to \mathcal{M} -gaps. Adhesive categories use Van Kampen colimits in their definition and arose from the desire to give a general categorical framework in which double-pushout rewriting was possible. On the other hand, a key construction for join restriction categories is the manifold completion [2]. This implies that this sort of rewriting can be viewed abstractly as surgery on manifolds.

Given a restriction category there is a universal way to complete it to a join restriction category by considering down-closed subsets of compatible maps as the maps. On the other hand, given a representation of a restriction category into a partial map category $\text{Par}(\mathbf{X}, \mathcal{M})$, where \mathbf{X} is \mathcal{M} -adhesive, there is a canonical way of enlarging the system of monics \mathcal{M} to include all the \mathcal{M} -gaps. This gives, therefore, an alternative method for adding joins as the standard Yoneda representation of restriction categories [1] is of this form.

This second technique for obtaining a join completion underlies, for example, the building of schemes in algebraic geometry: here one starts with $\text{Par}(\text{CRing}^{\text{op}}, \mathcal{L})$ (where \mathcal{L} is the class of localizations) and using the Yoneda representation one builds spaces which are locally ringed. The fact that these two constructions actually coincide, therefore, gives an alternate view of schemes as manifolds over the, possibly more simply described, universal join completion.

REFERENCES

- [1] J.R.B. Cockett and S. Lack, *Restriction categories I: categories of partial maps* Theoretical Computer Science 270 (2002) 223-259.
- [2] M. Grandis, *Cohesive categories and manifolds*, Ann. Mat. Pura Appl. 157 (1990) 199-244.
- [3] S. Lack and P. Sobociński, *Adhesive and quasiadhesive categories*, Theoretical Informatics and Applications 39 (2005) 511-546.

^{*}Joint work with Xiuzhan Guo.

Quantum physics as it is practised in the lab

Bob Coecke

Quantum physics as it is practised in the lab can be given a purely category-theoretic axiomatics. We define both quantum and classical systems as structured objects within a symmetric monoidal category. Quantum measurements turn out to be Eilenberg-Moore coalgebras relative to a comonad induced by a classical structure. In particular, the Eilenberg-Moore commuting square condition embodies von Neumann's projection postulate of quantum mechanics: when one repeats a measurement one obtains the same outcome in the second measurement as one obtained in the first one. This language provides an extremely powerful tool for the design and analysis of sophisticated quantum informatic protocols which involve highly non-trivial interaction between the quantum world and the classical world.

What is particularly appealing to the practising physicist is that this categorical language comes with an intuitive diagrammatic calculus. In fact, the graphical language for monoidal categories, quantum structures and classical structures is in fact a two-dimensional "correction" and "refinement" of the so-called Dirac-notation, which is very popular among physicists.

The category-theoretic background for this talk traces back to work by Kelly & Laplaza, Carboni & Walters, Joyal & Street and others. This talk is based on several papers which also include work by/with Abramsky, Duncan, Paquette, Pavlovic & Selinger. For papers and links please consult:

http://se10.comlab.ox.ac.uk:8080/BobCoecke/Maintopic_en.html

Coherence With and Without Unique Normal Forms

Jonathan A. Cohen

One may define a structure on a category to be a two-dimensional system of generators and relations. In the case where all of the structuring functors are covariant in all of their arguments, Kelly [3] showed how to generate a doctrine, \mathcal{D} , whose algebras are categories endowed with the free such structure. Viewing a coherence problem as being concerned with the commutativity of diagrams in the initial \mathcal{D} -algebra on a discrete category, one may delineate two related coherence questions: (1) Do all diagrams commute?, and (2) Is there a decision procedure that determines whether a given diagram commutes?

We recast Kelly's construction in the framework of term rewriting theory, where the initial \mathcal{D} -algebra on a discrete category becomes a term rewriting system modulo a two-dimensional congruence. Within this framework, it is possible to resolve the above coherence questions in terms of the underlying rewrite system. We shall briefly discuss a resolution to (1) for structures described by rewriting systems with unique normal forms. Such systems lend themselves to arguments involving induction on the length of the longest reduction to a normal form. The solution we provide can be seen as a natural generalisation of Laplaza's proof of coherence for distributive categories [4] and a close relation of Johnson's general coherence theorem for a class of n -categorical pasting schemes [2]. Following this, we describe an alternative approach via planar subdivisions, which applies also in the case where the system does not enjoy unique normal forms, and obtain sufficient conditions for the above two questions to be resolved in the affirmative. An important example of a system without unique normal forms is provided by iterated monoidal categories [1] and we demonstrate how our results can be used to give a conceptual approach to the coherence problem for these structures.

REFERENCES

- [1] C. Balteanu, Z. Fiedorowicz, R. Schwänzl, and R. Vogt, Iterated monoidal categories, *Adv. Math.*, 176(2):277–349, 2003.
- [2] Michael Johnson, *Pasting diagrams in n -categories with applications to coherence theorems and categories of paths*, PhD thesis, The University of Sydney, 1988.
- [3] G. M. Kelly, On clubs and doctrines, in: *Category Seminar (Proc. Sem., Sydney, 1972/1973)*, pages 181–256, Lecture Notes in Math. 420, Springer, Berlin, 1974.
- [4] Miguel L. Laplaza, Coherence for distributivity, In *Coherence in categories*, pages 29–65, Lecture Notes in Math., Vol. 281, Springer, Berlin, 1972.

Analysis and enriched Category Theory

Geoff Cruttwell

In his classic 1973 paper [1], Lawvere showed that metric spaces are an example of enriched categories. In this talk, we will investigate the question: if metric spaces are enriched categories, what are normed spaces?

REFERENCES

- [1] F.W. Lawvere, *Metric spaces, generalized logic, and closed categories*, TAC Reprints, www.tac.mta.ca/tac/reprints/.

Modular categories and (extended) topological field theories

Alexei Davydov

It is believed that 3d Topological Field Theories (TFTs) associated with modular categories (in the sense of Turaev) extend to functors between bicategory of cobordism and 2-category of “2-vector spaces”. We will explain how it works in the case of Dijkgraaf-Witten gauge TFT with a finite gauge group.

2-Proobjects

Eduardo J. Dubuc

Recall that an object A in a topos \mathcal{E} is a Galois object if it is an $\text{Aut}(A)$ -torsor, and that a Galois topos is an atomic topos generated by its Galois objects. As an application of the construction of 2-filtered colimits of categories [3] we derive a theorem of existence of 2-filtered inverse limits of topoi. We show then that every Galois topos has a point.

REFERENCES

- [1] E. J. Dubuc, R. Street, *A construction of 2-filtered bicolimits of categories*, Cahiers de Topologie et Géométrie Différentielle 47 (2006) 83–106.

$*$ -Autonomous functor categories

J.M. Egger $*$

Given a mere functor between $*$ -autonomous categories, $M: \mathcal{J} \rightarrow \mathcal{K}$, it is possible to define a second functor $G: \mathcal{J} \rightarrow \mathcal{K}$ by $G(x) = (M(x^*))^*$ which should be intuitively thought of as the dual of M . [In the non-symmetric case, it is natural to consider two dual functors: $G^+(x) = (M(*x))^*$, and $G^-(x) = *(\overline{M(x)})$. The remarks below assume symmetry for ease of presentation only.]

A particular case of interest is when M is (lax) monoidal with respect to the tensor structures on \mathcal{J} and \mathcal{K} . [Recall that a $*$ -autonomous category has, in general, two monoidal structures: one closed, the other co-closed; the former is called *tensor* and the latter, *par*.] Then, of course, G is comonoidal (oplax monoidal) with respect to the par structures on \mathcal{J} and \mathcal{K} . But, as pointed out by Cockett and Seely [2], this is really only the beginning: M and G are related by four natural transformations (*linear strengths* and *linear costrengths*) which, intuitively, describe a two-sided action of M on G and a two-sided coaction of G on M . These natural transformations are required to satisfy coherence conditions which, again intuitively, say that the action of M on G is G -coequivariant, and that the coaction of G on M is M -equivariant.

Certainly, in the case where \mathcal{J} is the terminal category, these intuitions are borne out, as shown by the author in [3].

Now suppose that \mathcal{K} has colimits of size \mathcal{J} ; then it is possible to define a *convolution tensor* on the functor category $\mathcal{K}^{\mathcal{J}}$ in such a way that monoidal functors $\mathcal{J} \rightarrow \mathcal{K}$ are in bijective correspondence with monoids in $\mathcal{K}^{\mathcal{J}}$, and such that actions of monoidal functors on mere functors are just that: actions of a monoid on an object. Dually, if \mathcal{K} has limits of size \mathcal{J} , then we can also define a *co-convolution par* on $\mathcal{K}^{\mathcal{J}}$ such that comonoidal functors $\mathcal{J} \rightarrow \mathcal{K}$ correspond to comonoids in $\mathcal{K}^{\mathcal{J}}$.

But in order to fully substantiate our intuitions, it is necessary to first construct linear distributions relating the convolution tensor and the co-convolution par—without them we have, for instance, no means of describing a G -coaction on $M \otimes G$, and therefore cannot make sense of the idea that the left action $M \otimes G \rightarrow G$ should be G -coequivariant. The construction of such linear distributions is our first main result. [This is done in greater generality than that presented above: it suffices, for example, that \mathcal{J} and \mathcal{K} be *bilinear*, in the sense of [1].]

Given the existence of a linear distributive structure on $\mathcal{K}^{\mathcal{J}}$, we can revisit original intuition: that (even in the case where M is a mere functor) the functor G should be regarded as the dual of M . Our second main result is that one can construct a

$*$ Joint work with D. Kruml.

linear adjoint overlying M and G ; this entails that the functor category $\mathcal{K}^{\mathcal{J}}$ is itself $*$ -autonomous and that G is indeed (isomorphic to) M^* . [In the non-symmetric case, $G^+ \cong M^*$ and $G^- \cong {}^*M$.]

These theorems were first proven in the case where $\mathcal{J} = \rightarrow$ and $\mathcal{K} = \mathbf{Sup}$ [4]; the concept of *cyclic Frobenius monoid* in the $*$ -autonomous category $\mathbf{Sup}^{\rightarrow}$ has proven surprisingly useful, and will be discussed in David's talk.

REFERENCES

- [1] J. R. B. Cockett and R. A. G. Seely, *Proof theory for full intuitionistic linear logic, bilinear logic, and MIX categories*, Theory Appl. Categ. 3 (1997) 85–131 .
- [2] J. R. B. Cockett and R. A. G. Seely, *Linearly distributive functors*, J. Pure Appl. Algebra 143 (1999) 155–203.
- [3] J.M. Egger, *The Frobenius relations meet linear distributivity*, submitted to *Theory and Applications of Categories*.
- [4] J.M. Egger and David Kruml, *Girard couples in quantale theory*.

Extended real number object in the bornological topos

Luis Espa  ol

A few years ago, F. William Lawvere [2] pointed out that, over the category Set , $\overline{\mathbb{R}}^+ = [0, \infty]$ is a closed category, and (generalized) metric spaces are $\overline{\mathbb{R}}^+$ -categories. This means that $\overline{\mathbb{R}}^+$ is a good recipient for metrics and norms. Later, Reichman [3] worked out the (Dedekind) extended real line $\overline{\mathbb{R}}_{\mathcal{E}}$, and its positive part $\overline{\mathbb{R}}_{\mathcal{E}}^+$, in an elementary topos \mathcal{E} with natural number object.

Our goal is to study these real objects, denoted $\overline{\mathbb{R}}_b$ and $\overline{\mathbb{R}}_b^+$, in the bornological topos \mathcal{B} , which is a subtopos of M -sets with the monoid $M = Set(\mathbb{N}, \mathbb{N})$ of all sequences of natural numbers. This topos can be presented has the subtopos of all sheaves for the topology generated by the finite epimorphic families in M . The study of this topos has been encouraged by Lawvere, and some first steps about are given in [1]. It was proved there that the (Dedekind) real number object in \mathcal{B} is the well known space $\mathbb{R}_b = \ell^\infty$.

In order to calculate $\overline{\mathbb{R}}_b$ it is useful to consider presheaves at first, that is, the extended real line $\overline{\mathbb{R}}_m$ in $MSet$, with $M = Set(\mathbb{N}, \mathbb{N})$. From Reichman's result for arbitrary monoids we calculate

$$\overline{\mathbb{R}}_m = \{\mu : \mathcal{P}(\mathbb{N}) \rightarrow \overline{\mathbb{R}} \mid \mu \text{ monotone, } \mu(\emptyset) = -\infty\},$$

and the same for $\overline{\mathbb{R}}_m^+$, but with $\overline{\mathbb{R}}^+$ and $\mu(\emptyset) = 0$ at the right side. Note that the action is $(\mu \circ f)(A) = \mu(f(A))$. In the bornological topos we obtain M -subsets of the above real presheaves, actually:

$$\overline{\mathbb{R}}_b = \{\mu : \mathcal{P}(\mathbb{N}) \rightarrow \overline{\mathbb{R}} \mid \mu \text{ preserves finite } \vee\},$$

and similarly $\overline{\mathbb{R}}_b^+$. Of course, there are the canonical inclusion $\mathbb{R}_b \hookrightarrow \overline{\mathbb{R}}_b$, the locale structure given by $inf, sup : \mathcal{P}(\overline{\mathbb{R}}_b) \rightarrow \overline{\mathbb{R}}_b$, the closed structure on $\overline{\mathbb{R}}_b^+$, etc.

REFERENCES

- [1] L. Espa  ol, L. Lamb  n, *On bornologies, locales and toposes of M -Set*, J. Pure and Appl. Algebra 176 (2002) 113-125.
- [2] F. W. Lawvere, *Metric spaces, generalized logic, and closed categories*, Rend. Sem. Mat. e Fisico di Milano 43 (1973) 135-166.
- [3] J. Z. Reichman, *Semicontinuous real numbers in a topos*, J. Pure and Appl. Algebra 28 (1983) 81-91.

Generalized higher Hopf formulae and a long Stallings-Stammbach exact sequence

Tomas Everaert *

Using categorical Galois structures [7] of *higher extensions* and *higher central extensions*, Brown and Ellis's higher Hopf formulae [1] for the homology of a group can be defined in any semi-abelian category \mathcal{A} [8], relative to a choice of Birkhoff subcategory \mathcal{B} of \mathcal{A} . We establish a long exact sequence of higher Hopf formulae, which extends the well-known Stallings-Stammbach five term exact sequence in the case $\mathcal{A} = \mathbf{Gp}$ is the variety of groups and $\mathcal{B} = \mathbf{Ab}$ is the subvariety of abelian groups. We apply our results to the situation where $\mathcal{A} = \mathbf{PXMod}$ is the category of precrossed modules and $\mathcal{B} = \mathbf{XMod}$ the category of crossed modules.

REFERENCES

- [1] R. Brown and G. J. Ellis, *Hopf formulae for the higher homology of a group*, Bull. London Math. Soc. 20 (1988) 124-128.
- [2] T. Everaert, *An Approach to Non-Abelian Homology based on Categorical Galois Theory*, PhD. Thesis, Vrije Universiteit Brussel, 2007.
- [3] T. Everaert and M. Gran, *On low-dimensional homology in categories*, Homology, Homotopy and Appl 9 (2007) 275-293.
- [4] T. Everaert, M. Gran and T. Van der Linden, *Higher Hopf formulae for homology via Galois Theory*, math.AT/0701815 (2007).
- [5] T. Everaert and T. Van der Linden, *Baer invariants in semi-abelian categories I: General theory*, Theory Appl. Categ. 12 (2004) 1-33.
- [6] T. Everaert and T. Van der Linden, *Baer invariants in semi-abelian categories II: Homology*, Theory Appl. Categ. 12 (2004) 195-224.
- [7] G. Janelidze, *Pure Galois theory in categories*, J. Algebra 132 (1990) 270-286.
- [8] G. Janelidze, L. Marki and W. Tholen *Semi-abelian categories*, J. Pure Appl. Algebra 168 (2002) 367-386.

*Joint work with Marino Gran and Tim Van der Linden.

Towards a mathematical theory of substitution

Marcelo Fiore

The notion of substitution is ubiquitous in mathematics. However, there is as yet no theory that treats this commonsense notion in full generality. This talk is a step in this direction. Specifically, through an analysis of the concept and role of substitution in algebra, combinatorics, type theory, *etc.* I will synthesise a general unifying framework for defining and studying substitution. Along the way, I will use the theory to extract correct substitution algorithms, provide initial-algebra semantics that respect substitution, establish the admissibility of the cut rule in type theories, initiate the reduction of type theory to algebra, *etc.* Overall, the mathematical development will touch upon models of algebraic, combinatorial, logical, computational, and physical structures.

Model structures on the category of double categories

Tom Fiore *

When are two categories the same? One possible notion of sameness is equivalence of categories: two categories are the same if there is a fully faithful essentially surjective functor between them. A coarser possibility is to consider two categories the same if there is a functor between them whose nerve is a weak homotopy equivalence of topological spaces. As is well known, these distinct notions of weak equivalence between categories have been encoded in model structures by [2] and [5].

One can ask the same question for double categories: when are two double categories the same? We collate the various notions of weak equivalence into model structures on the category of double categories. This complements recent work on model structures on **BiCat** and **2-Cat** in [3], [4], and [6] as well as the work [1] on model structures on internal categories.

REFERENCES

- [1] T. Everaert, R.W. Kieboom and T. Van der Linden, Model structures for homotopy of internal categories, *Theory Appl. Categ.*, 15(3) 66–94 (electronic), 2005/06.
- [2] A. Joyal and M. Tierney, Strong stacks and classifying spaces, *Category theory (Como, 1990)*, 213–236, Lecture Notes in Math., 1488, Springer, Berlin, 1991.
- [3] Stephen Lack, A Quillen model structure for 2-categories, *K-Theory* 26(2) 171–205, 2002.
- [4] Stephen Lack, A Quillen model structure for bicategories, *K-Theory*, 33(3) 185–197, 2004.
- [5] R.W. Thomason. Cat as a closed model category. *Cahiers Topologie Géom. Différentielle* 21(3) 305–324, 1980.
- [6] K. Worytkiewicz, K. Hess, P. E. Parent and A. Tonks, A model structure à la Thomason on **2-Cat**, *J. Pure Appl. Algebra* 208 (2007), no. 1, 205–236.

*Joint work with Simona Paoli and Dorette Pronk.

F-inverse semigroups and toposes

Jonathon Funk

We focus on two aspects of how semigroups and toposes are related [1, 4]. One aspect has to do with “left *-semigroups,” and how to present the classifying topos $\mathcal{B}(T)$ of an inverse semigroup T in semigroup terms. The other is a ‘dictionary’ project connecting semigroup theory and topos theory. For instance, we explain that if a semigroup T is F-inverse [3], then $\mathcal{B}(T)$ has a torsion-free generator that is locally split [2].

REFERENCES

- [1] J. Funk, *Semigroups and toposes*, Semigroup Forum, 1-41, 2007.
- [2] A. Kock and I. Moerdijk, Presentations of étendues, *Cahiers de Top. et Géom. Diff. Catégoriques*, 32(2):145-164, 1991.
- [3] Mark V. Lawson, *Inverse Semigroups: The Theory of Partial Symmetries*, World Scientific Publishing Co., Singapore, 1998.
- [4] M. Lawson and B. Steinberg, *Etendues and ordered groupoids*, Cahiers de Top. et Géom. Diff. Catégoriques, 40(2):127-140, 2002.

Towards Bicategorical Quantum Mechanics the calculus of quantaloidal enrichment and its applicability to Quantum Mechanics

Emmanuel Galatoulas

The application of conceptual and formal tools from category theory to physics in general and to Quantum Mechanics (QM) in particular is a relatively new yet dynamically developing field of research, attracting the interest of both categorists and theoretical physicists. The distinctive and at the same time puzzling features of quantum mechanical systems like *entanglement*, *non-locality* and *indeterminism* suggest that a categorical approach to QM should primarily afford the grasping and formalisation of a notion of *quantum varying structure* representing the evolution of quantum systems and processes in a profound manner.

Understandably enough, the first attempts for such a grasping were implementations of the *topos-theoretical* machinery, since topoi, in particular topoi of (*pre*)sheaves, are the almost canonical example of varying categorical structure. However, not only is the intuitionistic internal logic of topoi rather incompatible with the distinctively non-commutative “logic” of QM, but also (pre)sheaves themselves, standardly being a kind of *varying sets*, can only provide a restricted representation of quantum mechanical variation. The latter seems to necessitate a more involved and implicated notion of “quantum sheaf” and “space” than the topos-theoretic one. Indeed, both the physics and the formalism of QM indicate that quantum mechanical systems and processes should be considered as *structured* objects, in other words *categories themselves*, having a non-trivial internal structure, living accordingly in a non-trivial ambient category too.

Category theory does provide the tools to develop a theory of such non-trivial objects-categories and their ambient environment. The relative “cost” in achieving this is that someone has to ascend the ladder of categorical complexity, employing not ordinary categories but categories of a higher order, known also as 2-categories and *bicategories*. To be precise, quantum systems *qua* objects-categories should be considered *enriched* categories. In fact, not only their ambient category but, even more importantly, the background category for the enrichment as well should be 2-categories. Indeed, it is enriching upon such a 2-category or a bicategory that the internal structure and the “dynamics” of such quantum objects can be properly probed and understood. For example, the 2-dimensional generalisation of ordinary sheaves (eg. in the form of stacks, see [3]) is implemented in the context of *variable categories* and a notion of *processes between processes*, as explained for instance in [1],

does naturally involve bicategories (the special place of bicategories in the categorical ladder is nicely commented for instance in [2]).

At first sight enriching over a bicategory looks quite a taunting task. However, there is a particular kind of bicategories which is very convenient to start with, namely quantaloids. Intuitively speaking, a quantaloid is a category whose homobjects are not just sets but complete suplattices. Quantaloids provide thus a handy calculus where 2-cells are just inequalities and all diagrams involving them commute by definition (for the relevant theory someone may follow for instance the series of papers by Isar Stubbe, eg. [4]).

Enriching over a quantaloid \mathcal{Q} gives rise to two categorical structures, the quantaloid **Dist**(\mathcal{Q}) of \mathcal{Q} -enriched categories and the *distributors* (also known as profunctors or bimodules) between them and the locally partially ordered 2-category **Cat**(\mathcal{Q}) of \mathcal{Q} -enriched categories and *functors* between them.

Our first aim towards what may be called **Bicategorical Quantum Mechanics**, is to develop and interpret the calculus of quantaloidal enrichment in order to recover the main elements of the standard QM and its formalism. Indeed, interpreting \mathcal{Q} -enriched categories as the quantum mechanical objects of our discourse, homarrows as *transition* amplitudes, the so-called presheaves on them as generalised “states”, and functors as generalised “operators”, leads to a considerable transcription of the main body of the formalism of ordinary QM. Even further, by means of a tentative analysis of the embedding of the quantum system into its free cocompletion (what we call the *space of dynamics* of the system) and to its *Cauchy* completion, we entertain the idea of a radical re-interpretation of quantum mechanical *measurement* as a *universal* construction in this bicategorical context.

Although still in the beginning of developing such an approach, we may argue that it has the potential to provide us with valuable insights to a consistent and intuitive understanding of the renowned conceptual intricacies of QM.

REFERENCES

- [1] P.Katis, Sabadini, RFC Walters, *Bicategories of Processes*, Journal of Pure and Applied Algebra 115 (1997) 141-178.
- [2] F.W. Lawvere, *Adjoints in and among Bicategories*, Lecture Notes in Pure and Appl. Math. 180, Dekker, 1996.
- [3] R. Street, *Cauchy characterization of enriched categories*, Reprints in Theory and Applications of Categories 4 (2004) 1-16.
- [4] I.Stubbe, *Categories enriched over quantaloids: categories, distributors and functors*, Theory and Applications of Categories 14 (2005) 1-45.

The theory of glueing things on

Richard Garner

The notion of weak factorisation system provides a convenient framework for the discussion of higher-dimensional structure that a mere category may possess. Yet its definition deals not in universal properties, but in *weakly* universal properties, rendering inapplicable much of the machinery that category theory could otherwise bring to bear.

This deficiency may be overcome if we move from weakly universal properties to algebraic structure. The *natural weak factorisation systems* (n.w.f.s.'s) of [1] effect such a transition: they are given by a monad-comonad pair satisfying various equational laws, which, whilst retaining the expressive power of weak factorisation systems, gain many of the pleasant properties of the theory of monads.

In this talk, we study one instance of this. The *cofibrantly generated* n.w.f.s.'s of [2] provide a notion of n.w.f.s. generated by a signature of basic operations: these operations being specified by a regulus of maps which we wish to belong to the left class of our n.w.f.s. This construction is “free”, providing the syntax side of a syntax/semantics adjunction; and may be viewed on the one hand, as a suitably refined free monad construction of the sort studied by Barr, Kelly, Wolff, *et al*; and on the other, as an improved version of the “small object argument” deployed in the construction of plain w.f.s.'s.

Cofibrantly generated n.w.f.s.'s frequently give rise to “weak morphism classifiers”; and we end with some examples of this.

REFERENCES

- [1] Richard Garner, *Cofibrantly generated natural weak factorisation systems*, arXiv preprint (2007), math.CT/0702290.
- [2] Marco Grandis and Walter Tholen, *Natural weak factorization systems*, Archivum Mathematicum 42 (2006) 397–408.

Projections of weak Hopf algebras and weak Yang-Baxter operators

R. González Rodríguez *

In this talk we show that, for a weak Hopf algebra projection $g : B \rightarrow H$, the subalgebra of coinvariants B_H of B , defined in [1], is a Hopf algebra in the category ${}^H_H\mathcal{YD}$, i.e., the category of Yetter-Drinfeld modules defined by Böhm in [4]. Also, we complete this result, obtaining a one to one correspondence between Hopf algebras in the category ${}^H_H\mathcal{YD}$ and projections of weak Hopf algebras $g : B \rightarrow H$ ([2] and [3]).

To prove these results, in [3] we introduce the notions of weak Yang-Baxter operator and weak braided Hopf algebra and we show that it is possible to construct non-trivial examples of this algebraic structures using Hopf algebras in ${}^H_H\mathcal{YD}$ where the antipode of H is an isomorphism. The definition of weak braided Hopf algebra, generalizes the one introduced by Takeuchi in [8], and as a particular instances we obtain the definition of weak Hopf algebra (see [5]) and, if the weak Yang-Baxter operator is the braiding of a braided category, the new notion of weak Hopf algebra in a braided setting.

Finally, we want to emphasize that these results give a good weak Hopf algebra interpretation of well-known theorems proved by Radford [7], Majid [6] and others in the Hopf algebra setting, that provides a correspondence between Hopf algebra projections and Hopf algebras in the category of Yetter-Drinfeld modules.

REFERENCES

- [1] J.N. Alonso Álvarez, R. González Rodríguez, *Crossed products for weak Hopf algebras with coalgebra splitting*, J. of Algebra 281 (2004) 731-752.
- [2] J.N. Alonso Álvarez, R. González Rodríguez, J.M. Fernández Vilaboa, *Yetter-Drinfeld modules and projections of weak Hopf algebras*, to appear in J. of Algebra (2006).
- [3] Alonso Álvarez, J.N., González Rodríguez, R., Fernández Vilaboa, J.M., *Weak Hopf algebras and weak Yang-Baxter operators*, preprint (2006).
- [4] G. Böhm, *Doi-Hopf modules over weak Hopf algebras*, Comm. in Algebra, 28 (2000) 4687-4698.
- [5] G. Böhm, F. Nill, K. Szlachányi, *Weak Hopf algebras, I. Integral theory and C^* -structure*, J. of Algebra, 221 (1999) 385-438.

*Joint work with J.N. Alonso Álvarez and J.M. Fernández Vilaboa.

- [6] S. Majid, *Cross products by braided groups and bosonization*, J. of Algebra, 163 (1994) 165-190.
- [7] D.E. Radford, *The structure of Hopf algebras with a projection*, J. of Algebra, 92 (1985) 322-347.
- [8] M. Takeuchi, *Survey of braided Hopf algebras*, Contemporary Math., 267 (2000) 301-323.

The categorification of linear theories is presentation-independent

Miles Gould

The definition of pseudo-algebra for a 2-monad is equivalent, in the case of a strongly regular algebraic theory, to a definition stated in terms of factorization systems. This allows us to define a general categorification for strongly-regular theories equipped with a presentation (recovering, for instance, the theory of classical monoidal categories) and show that this categorification is independent (up to equivalence) of the chosen presentation of the theory.

This definition, and the presentation-independence theorem, can be extended to theories which are described by symmetric operads (“linear” theories); starting with the standard presentation of commutative monoids, for instance, we recover the theory of symmetric monoidal categories. Interestingly, in this case the algebras for our categorified theory are no longer pseudo-algebras for the original theory.

On Galois structures in algebraic categories

Marino Gran

New examples of categorical Galois structures [6] have been discovered and studied in various algebraic contexts in the last years.

In this talk we first place ourselves in the general context of *factor permutable categories* [4], and show how the nice centrality properties of these categories allow one to describe the corresponding Galois coverings.

In the second part we restrict ourselves to *homological categories* [1]. The Galois structures arising from various kinds of fibered reflections [2] will be considered, and examples of coverings will be given in the categories of groups, topological groups [5] and crossed modules.

Finally, we shall focus our attention on *semi-abelian categories* [7]. By building a sequence of Galois structures from any Birkhoff subcategory of a semi-abelian category, we shall explain how useful the characterization of the coverings is to get higher Hopf formulae for homology (joint work with T. Everaert and T. Van der Linden [3]).

REFERENCES

- [1] F. Borceux and D. Bourn, *Mal'cev, Protomodular, Homological and Semi-Abelian Categories*, Math. and its Appl. 566, Kluwer Academic Publishers (2004).
- [2] D. Bourn and M. Gran, *Torsion theories in homological categories*, J. Algebra 305 (2006), 18-47.
- [3] T. Everaert, M. Gran and T. Van der Linden, *Higher Hopf formulae for homology via Galois Theory*, math.AT/0701815, January 2007.
- [4] M. Gran, *Applications of Categorical Galois Theory in Universal Algebra*, in Galois Theory, Hopf Algebras and Semiabelian Categories, The Fields Institute Communications Series, American Mathematical Society, vol. 43 (2004), 243-280.
- [5] M. Gran and V. Rossi, *Torsion theories and Galois coverings of topological groups*, J. Pure Appl. Algebra 208 (2007), 135-151.
- [6] G. Janelidze, *Pure Galois Theory in Categories*, J. Algebra 132 (1990), 270-286.
- [7] G. Janelidze, L. Márki, and W. Tholen, *Semi-abelian categories*, J. Pure Appl. Algebra 168 (2002), 367-386.

Cospans and weak cubical categories in Algebraic Topology

Marco Grandis

Topological cospans and their composition, by pushout, appear in the theories of tangles, ribbons, cobordism, etc. Various algebraic invariants have been introduced for their study. We show how (co)homotopy and (co)homology functors can be used to provide new algebraic invariants, in the less usual form of spans, or cospans, or relations between sets, or groups, or abelian groups. Under suitable restrictions, such functors can be ‘linearised’ (with values in finite dimensional vector spaces), and give – for instance – topological quantum field theories for manifolds and cobordisms.

We end with introducing n -cubical cospans $\Lambda^n \rightarrow X$ (in a category X with pushouts), where Λ is the ‘formal cospan’ category. These diagrams form a ‘weak cubical category’, i.e. a cubical set with compositions in all directions, which behave categorically in a weak sense, up to suitable comparisons. This structure will be used for the higher dimensional versions of the previous invariants.

Stability for pseudomonoids

Nick Gurski

The Stabilization Hypothesis of Baez and Dolan [1] states, amongst other things, that the sequence of structures

$(n + k + 1)$ -dimensional categories with a single 0-, 1-, ..., k -cell

eventually stabilizes. When $k = 0$, the objects in question should be monoidal n -categories, and as k grows larger they should become more and more commutative. At the heart of the Stabilization Hypothesis lies a higher dimensional version of the Eckman-Hilton argument. We will discuss a 2-dimensional version of this kind of stability result as well as 2-dimensional analogues of the Eckman-Hilton argument. More specifically, we will give the statement “symmetric pseudomonoids internal to a symmetric monoidal bicategory are stable” a precise meaning while recovering some familiar theorems along the way.

REFERENCES

- [1] John Baez and James Dolan, *Higher-dimensional algebra and topological quantum field theory*, Jour. Math. Phys. 36 (1995) 6073-6105.

Placed composition in higher dimensional categories

Victor Harnik ^{*}

The notion of *multicategory* has been introduced by Lambek [1] and extended in [2]. A multicategory C has a set of *objects* $O = O(C)$ and a set of *arrows* $A(C)$ such that each arrow has a *source*, which is a finite sequence of (not necessarily distinct) objects and a *target* which is an object. If the target of an arrow v matches the object x_r that occurs in *place* r in the source of another arrow u then we have the arrow $u \circ_r v$, the *placed composition* of u and v at place r . We define, in an obvious way, when is a multicategory C , with set of objects O , *free* with set of generators $J \subseteq A(C)$.

We now turn to higher dimensional categories. Let an n -category \mathbf{B} be a *free* extension of the $(n - 1)$ -category \mathbf{A} with a set of generators I . By this we mean that \mathbf{A} is the $(n - 1)$ th truncation of \mathbf{B} , I is a set of n -cells and the obvious universal property holds. We denote this situation by $\mathbf{B} = \mathbf{A}[I]$ and let's remark that each n -cell of \mathbf{B} *depends* on a finite sequence of generators $\in I$. If the $(n + 1)$ -category \mathbf{X} is an extension of \mathbf{B} then we say that an $(n + 1)$ -cell u of \mathbf{X} is *may-to-one* if its codomain $cu \in I$. We define, in a natural way, the *multicategory* $C_{\mathbf{X}}$ of *many-to-one cells* of \mathbf{X} with set of objects I , and with arrows the many-to-one $(n + 1)$ -cells such that the source of u is the sequence of elements of I on which the domain du depends. Our main result is concerned with this setup, when \mathbf{X} itself is a free extension of \mathbf{B} with a set of many-to-one generators.

Theorem 1. If $\mathbf{B} = \mathbf{A}[I]$ and $\mathbf{X} = \mathbf{B}[J]$ where J is a set of *many-to-one* $(n + 1)$ -cells of \mathbf{X} , then $C_{\mathbf{X}}$ is a free multicategory with set of generators J .

As a corollary, we obtain a result that links the notions of *many-to-one computads* on one hand and *multitopic sets* on the other. A many-to-one computad is an ω -category \mathbf{X} such that for each n , the $(n + 1)$ th truncation of \mathbf{X} is a free extension of its n th truncation with a set of generators that are many-to-one cells. A multitopic set is, roughly speaking, a sequence $\langle C_n \rangle$ of free multicategories, such that for each n , the generators of C_n are the objects of C_{n+1} . This notion has been studied extensively in [2].

Theorem 2. The category of multitopic sets is equivalent to the category of many-to-one computads.

^{*}Joint work with Michael Makkai and Marek Zawadowski.

REFERENCES

- [1] J. Lambek, *Deductive systems and categories II*, in: P.Hilton(Ed.), Category Theory, Homology Theory and their applications, Lecture Notes in Math. 86, Springer, (1969) 76-122.
- [2] C.Hermida, M. Makkai and J. Power, *On weak higher-dimensional categories I,1-3*, J. Pure and Appl. Algebra 153 (2000) 221-246, 157 (2001) 247-277, 166 (2002) 83-104.

Orthogonality and Injectivity Logics for proper classes of morphisms

Michel Hébert *

Completeness of orthogonality and injectivity logics can be seen as refined versions of two classical problems: the Orthogonal Subcategory Problem and the Small Object Argument. In [1] and [2] we defined these logics on morphisms in appropriate categories, and proved various completeness results of the form

$$H \models f \iff H \vdash f .$$

There, given an object A and a class of morphisms H , we write $A \models H$ to mean that A is *H-injective* (resp., *H-orthogonal*), i.e., for all $h \in H$, every $k: \text{dom}(h) \rightarrow A$ factorizes (resp. uniquely) through h . A morphism f is a (*injectivity*, or resp. *orthogonality*) *consequence of H*, written $H \models f$, if for all A , we have $A \models H \Rightarrow A \models f$. We say that f is a *formal consequence of H*, written $H \vdash f$, if f can be obtained from the members of H by the utilization of some (categorical) “deduction rules” defined there.

Here, we obtain new such completeness results, focusing on proper classes H of morphisms. As a particular consequence, Kelly’s result ([4], Theorem 10.2) on the existence of constructive reflectors in orthogonal subcategories with respect to classes “mostly” made of epis, is extended in various ways. Completeness theorems for injectivity logic are also obtained, showing in particular that the Small Object Argument holds for classes mostly made of epis (resp. strong epis) in locally bounded (resp. locally presentable) categories, but proving, rather surprisingly, that this does not hold in locally ranked categories. This adds to the main result in [3]. The wide picture is further clarified through counterexamples and some intriguing open problems, contrasting the situation in the three types of categories mentioned above, as well as between the orthogonality and the injectivity cases.

REFERENCES

- [1] Adámek, J., Hébert, M., Sousa, L., *A logic of injectivity*, submitted.
- [2] Adámek, J., Hébert, M., Sousa, L., *A logic of orthogonality*, Archivum Mathematicum 42 (2006) 309-334.

*Joint work with Jiří Adámek and Lurdes Sousa.

- [3] Adámek, J., Herrlich, H., Rosický, J., Tholen, W., *On a generalized small-object argument for the injective subcategory problem*, Cah. Topol. Gém. Différ. Catég., 43 (2002) 83-106.
- [4] Kelly, G.M., *A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on*, Bull. Austral. Math. Soc. 22 (1980) 1-83.

Universality of coherence

Claudio Hermida

The coherence theorem for pseudo-algebras is shown here as *freely forcing (strict) algebras to be invariant under equivalence*. The precise formulation involves (free) relative 2-fibrations: the forgetful 2-functor from pseudo-algebras to the base is the free **Equ**-fibration over the forgetful 2-functor for strict algebras. We exhibit the corresponding result for lax algebras, involving adjoint retracts instead of equivalences, and consider various homotopical implications.

REFERENCES

- [1] C. Hermida, *Some properties of **Fib** as a fibred 2-category*, JPAA (1999) 59(1):1-41.
- [2] C. Hermida, *From coherent structures to universal properties*, JPAA (2001) 165(1):7-61.
- [3] S. Lack, *Homotopy-theoretic aspects of 2-monads*, arXiv:math.CT/0607646 (2006).

Categorical groups for exterior spaces

L.J. Hernández *

The category of exterior spaces is a (co)complete extension of the category of spaces and proper maps which is often used to study non-compact spaces. This category admits some closed model structures by taking weak equivalences associated to homotopy groups induced by either the semi-open interval or by the space of natural numbers. As in standard homotopy theory an interesting technique to solve some important questions is the construction of algebraic models for the category of exterior n -types.

In this communication, we would like to present three possible algebraic models for exterior 2-types: pro-categorical groups and the exterior categorical groups induced by the semi-open interval and by the set of natural numbers.

A long exact sequence gives some relations between all these categorical groups and the higher dimensional analogues associated to an exterior space with a base ray and the relation between pro-categorical groups and exterior categorical groups will be also analyzed. The aim of our work is to find adequate algebraic models for 2-types of exterior spaces with one Freudenthal end. Another objective is the adaptation of all these exterior categorical groups to give algebraic models of 2-types in shape, strong shape and proper homotopy theories.

REFERENCES

- [1] J. García Calcines, M. García Pinillos, L.J. Hernández, *A closed model category for proper homotopy and shape theories*, Bull. Aust. Math. Soc. 57 (1998) 221-242.
- [2] J. I. Extremiana, L.J. Hernández, M.T. Rivas , *Postnikov factorizations at infinity*, Top. Appl. 153 (2005) 370-393.
- [3] A. R. Garzón, J. G. Miranda, A. Del Río, *Tensor structures on homotopy groupoids of topological spaces*, International Mathematical Journal 2 (2002) 407-431.

*Joint work with A. Del Rio and M.T. Rivas.

Topological spaces, categorically

Dirk Hofmann *

Since F.W. Lawvere's famous 1973 paper [4] it is well-known that (generalised) metric spaces may be seen as examples of V -categories for $V = [0, \infty]$, "so that enriched category theory can suggest new directions of research in metric space theory and conversely". Thanks to M. Barr [1] we know that topological spaces can be presented as categories as well, by interpreting the convergence relation $\mathfrak{x} \rightarrow x$ between ultrafilters and points of a topological space X as arrows in X . Indeed, in this talk we wish to show that general categorical notions and results are meaningful and helpful in topology.

First of all we have to agree on a setting which incorporates both above mentioned sources and which enables us to develop the principal constructions and results. In particular our work on completeness [2] suggested that the concept of a *topological theory* as introduced in [3] fulfils these requirements. Here a topological theory is a triple $\mathcal{T} = (\mathbb{T}, V, \xi)$ consisting of a monad $\mathbb{T} = (T, e, m)$, a quantale $V = (V, \otimes, k)$ and a map $\xi : TV \rightarrow V$ compatible with the monad and the quantale structure. Given such a theory \mathcal{T} , a \mathcal{T} -category is defined as a pair (X, a) with "hom-objects" $a : TX \times X \rightarrow V$ and with "identity" and "composition", i.e. such that

$$k \leq a(e_X(x), x) \quad \text{and} \quad T_\xi a(\mathfrak{X}, \mathfrak{x}) \otimes a(\mathfrak{x}, x) \leq a(m_X(\mathfrak{X}), x)$$

where $\mathfrak{X} \in TTX$, $\mathfrak{x} \in TX$ and $x \in X$, and T_ξ is the induced extension of the Set-functor T to V -relations. Accordingly, a \mathcal{T} -functor $f : (X, a) \rightarrow (Y, b)$ is a Set-map $f : X \rightarrow Y$ such that

$$a(\mathfrak{x}, x) \leq b(Tf(\mathfrak{x}), f(x))$$

for all $\mathfrak{x} \in TX$ and $x \in X$. Clearly, for each quantale V we can consider the identity monad and choose ξ as the identity map, so that we obtain V -categories and V -functors. On the other hand, for the theory \mathcal{U} consisting of the ultrafilter monad, the two-element Boolean algebra 2 and ξ (essentially) the identity map, \mathcal{U} -categories are precisely topological spaces and \mathcal{U} -functors continuous maps.

In this setting, and guided by notions and results of enriched category theory, we study

1. lax Hopf monads and monoidal structures,
2. the dual of a \mathcal{T} -category,

*The talk is based on a joint work with M.M. Clementino and W. Tholen.

3. \mathcal{T} -modules,
4. the Yoneda Lemma,
5. Lawvere-completeness (resp. Cauchy completeness),
6. Morita equivalence.

REFERENCES

- [1] Michael Barr, *Relational algebras*, In *Reports of the Midwest Category Seminar, IV*, pages 39–55, Lecture Notes in Mathematics, Vol. 137. Springer, Berlin, 1970.
- [2] Maria Manuel Clementino and Dirk Hofmann, *Lawvere completeness in Topology*, Technical report, University of Aveiro, 2007; [arXiv:math.CT/0704.3976](https://arxiv.org/abs/math/0704.3976).
- [3] Dirk Hofmann, *Topological theories and closed objects*, Technical report, University of Aveiro, 2006.
- [4] F. William Lawvere, *Metric spaces, generalized logic, and closed categories*, Rend. Sem. Mat. Fis. Milano, 43:135–166 (1974). Also in: Repr. Theory Appl. Categ. 1:1–37 (electronic), 2002.

Fibrations in Logic

Martin Hyland

The categorical approach to logic based on fibrations is now well established. Its use for example to analyse Realizability Toposes has been very effective. However most workers restrict their attention to preordered fibrations. The idea that more general fibrations provide a mathematical theory of proofs is explained already in the introduction to Lawvere's influential paper "Adjointness in Foundations". This talk will focus on the value of the more general point of view.

It turns out that (as suggested by Lawvere) many non-standard logics of a pre-ordered kind are best regarded as the pre-ordered reflection of a richer fibred category of types. While this does not itself give much insight, it is also the case that constructions on non-standard logics seem much clearer at the level of types. I shall give examples of this 'Propositions from Types' approach and use it in particular to describe variants of the Dialectica Interpretation of Gödel.

Ideals in semi-abelian categories

George Janelidze

We discuss the relationship between Magari-Ursini ideals in universal algebra and normal monomorphisms in categorical algebra established in joint work with L. Márki and A. Ursini [2]. Using internal object actions [1] we make further comparisons between categorical concepts related to Bourn protomodularity and various concepts in universal algebra independently invented by A. Ursini and his collaborators.

REFERENCES

- [1] F. Borceux, G. Janelidze, and G. M. Kelly, *Internal object actions*, Commentationes Mathematicae Universitatis Carolinae 46, 2 (2005) 235-255.
- [2] G. Janelidze, L. Márki, and A. Ursini, *Ideals and clots in universal algebra and in semi-abelian categories*, Journal of Algebra 307, 1 (2007) 191-208.

Relative semi-abelian categories

Tamar Janelidze

We introduce a relative semi-abelian category as a pair (\mathbf{C}, \mathbf{E}) , where \mathbf{C} is a pointed category with finite limits, and \mathbf{E} is a class of normal epimorphisms in \mathbf{C} satisfying certain conditions, stronger than those defining a relative homological category [2]. In the absolute case, i.e. when \mathbf{E} is the class of all regular epimorphisms in \mathbf{C} , the pair (\mathbf{C}, \mathbf{E}) is relative semi-abelian if and only if \mathbf{C} is semi-abelian in the sense of [1]. Another extreme case is the “trivial” one, where \mathbf{E} is the class of isomorphisms in \mathbf{C} , and then (\mathbf{C}, \mathbf{E}) is always relative semi-abelian. Some results of [1] on the equivalence of the so-called old-style and new-style axioms are extended to the relative case.

REFERENCES

- [1] G. Janelidze, L. Márki, and W. Tholen, *Semi-abelian categories*, Journal of Pure and Applied Algebra 168 (2002) 367-386.
- [2] T. Janelidze, *Relative homological categories*, Journal of Homotopy and Related Structures 1 (2006) 185-194.

Cover relations on categories

Zurab Janelidze

We define a *precover relation* on a category \mathbb{C} to be any binary relation \sqsubset on the class of morphisms of \mathbb{C} such that if $f \sqsubset g$ then f and g have the same codomain. If \mathbb{C} has finite limits, then a precover relation on \mathbb{C} is in some sense the weakest structure on \mathbb{C} which allows to interpret in \mathbb{C} first order sentences of the form $\forall_{x_1, \dots, x_n} (\phi \Rightarrow \psi)$, where ϕ and ψ are formulas made up of atomic ones using the connective \wedge and the quantifier \exists . One of the uses of categorical interpretations of such first order sentences is that it allows to reformulate purely categorically certain term conditions of Universal Algebra. For instance, the condition that defines a Mal'tsev category can be obtained in this way from the term condition that defines a Mal'tsev variety.

A *cover relation* is a precover relation having the following two properties: (i) if $f \sqsubset g$ and f, g are composable with h , then $hf \sqsubset hg$, (ii) if $f \sqsubset g$ and e is composable with f then $fe \sqsubset g$. We show that by imposing natural axioms on cover relations we obtain, on the one hand, a special class of factorization systems, and, on the other hand, a special class of monoidal structures.

Homotopy, naturality and homotopy naturality

Michael Johnson

As homotopy theory and category theory influence one another in contemporary studies as diverse as concurrency theory and the development of higher categories, we study the basic notions of both homotopy theory and category theory in a common framework. The connections are surprising in the light of the early history of category theory and its early applications in algebraic topology, and illuminating when applied to modern problems in higher dimensional category theory.

The connection between equivalence relations and subgroups

Toby Kenney *

It is well-known that the lattice of equivalence relations on a set embeds in the lattice of subgroups of the permutation group of that set, while the lattice of subgroups of a group embeds in the lattice of relations on its underlying set [1].

In fact, both these lattices are sublattices of involutive quantales – the quantale of all relations on X , and the quantale of all subsets of G . In their respective quantales, these lattices are both the sublattice of reflexive, symmetric idempotent elements. Therefore, a quantale embedding between these quantales would induce a \vee -semilattice embedding between the lattices, and a \wedge -preserving quantale embedding would induce a lattice embedding.

These quantales can both be constructed in the following way: take a category \mathcal{C} , and take the subsets of its set of morphisms. This is a quantale, with meet and join given by intersection and union respectively, and multiplication given by the set of morphisms that are formed as the composite of a morphism in the first set with a morphism in the second set. The quantale of relations on X is obtained by applying this construction to the indiscrete category on X , while the quantale of subsets of G is obtained by applying this construction to G considered as a 1-object category. The embedding of the quantale of subsets of G into the quantale of equivalence relations of its underlying set is then induced by the functor from the coslice $*/G$ to G , where $*$ is the object of G .

REFERENCES

- [1] G. Birkhoff, *On the Structure of abstract algebras*, Proc. Cambridge Phil. Soc. 31 (1935) 433-454.

*Joint work with Geoff Cruttwell, Robert Paré and Richard Wood.

Loops from the (semi-abelian) categorical point of view

Rudger Kieboom

Up to now, the variety of loops (= quasigroups with identity) has been studied mainly from the point of view of universal algebra. It is well known that this variety is a Mal'cev variety. From the categorical point of view, more can be said: loops and loop homomorphisms form a semi-abelian category. It is therefore of interest to investigate in this category some of the notions, which have recently been developed in the semi-abelian framework, like commutators, central extensions, semi-direct products, (co)homology, homotopy, and so on, and to compare them with similar notions introduced for loops by people working in (universal) algebra.

Some recent advances in Synthetic Differential Geometry

Anders Kock

Some of these recent advances have been connected with aspects of the *neighbour relation*, deriving from the “first neighbourhood of the diagonal”, as known from algebraic geometry.

The neighbour relation gives rise to a simplicial set, and hence to “combinatorial differential forms” in the sense of the author, and of Breen and Messing. But the neighbour notion also gives rise to a *cubical* set, consisting of “infinitesimal parallelepipeda”. This cubical set provides an alternative basis for a theory of combinatorial differential forms, which is well suited to be generalized into a theory of higher connections with values in multiple groupoids.

For instance, the curvature of a 1-connection in a groupoid is a (flat) 2-connection in the associated double groupoid.

The tool behind the construction of infinitesimal parallelepipeda is the possibility of forming affine combinations of mutual neighbour points in a manifold.

Comprehending structure types and stuff types

Jürgen Koslowski

Graph comprehension provides a means of reconciling two different views of labeled transition systems: as (faithful) graph morphisms into the graph, or better category, of labels, and as graph morphisms from the later into the bicategory **Spn** of spans (**Rel** of relations) over sets. It specializes to a number of known equivalences, for instance between fibrations into a category B and contravariant pseudo-functors from B into the bicategory **Cat**, or between functions into a set C and C -indexed families of sets.

Species of structures, on the other hand, were introduced 1981 by Joyal in order to “categorify” combinatorics. They can be viewed as functors from the groupoid E of finite sets into itself, or into the category of finite sets. Viewed from a different angle, they correspond to so-called “structure types”, namely faithful functors from some groupoid into E . In 2001 the later concept was generalized to “stuff types” by Baez and Dolan by dropping the assumption of faithfulness.

Building upon the categorical analysis of stuff types started by Simon Byrne in 2005, we show how stuff types fit into the framework of graph comprehension, namely by specializing to symmetric graphs, respectively, symmetric spans. The corresponding transition systems account for reversible computations, like those that can be performed by a quantum computer. Furthermore, various types of morphisms between labeled transition systems that graph comprehension supplies can now be specialized to stuff types, in particular simulations as well as modules.

Girard structures on discrete quantales

David Kruml ^{*}

W^* -algebras are supposed to model non-commutative measure theory in the same way that C^* -algebras model non-commutative topology. One might therefore expect that the quantale-theoretic *spectrum* of a W^* -algebra should admit some form of complementation. The most obvious sense in which this could happen is if the spectrum were to admit a cyclic dualizing element, but this only occurs in the finite-dimensional case.

We propose to solve this problem by introducing a concept of *coupled* pair of quantales. We show that every W^* -algebra gives rise to a *Girard couple* (i.e., coupled quantales equipped with a cyclic dualizing element). Another example of a Girard couple arising in quantale theory comes from considering $S \otimes S^*$ and $\text{Hom}(S, S)$ for an arbitrary sup-lattice S .

We also show how to construct a single Girard quantale from a Girard couple, and applying this construction to the latter example, we obtain a Girard quantale whose sublattice of right-sided elements is isomorphic to S .

Most of our results appear as applications of theory of $*$ -autonomous categories on sup-lattices.

^{*}Joint work with Jeff Egger.

Lawvere 2-theories

Steve Lack *

A fundamental result in category universal algebra is the correspondence between Lawvere theories and finitary monads on **Set**. Power showed how to generalize this to the enriched context, to obtain a correspondence between *Lawvere V-theories* and finitary V-monads on (a suitable monoidal category) V. Nishizawa and Power described a further generalization to deal with finitary V-monads on any locally finitely presentable V-category K. In each case the theories involve limits more general than the usual finite products. I will outline the general theory, before looking in detail at what happens in the special case V=Cat of 2-theories and 2-monads, where various lax and pseudo issues arise.

*Joint work with John Power.

Strong Nullstellensatz for Euler continua in cohesive toposes

F. William Lawvere

The axiomatic theory of cohesive toposes is based on common features of algebraic, analytic, and smooth geometry/topology and of combinatorial toposes like simplicial sets (1950) and reflexive graphs. The aim is to make explicit a background for dynamical models of the motion of bodies and waves. Such toposes E are defined over other toposes K considered to be (relatively) non-cohesive, for example K might be “the” Galois or “the” Cantor topos; there is given a product-preserving functor from E to K considered to “count cohesive pieces” which determines a string of right adjoints pieces, discrete, points, codiscrete. The assumption “the natural map from discrete to codiscrete in E is monic” is a (weak) Nullstellensatz because it is equivalent to the epimorphicity of the corresponding map from points to pieces in K . In commutative algebra a stronger Nullstellensatz (in the spirit of Birkhoff) affirms that the homomorphisms from a given function algebra to “infinitesimal” algebras are jointly monic; a dual idea is that for certain spaces R in a topos (but not all), an equation between two maps from X to R would follow from their equality on each infinitesimal figure in X . Of course nilpotents and ATOMs play key roles, but in the present context a natural broad definition of “infinitesimal” captures those spaces for which every piece has a unique point. (For example, all arrows in an infinitesimal reflexive graph are loops). Every space then has a maximal infinitesimal subspace, which is its substance or canonical intensive quality (in the sense of my CT06 contribution, “Axiomatic Cohesion” to appear in TAC). A space R satisfies the strong Nullstellensatz relative to this notion of infinitesimal if substantially equal R -valued functions are equal. The spaces satisfying this strong Nullstellensatz are closed under exponentiation by infinitesimals: given that $R = B^A$ where A is infinitesimal, if B satisfies the strong Nullstellensatz, then so does R . Euler described reals as ratios of infinitesimals; because spaces of ratios are retracts of map spaces, it follows that the strong Nullstellensatz holds for Euler continua in E if it holds for the infinitesimal spaces themselves.

Jónsson–Tarski toposes

Tom Leinster

In 1961, Jónsson and Tarski defined a certain algebraic theory with the unusual feature that its category of algebras is a topos. This is the classical Jónsson–Tarski topos. In fact, it is just one member of a whole family of ‘Jónsson–Tarski toposes’; there is one such topos for each small category \mathbf{A} and two-sided \mathbf{A} -module M . I will introduce this family and place it in a wider context. I will also state some open questions.

Congruences on algebras over the triple from Banach Space unit discs

Fred Linton

The topic will disclose a hodgepodge of observations regarding closure properties of congruence classes under the congruence relations arising in the category of algebras over the monad that arises from the UnitDisc/Discrete L^1 adjunction between the categories of BanachSpaces and of Sets.

Computads and weak higher dimensional categories

Michael Makkai

Computads were introduced by Ross Street. They were re-introduced and renamed as polygraphs by Albert Burroni (*Theoretical Computer Science* 115 (1993)). The following two papers reproduce the basic definitions, and are relevant to my subject in essential ways: [Jacques Penon, “Approche polygraphique des inf-categories non strictes”, *Cahiers TGDC*, vol. XL-1 (1999)]; and [Francois Metayer, “Resolutions by polygraphs”, *TAC* 11, no.7, 2003].

Multitopic categories are a proposal for the still-contested general concept of weak higher dimensional category (WHDC). References are: [C. Hermida, M. Makkai and J. Power *JPAA* 153 (2000), 157 (2001) and 166 (2002)]; and [M. Makkai, “The multitopic omega-category of all multitopic omega-categories”, 1999 and 2004, www.math.mcgill.ca/makkai].

The connection between computads and multitopic categories is given by the fact that the category of multitopic sets is equivalent to a full subcategory of the category of computads, namely the one whose objects are the so-called many-to-one computads. This fact is proved in the paper [V. Harnik, M. Makkai and M. Zawadowski, “Multitopic sets are the same as many-to-one computads”, June 2002, www.math.mcgill.ca/makkai].

A natural question is whether one could use arbitrary computads in place of the many-to-one ones in a new definition of WHDC’s. The answer is “yes”; and in fact, the definition becomes simpler because of the added flexibility gained by removing the “many-to-one” restriction from computads. The new definition uses a notion of equivalence of cells within an omega-category that is very natural but seems not to have been mentioned in the literature; the notion is called coinductive equivalence for a good reason. The interesting thing that happens is that (Theorem) the new notion of WHDC thus obtained turns out to be equivalent to that of ordinary, strict omega category. Thus, the new definition is a bad one, since we know that even 3D WHDC’s (tricategories) are not equivalent to 3-categories.

The proof of the theorem uses a very interesting “path-object” construction in Francois Metayer’s above mentioned paper. I take the theorem to be an important justification for the concept of multitopic category. It also seems to be a first one among results that are highly desired (at least by me), ones that rigorously compare proposed notions of WHDC’s, and thus show that the game of defining the concept of WHDC is not completely rule-free.

Closed multicategory of A_∞ -categories

Oleksandr Manzyuk ^{*}

Over the last decade, A_∞ -algebras and A_∞ -categories have enjoyed a resurgence of interest due to applications in non-commutative geometry, representation theory, and physics. Our interest in A_∞ -categories is due to recent developments in homological algebra (Bondal and Kapranov, Drinfeld, Keller, . . .). We share the confidence that large part of homological algebra can be rewritten in the language of so called pretriangulated A_∞ -categories rather than that of triangulated categories. We are developing a comprehensive theory of pretriangulated A_∞ -categories in the book in progress [1]. It is based on the observation that A_∞ -categories constitute a closed symmetric multicategory. In the talk, I am going to briefly discuss relevant notions and ideas of proof.

REFERENCES

- [1] Yuri Bespalov, V. V. Lyubashenko, and Oleksandr Manzyuk, *Closed multicategory of pretriangulated A_∞ -categories*, book in progress, 2007.

^{*}Joint work with Yuri Bespalov and Volodymyr Lyubashenko.

Constructively completely distributive lattices in presheaf categories

Francisco Marmolejo ^{*}

An ordered set L is a sup lattice if and only if the down-segment embedding of L into its complete lattice of down-sets, $\downarrow : L \rightarrow DL$, has a left adjoint $\vee : DL \rightarrow L$. Since [3] a complete lattice has been said to be constructively completely distributive (CCD) if $\vee : DL \rightarrow L$ has itself a left adjoint. In [4] there is a characterization of sup-lattices in terms of functors $L : \mathbf{C}^{op} \rightarrow \mathbf{sup}$ (**sup** the category of sup-lattices and sup-preserving arrows with respect to the base topos **Set**). We produce a similar characterization of CCD lattices in terms of functors $L : \mathbf{C}^{op} \rightarrow \mathbf{ccd}$ (**ccd** the category of ccd lattices and arrows that preserve both, suprema and infima). Since L takes values in **ccd** we have, for every $f : B \rightarrow C$ in **C** an adjoint string $\vee_f \dashv Lf \dashv \wedge_f$. This characterization hinges on the (surprising) appearance of a right adjoint to \wedge_f and on a condition called complete Frobenius reciprocity.

REFERENCES

- [1] G. S. H. Cruttwell, *A study of CCD lattices in a functor category*, Master's degree thesis, Dalhousie University (2005).
- [2] G. S. H. Cruttwell and F. Marmolejo and R. J. Wood, *CCD lattices in presheaf categories*, Theory and Applications of Categories 18 No. 6 (2007) 157–171.
- [3] B. Fawcett and R. J. Wood, *Constructive complete distributivity I*, Math. Proc. Cambridge Philos. Soc. 107 (1990), 81–89.
- [4] A. Joyal and M. Tierney, *An extension of the Galois theory of Grothendieck*, Mem. Amer. Math. Soc. 51 (1984), no. 309,
- [5] R. Rosebrugh and R. J. Wood, *Constructive complete distributivity II*, Math. Proc. Cambridge Philos. Soc. 110 (1991), 245–249.
- [6] R. Rosebrugh and R. J. Wood, *Constructive complete distributivity III*, Canad. Math. Bull. Vol 35 (1992) no. 4 537–547.

^{*}Joint work with G. S. H. Cruttwell and R. J. Wood.

The tricategory of internal bicategories in Ab

Nelson Martins Ferreira

We describe the tricategory of bicategories internal to abelian groups; specifically we calculate internal bicategories, homomorphism of bicategories, strict and pseudo natural transformations and also modifications with the respective formulas for composition. The results are, as expected, closely related to homotopies and chain complexes as pointed out by D. Bourn in his paper [1] seventeen years ago.

REFERENCES

- [1] D. Bourn, *Another denormalization theorem for the abelian chain complexes*, J. Pure Appl. Algebra 66 (1990) 229–249.

Läuchli's completeness theorem from a topos-theoretic perspective

Matías Menni *

We prove a variant of Läuchli's completeness theorem for intuitionistic predicate calculus [3]. (See also [2], [1], [5] and [4].) The formulation of the result relies on the observation (due to Lawvere) that Läuchli's theorem is related to the logic of the canonical indexing of the atomic topos of \mathbf{Z} -sets. We show that the process that transforms Kripke-counter-models into Läuchli-counter-models is (essentially) the inverse image of a geometric morphism. Completeness follows because this geometric morphism is an open surjection.

REFERENCES

- [1] V. Harnik and M. Makkai, Lambek's categorical proof theory and Läuchli's realizability, *The Journal of Symbolic Logic*, 57(1), 200–230, 1992.
- [2] A. Kock. *On a theorem of Läuchli concerning proof bundles*, Unpublished, August 1970.
- [3] H. Läuchli, An abstract notion of realizability for which intuitionistic predicate calculus is complete, In A. Kino, J. Myhill, and R. E. Vesley, editors, *Intuitionism and Proof Theory (Proc. Conf., Buffalo, N. Y., 1968)*, 227–234, North Holland, 1970.
- [4] F. W. Lawvere, Adjoints in and among bicategories, In *Logic and Algebra, Lecture Notes in Pure and Applied Algebra*, volume 180, pages 181–189. Marcel Dekker, Inc., 1996, Proceedings of the 1994 Siena conference in memory of Roberto Magari.
- [5] M. Makkai, The fibrational formulation of intuitionistic predicate logic 1: completeness according to Gödel, Kripke and Läuchli. Parts 1 and 2, *Notre Dame Journal of Formal Logic*, 34(3 and 4 resp.), 1993.

*The research reported here was funded by Conicet, ANPCyT and Lafia.

A Geometry for Diagrammatic Computads

Thorsten Palm

This talk is a preliminary report on a project suggested in [3]. The goal is to give a (combinatorially) geometric description of a rather large class of computads (= free ∞ -categories with distinguished generators; all higher-dimensional categories mentioned here are strict), much as [3] does for the class of many-to-one computads. Sources of inspiration were the articles [1], [6] and [5], in which computads are constructed from certain “complexes”. In each of them a condition of global acyclicity is imposed (allowing composition to be represented by plain set-theoretic union), which the present work will avoid.

The central notion here is that of a *hypotopic set*. It differs from the notion of an oriented polytopic set in that facets of a cell are allowed (and by further constraints in fact demanded) to occur in arbitrary lower dimensions (hence the prefix ‘hypo’). Simplicial and cubical sets as well as dendrotopic sets all can be interpreted as instances of hypotopic sets.

Legitimate cell configurations in a hypotopic set, the *tissues*, form a category, which the expected pasting operations render ω -*categorial*. (So the ω -category laws, including the boundary ones, are satisfied in a weakened form.) Attention is restricted to those hypotopic sets that are *well formed* in that all boundary tissues of cells are well formed, that is, generated from (lower-dimensional) cells by the pasting operations. The problem now is to find “nice” conditions that, while being reasonably inclusive, ensure that (a) the (∞ -categorial) full subcategory of well-formed tissues is essentially discrete — then its components form an ∞ -category — and (b) this ∞ -category is free.

A certainly nice condition is *semiregularity*: the absence of “singular” faces in the source tissues of cells. It can be subdivided according to the *ranks* of these faces. (Facets have rank 0, ridges have rank 1, and so on.) Rank-1 semiregularity is sufficient for (a) alone. As for the stated problem (regarding both (a) and (b)), the presenter’s best proven solution to date is rank-1 and -2 semiregularity in conjunction with “translationality” of “enhanced” dual hemigraphs. It includes all dendrotopic sets, as well as simplicial and cubical sets up to dimension 3 (but, unfortunately, not beyond). The presenter’s best conjectural (and as yet unrefuted) solution is full semiregularity on its own. It would include all dendrotopic, simplicial and cubical sets. It would also include, for instance, A. J. Power’s 4-dimensional (counter-)example.

The category of well-formed hypotopic sets satisfying (a) and (b) is in fact concretely embedded into the category of computads. The full subcategory described by

either solution is a concrete presheaf category, as can be shown using the methods of [4]. The composite result is a counterpart to the fact, expounded in [2], that the concrete category of all computads is not a presheaf category.

REFERENCES

- [1] Michael Johnson, *The combinatorics of n -categorical pasting*, Journal of Pure and Applied Algebra 62 (1989), 211–225.
- [2] M. Makkai, *The word problem for computads*, www.math.mcgill.ca/makkai.
- [3] Thorsten Palm, *Dendrotopic Sets*, Fields Institute Communications 43 (2004), 393–443.
- [4] Thorsten Palm, *Categories with Slicing*, www.ics.mq.edu.au/~palm.
- [5] Richard Steiner, *The Algebra of Directed Complexes*, Applied Categorical Structures 1 (1993), 247–284.
- [6] Ross Street, *Parity complexes*, Cahiers de Topologie et Géométrie Différentielle Catégoriques 32 (1991), 315–343. *corrigenda*: 35 (1994), 359–361.

Semistrict Tamsamani n -groupoids and connected n -types

Simona Paoli

A very important and non-trivial problem in higher category theory is that of finding coherence theorems for weak higher categorical structures. Broadly speaking, one way to formulate a coherence theorem for a weak higher categorical structure consists in saying that it is, in a suitable sense, equivalent to one in which some of the associativity and identity laws hold strictly. These structures are often called “semistrict”.

The fundamental information carried by a weak n -groupoid is its homotopy type. In low dimension, it is known that strict 2-groupoids model 2-types and Gray groupoids model 3-types. These low dimensional results had lead several people to formulate the *semistrictification hypothesis* for homotopy types: In every model of weak n -category, a weak n -groupoid should be suitably equivalent to a semistrict one.

In this talk we illustrate a semistrictification result in the higher groupoid case, valid for any n , relative to the model of weak higher category developed by Tamsamani. Our main result states that every Tamsamani’s weak n -groupoid representing a connected n -type is in a suitable sense equivalent to a semistrict one. Our semistrictification theorem is the result of a comparison between cat^n -groups and Tamsamani’s model for the path-connected case.

REFERENCES

- [1] S. Paoli, Semistrict models of connected 3-types and Tamsamani weak 3-groupoids, *Jour. of Pure and Appl. Algebra*, to appear. Available at ArXiv [math.AT/0607330](#) .
- [2] S. Paoli, Semistrict Tamsamani n -groupoids and connected n -types, ArXiv preprint (2007), [math.AT/0701655](#) .

Kan extensions for double categories

Robert Pare ^{*}

Two dimensional category theory is category theory based on Cat , the category of categories. One of the insights provided by double category theory is that Cat should be considered as a double category with functors and profunctors as arrows. Thus it is important to understand its completeness properties. It transpires that limits are only lax functorial. Of course the whole story of limits must include Kan extensions, which are parametrized limits. We show that companions and conjoints (a kind of adjointness between horizontal and vertical arrows) are special cases of Kan extensions, and that these together with limits are sufficient for constructing Kan extensions along double functors satisfying a kind of Conduche condition. This is the best that can be expected as the right adjoint for “pulling back” along a functor appears as a special case. Thus we take the existence of such Kan extensions as our notion of completeness in double categories.

^{*}Joint work with Marco Grandis.

Category theory in a $(\mathcal{E}, \mathcal{M})$ -category

Claudio Pisani

We propose a generalization of the category of categories which is based on a factorization system $(\mathcal{E}, \mathcal{M})$, playing the role of the comprehensive factorization system on **Cat** [7].

Let \mathcal{C} be any finitely complete category with a factorization system on it. Let us call the arrows in \mathcal{E} and in \mathcal{M} “final maps” and “discrete maps” respectively; and let us denote by $\downarrow(-) : \mathcal{C}/X \rightarrow \mathcal{M}/X$ the reflection in discrete maps over X . If the map $m : M \rightarrow X$ in \mathcal{M}/X is the discrete reflection $\downarrow x$ of an “object” $x : 1 \rightarrow X$ (i.e. if M has a final “object” $e : 1 \rightarrow M$), we say that m is a “principal map”. (In **Cat**, the principal maps over X are the discrete fibrations X/x , corresponding to the representable presheaves).

If a map $p : P \rightarrow X$ in \mathcal{C} has a reflection in principal maps over X , this is called a “colimit” of p . If the discrete reflection $\downarrow p$ is already principal, then it is an “absolute” colimit of p [4, 5]. If the pullback $f^* \downarrow y$ of the principal map on $y : 1 \rightarrow Y$ along $f : X \rightarrow Y$ is itself principal, we say that the corresponding final object $1 \rightarrow f^* \downarrow y$ is a “universal arrow” from f to y . One can also naturally define categorical concepts such as “dense” or “full and faithful” maps.

This very general context seems best suited to enlighten some basic classical facts of category theory, such as the following:

1. If $e : P \rightarrow X$ is a final map in \mathcal{C} , then for any map $f : X \rightarrow Y$ the colimits of f and $f \circ e$ are the same (either existing if the other one does).
2. If a final map $e : P \rightarrow X$ has a colimit, then this is absolute and is (the reflection of) a final object of X .
3. A map $f : X \rightarrow Y$ admits a universal arrow to an object y of Y , if and only if $f^* \downarrow y \in \mathcal{M}/X$ has a colimit which is preserved by f itself.
4. If a map $f : X \rightarrow Y$ admits a universal arrow to any object of Y , then it preserves colimits.

We say that a map $\mathbf{y} : X \rightarrow \mathcal{P}X$ presents $\mathcal{P}X$ as a “power object” of X if it induces an equivalence between the category of principal maps over Y and the category of discrete maps over X . If this is the case, the “Yoneda map” \mathbf{y} is full and faithful and dense and the following hold:

1. A map $e : P \rightarrow X$ is final if and only if for any map $f : X \rightarrow Y$ the colimits of f and $f \circ e$ are the same (either existing if the other one does).

2. A colimit of $p : P \rightarrow X$ is absolute if and only if it is preserved by any map $f : X \rightarrow Y$.
3. The discrete reflection of a map $p : P \rightarrow X$ can be obtained as the colimit of $\mathbf{y} \circ p$ in $\mathcal{P}X$ (or more precisely, as its corresppective via the above equivalence).

Axioms regarding a duality functor $(-)' : \mathcal{C} \rightarrow \mathcal{C}$, exponentiability and an arrow object may be added to obtain a more faithful abstraction of **Cat**. E.g., if the object $\Omega = \mathcal{P}1$ of “internal sets” exists, and if $\mathcal{P}X = \Omega^{X'}$, then X has a “hom map” $h : X \times X' \rightarrow \Omega$. An arrow object for \mathcal{C} is a bipointed object $s, t : 1 \rightarrow 2$ such that the factorization system and its dual are generated by t and s respectively. In this case, to an “arrow” $f : 2 \rightarrow X$ of X there corresponds a morphism in \mathcal{M}/X between the principal maps on its domain and codomain; and the map $X \rightarrow 1$ is discrete if and only if X is discrete in the sense of [1, 2], since $2 \rightarrow 1$ is itself final.

As a further instance of the theory, we consider the category of posets with the usual “cofinal” mappings and the inclusions of lower sets as discrete maps, wherein Ω is the two-elements truth values poset.

REFERENCES

- [1] F.W. Lawvere, *The Category of Categories as a Fundation for Mathematics*, Proceedings of the Conference on Categorical Algebra, La Jolla, 1965, Springer, New York, 84-95.
- [2] F.W. Lawvere, *Foundations and Applications: Axiomatization and Education*, Bull. Symb. Logic 9 (2) (2003) 213-224.
- [3] R. Paré, *Connected Components and Colimits*, J. Pure Appl. Algebra 3 (1973) 21-42.
- [4] C. Pisani, *Components, Complements and Reflection Formulas*, preprint (2007), [math.CT/0701457](http://arxiv.org/abs/math/0701457) .
- [5] R. Street and R.F.C. Walters, *The Comprehensive Factorization of a Functor*, Bull. Amer. Math. Soc. 79 (2) (1973) 936-941.

Categorical aspects of K -theory for operator algebras

Radu Popescu

We present some categorical aspects of K -theories for operator algebras (K , KK and E) that have played a crucial role in the development of the field.

A compelling example is the way the question if Kasparov's bivariant K -theory is exact or not, was tackled. The negative answer was provided by Skandalis by means of an example, but meanwhile Higson had shown that such a theory exists. A concrete description have been found by Connes and Higson using asymptotic morphisms, which later have been used as the main ingredient in the proof of the Baum-Connes conjecture for amenable groups.

We present some results in the equivariant setting, and some related questions which have appeared lately.

REFERENCES

- [1] A. Connes and N. Higson, *Déformations, morphismes asymptotiques et K -théorie bivariante*, C. R. Acad. Sci. Paris Sr. I Math. 311 (1990), no. 2, 101–106.
- [2] N. Higson, *Categories of fractions and excision in KK -theory*, J. Pure Appl. Algebra 65 (1990), no. 2, 119–138.
- [3] R. Popescu, *Coactions of Hopf C^* -algebras and equivariant E -theory*, front.math.ucdavis.edu/math.KT/0410023 .
- [4] G. Skandalis, *Le bifoncteur de Kasparov n'est pas exact*, C. R. Acad. Sci. Paris Sr. I Math. 313 (1991), no. 13, 939–941.

Free constructions on double categories

Dorette Pronk *

In the process of constructing various model structures on the 2-category **DblCat** of double categories, we need various free constructions into **DblCat**, as well as some special pushouts of double categories, which turn out to have nice descriptions. I will discuss the horizontal and the double categorification, and a nice presentation for the free double category on a double derivation scheme, which extends the construction of a free double category on a double graph as described by Dawson and Paré [1].

REFERENCES

- [1] R.J.M. Dawson, R. Paré, *What is a free double category like?* J. Pure Appl. Algebra 168 (2002) 19-34.

*Joint work with Tom Fiore and Simona Paoli.

Groupoid quantales beyond étale groupoids

M. Clarence Protin ^{*}

In this talk I describe a generalization of the correspondence between inverse quantal frames and étale groupoids of [1] so as to include open (and even semiopen) groupoids. I shall give an algebraic characterization of the class of quantales that corresponds to these groupoids. This seems to include the need for a somewhat troublesome “multiplicativity” axiom that in the case of inverse quantal frames is automatic. The talk will explain this along with ongoing work meant to settle the question of whether in the more general case the multiplicativity condition is still automatic. This is based on imposing a new axiom (which at least holds for many of the groupoids arising in analysis and geometry, in particular the locally compact groupoids), and on embedding such a quantale Q into an inverse quantal frame generated by “local bisections” of Q , which among other things requires Q to be “spatial” in the sense of having enough such bisections.

REFERENCES

- [1] P. Resende, *Étale groupoids and their quantales*, Adv. Math. 208 (2007) 147–209.

^{*}Joint work with P. Resende.

Quantale-valued sets, quantale modules, and groupoid actions

Pedro Resende *

In addition to the two traditional ways of understanding sheaves on a space B , namely as local homeomorphisms $X \rightarrow B$ and as presheaves $\Omega(B)^{\text{op}} \rightarrow \text{Set}$, there is a third way, according to which sheaves are regarded as locale-valued sets $S \times S \rightarrow \Omega(B)$ [1], which has been at least implicit in several generalizations of the notion of sheaf where locales are replaced by more general quantales (or even quantaloids), namely in works by Borceux, van den Bossche, Gyllys, Mulvey, Nawaz, van der Plancke, Stubbe, Walters, Zamora Ramos, etc. This talk has two purposes. First, I will describe, for involutive quantales, a fourth elementary way of understanding sheaves, namely as quantale modules equipped with a “Hilbert basis” — this view is interesting even for sheaves on locales because it provides an (at least metaphorical) analogue of Swan’s theorem for local homeomorphisms instead of vector bundles. And I will show, using the correspondence between groupoids and quantales of [2], that examples of sheaves on quantales are obtained canonically from sheaves (on locales) equipped with a groupoid action.

REFERENCES

- [1] M.P. Fourman and D.S. Scott, Sheaves and logic, In: *Applications of Sheaves* (Proc. Res. Sympos. Appl. Sheaf Theory to Logic, Algebra and Anal., Univ. Durham, Durham, 1977), Lecture Notes in Math., 753 (1979), 302–401.
- [2] P. Resende, Étale groupoids and their quantales, *Adv. Math.*, 208 (2007), 147–209.

*Joint work with M. Protin and E. Rodrigues.

Combinatorial model categories

J. Rosický

A model category \mathcal{K} is equipped with three classes of morphisms \mathcal{C} , \mathcal{F} and \mathcal{W} called cofibrations, fibrations and weak equivalences. Cofibrations (fibrations) which are weak equivalences are called trivial cofibrations (fibrations). A model category is called *combinatorial* if \mathcal{K} is locally presentable and both cofibrations and trivial cofibrations are generated by a set of morphisms. This concept was introduced by J. H. Smith approximately 10 years ago. Unfortunately, he has not yet published proofs of a number of important results he announced. A striking example is the fact that the colimit closure of simplices in topological spaces form a locally presentable category and thus a combinatorial model category.

I will supply some missing proofs and explain what I do not understand yet. In particular, it will be shown that a colimit closure of a small full subcategory in a fibre-small topological category is always locally presentable, which covers the just mentioned case of simplices. Another proved result is that \mathcal{W} is an accessible subcategory of \mathcal{K}^\rightarrow and that, which is a new result, \mathcal{C} is a full image of an accessible functor into \mathcal{K}^\rightarrow . J. H. Smith also announced that, for a given \mathcal{C} generated by a set of morphisms, there is the smallest \mathcal{W} making \mathcal{K} a combinatorial model category. Independently, we came to the same idea with W. Tholen in 2000 and showed that it is true under set-theoretical Vopěnka's principle. Independently and in the same time, D.-C. Cisinski introduced this idea as well and proved it, without any set theory, in the special case when \mathcal{K} is a Grothendieck topos and \mathcal{C} is the class of all monomorphisms. A current status of this problem will be discussed.

A term calculus for cartesian differential categories

R.A.G. Seely *

Last year, at the Calgary CMS meeting and at the White Point CT2006 meeting, we introduced “differential categories” and “cartesian differential categories”. Differential categories are a re-development of the notion of differential following the linear-logic-inspired approach due to Thomas Ehrhard, where a comonadic structure is used to account for two types of maps (“smooth” and “linear”), with a combinator D acting on smooth maps to form linear ones in the suitable way. The distinction between differential categories and cartesian differential categories is one of perspective: in the cartesian differential case we work in an “abstract coKleisli category”, so both cartesian and monoidal tensor structure is available to us. The intention is to characterize those categories that arise as the coKleisli categories of differential categories, and whose differential structure is induced from the differential structure of the base category.

One complication of working in the abstract coKleisli setting is that the mix of cartesian and monoidal structures can make deciding coherence questions (such as proving diagrams commute) harder. We have a good technology of circuit (or string) diagrams, but they are cleaner in the monoidal context, and become less helpful with cartesian structure. A tool we are finding useful in working with such problems is a term calculus for the maps of these categories — in effect such a calculus allows us to (almost) pretend we are doing first year calculus calculations, and certainly seems to make deciding coherence questions simpler.

In the talk I will briefly review the basic definitions of differential and cartesian differential categories, with some illustrations, and then will outline the term calculus, indicate how one shows it to be complete and sound, and illustrate its use in proving diagrams commute.

This is work in progress, and the exact nature of the material presented will probably be decided at the last minute!

REFERENCES

- [1] R.F. Blute, J.R.B. Cockett, R.A.G. Seely. *Differential Categories*, Mathematical Structures in Computer Science 16(2006) 1049–1083.

*Joint work with R.F. Blute and J.R.B. Cockett.

Topology for V -categories and (T, V) -categories

Walter Tholen *

We treat a commutative unital quantale V as a symmetric monoidal-closed category and first study V -enriched categories. Being simultaneously order-enriched, a topological category over Set, and a symmetric monoidal-closed category, the category $V\text{-Cat}$ has an immensely rich structure that is best exploited when studied in conjunction with the category $V\text{-Mod}$ of V -categories and of V -(bi-)modules between them. For example, every V -category comes with a strictly categorically-motivated but topologically equally interesting closure operation that leads naturally to notions of separation and completion, as first studied by Bill Lawvere in his famous 1973 Milano paper. We will present a synopsis of the topological aspects of the $V\text{-Cat}/V\text{-Mod}$ interplay and then demonstrate which steps are needed to rescue the essential parts of the theory at the more general (T, V) -level, where now T is a Set-monad that comes with a lax extension to $V\text{-Cat}$.

*The talk is based on a joint project with M. M. Clementino and D. Hofmann.

The equivalence between Barr-Beck cotriple homology and the Brown-Ellis higher Hopf formulae

Tim Van der Linden *

We use Categorical Galois Theory to interpret cotriple homology in arbitrary semi-abelian monadic categories in terms of generalized Brown-Ellis-Hopf formulae. Given such a category \mathcal{A} and a chosen Birkhoff subcategory \mathcal{B} of \mathcal{A} , thus we describe the Barr-Beck derived functors of the reflector of \mathcal{A} onto \mathcal{B} in terms of centralization of higher extensions. In case \mathcal{A} is the category Gp of all groups and \mathcal{B} is the category Ab of all abelian groups, this yields a new proof for Brown and Ellis's formulae.

REFERENCES

- [1] M. Barr and J. Beck, *Homology and standard constructions*, Seminar on triples and categorical homology theory, Lecture Notes in Mathematics, vol. 80, Springer, 1969, pp. 245–335.
- [2] F. Borceux and G. Janelidze, *Galois theories*, Cambridge Studies in Advanced Mathematics, vol. 72, Cambridge University Press, 2001.
- [3] R. Brown and G. J. Ellis, *Hopf formulae for the higher homology of a group*, Bull. London Math. Soc. 20 (1988), 124–128.
- [4] G. Donadze, N. Inassaridze, and T. Porter, *n-Fold Čech derived functors and generalised Hopf type formulas*, K-Theory 35 (2005), no. 3–4, 341–373.
- [5] T. Everaert, M. Gran, and T. Van der Linden, *Higher Hopf formulae for homology via Galois Theory*, [math.AT/0701815](#), January 2007.
- [6] T. Everaert and T. Van der Linden, *Baer invariants in semi-abelian categories II: Homology*, Theory Appl. Categ. 12 (2004), no. 4, 195–224.
- [7] G. Janelidze, *What is a double central extension? (The question was asked by Ronald Brown)*, Cah. Top. Géom. Diff. Catég. XXXII (1991), no. 3, 191–201.
- [8] G. Janelidze, *Higher dimensional central extensions: A categorical approach to homology theory of groups*, Lecture at the International Category Theory Meeting CT95, Halifax, 1995.
- [9] G. Janelidze, L. Márki, and W. Tholen, *Semi-abelian categories*, J. Pure Appl. Algebra 168 (2002), 367–386.

*Joint work with Tomas Everaert and Marino Gran. Research partly financed by the Eduard Čech Center for Algebra and Geometry, Brno, Czech Republic.

Weak complicial sets and internal quasi-categories

Dominic Verity

It is well known that we may represent (strict) ω -categories as simplicial sets, via Street's ω -categorical nerve construction [2]. What may be less well known, is that we may extend Street's nerve functor to one which has been shown to be fully-faithful (Verity [3]). This is achieved by augmenting each simplicial set in the codomain of this functor with a specified subset of *thin simplices* and restricting our attention to those simplicial maps that preserve the property of thinness, to obtain the category of *stratified* simplicial sets. Under this representation, we gain an equivalence between the category of (strict) ω -categories and a certain category of *complicial sets* which was originally defined and studied by Roberts [1].

Complicial sets are characterised, amongst the stratified simplicial sets, by a *strict horn filler* condition under which those simplicial horns that satisfy a certain admissibility criterion have *unique* fillers by thin simplices. By weakening this, to only insist on existence (rather than unique existence) of thin fillers for such horns, we obtain a larger class of *weak complicial sets*. Aside from extending the class of complicial sets, and thus the category of (strict) ω -categories, this class includes all Kan complexes and all quasi-categories (weak Kan complexes). Furthermore, weak complicial sets may be shown to be the fibrant objects for a Quillen model structure on the category of stratified simplicial sets, which extends both the canonical model structure on simplicial sets and Joyal's quasi-categorical model structure (see [4]).

Finally, it is also known that we may canonically enrich the category of weak complicial sets over itself, in a manner that corresponds directly to our enrichment of the category of 2-categories to a Gray-category. By taking the *homotopy coherent nerve* of this structure, we can show that the totality of all weak complicial sets can itself be regarded as a richly structured (large) weak complicial set of homomorphisms and higher strong transformations [5]. Taken as a whole, these results lead us to regard the theory of weak complicial sets as a full weak ω -category theory, modelled upon a simplicial rather than a globular geometry.

While the works cited above provide us with quite a bit of information about the homotopy theory of weak complicial sets, they do little to illuminate their (weak) category theory. To that latter end we follow the path taken by the founders of 2-category theory who observed, early on, that much of the fundamental category theory of their discipline arose directly from the observation that 2-categories could be regarded as categories internal to or enriched over $\underline{\text{Cat}}$, the category of all small categories. It turns out that we may employ an analogous approach to weak complicial

sets, representing them as certain *quasi-categories internal to* the category of weak complicial sets itself.

In this talk, we plan to introduce just enough weak complicial set theory to allow us to describe the construction of this quasi-categorical representation. This will lead naturally to a discussion of the intuitions that underlie the development of the resulting internal quasi-category theory. Finally, we apply this work to formulating a conjectured coherence result for weak complicial sets.

REFERENCES

- [1] J. E. Roberts, Mathematical aspects of local cohomology, in: *Proceedings of the Colloquium on Operator Algebras and their Application to Mathematical Physics*, Marseille, 1977.
- [2] R. H. Street, The algebra of oriented simplexes, *Journal of Pure and Applied Algebra*, 49:283–335, 1987.
- [3] D. Verity, *Complicial Sets: Characterising the Simplicial Nerves of Strict ω -Categories*, Memoirs. American Mathematical Society, to appear.
- [4] D. Verity, Weak complicial sets I, basic homotopy theory, *Advances in Mathematics*, to appear,
- [5] D. Verity. Weak complicial sets II, nerves of complicial gray-categories, in: A. Davydov (editor) *Categories in Algebra, Geometry and Mathematical Physics (Streetfest)*, Contemporary Mathematics, to appear.

Calculating limits and colimits compositionally

R.F.C. Walters *

The present authors have considered (bi-)categories of spans and cospans of graphs in the study of algebras of processes, cospan operations providing the sequential operations, and span operations corresponding parallel operations (see [1], [2], [3], [4], [5]). This paper is an attempt to answer the question as to why these categories of spans and cospans of graphs arise in applications, with the aim of understanding better the abstract context of this research. First notice that, in our work, graphs appear in different roles, and it is important to distinguish these roles. To this end we consider a category **E** whose objects are “state spaces of systems” – the category of graphs is just one example used very frequently in computer science. We are interested in describing systems constructed from parts. There are two ways of describing how a system is made of parts, one a geometrical description of the configuration of the parts, the other an algebraic description in which the total system is given by an expression in the parts, in an algebra of systems. The geometric point of view commonly consists in describing the system as a (co)limit of subsystems, the (co)limit parametrized by some form of graph (this is the second use of graphs). Our answer to the question raised above is based on [6]: the algebras which allow a system to be described compositionally are the well-supported compact closed (wscc) categories of spans and cospans in **E** ([7]). This result may be regarded as an abstract Kleene theorem.

REFERENCES

- [1] P. Katis, N. Sabadini, R.F.C. Walters, *Span(Graph): A categorical algebra of transition systems*, Proc. AMAST '97, SLNCS 1349, Springer Verlag, (1997) 307-321.
- [2] P. Katis, N. Sabadini, R.F.C. Walters, *On the algebra of systems with feedback and boundary*, Rendiconti del Circolo Matematico di Palermo Serie II, Suppl. 63: (2000) 123-156.
- [3] P. Katis, N. Sabadini, R.F.C. Walters, *A formalisation of the IWIM Model*, in: Proc. COORDINATION 2000, (Eds.) A. Porto, G.-C. Roman, LNCS 1906, Springer Verlag, (2000) 267-283.
- [4] P. Katis, N. Sabadini, R.F.C. Walters, *Feedback, trace and fixed-point semantics*, Theor. Informatics Appl. 36, (2002) 181-194.

*Joint work with R. Rosebrugh and N. Sabadini.

- [5] R. Rosebrugh, N. Sabadini, R.F.C. Walters, *Minimization and minimal realization in $\text{Span}(\text{Graph})$* , MSCS, Volume 14, (2004) 685-714.
- [6] R. Rosebrugh, N. Sabadini, R.F.C. Walters, *Generic commutative separable algebras and cospans of graphs*, Theory and Applications of Categories, Vol. 15, No. 6, (2004) 164-177.
- [7] R.F.C. Walters, *The tensor product of matrices*, Lecture, International Conference on Category Theory, Louvain-la-Neuve, (1987).

Bicategories and the philosophy of language

Graham White

This paper analyses the arguments of a group of philosophers – Davidson [7, 6], Kim [9], Bennett [1] and Parsons [15] – who have worked on the semantics of actions. They argue that inference patterns involving adverbs require a semantics, formalised in first-order logic, in which actions are first-class individuals. In particular, equalities between actions must be meaningful. (We should note, in passing, that this work is semantically focused, and thus it addresses concerns which are rather orthogonal to the syntactic investigations of, for example, Lambek and his co-authors [4, 3, 11, 12, 10].)

The inferences involving adverbs are those of the form

$$\text{Susan ran quickly} \vdash \text{Susan ran}, \tag{1}$$

and we argue that a much less problematic, and more direct, formalisation would be bicategorical. Objects are states, 1-cells are actions (i.e. transitions between states), and 2-cells are inferences such as (1): we have, then, a locally posetal bicategory. In order to handle assertions, we use a 2-fibration [8] in Boolean algebras over this bicategory: objects of the total category are assertions about a state, 1-cells of the total category are sets of physical processes implementing the action, and 2-cells (liftings of inferences such as (1)) are containments between the corresponding sets of physical processes. We also need a 2-comprehension (i.e. a right adjoint to the \top functor from base to total category) in order to construct actions out of sets of physical processes, and we need equality predicates (i.e. right adjoints to diagonals) in order to say when the results of two actions are the same.

We can prove the following. Given the above setup, we can use the results of Carboni and Walters [2] to show that these fibrations arise as subobject fibrations of categories of relations over regular categories. We also give a sequent calculus formulation of inference in the internal language, and prove soundness and completeness.

We can, thus, give a rather direct formalisation of the inferences in question, and thus move the philosophical debate significantly forward. Equalities do turn out to be significant, but they are the equalities which, in the Carboni-Walters construction, come from the right adjoints to diagonals: in philosophical terms, they are equalities between possible worlds, rather than equalities between actions. We have, then, using a methodology very like Lawvere's [13, 14] to analyse the logical infrastructure necessary for inferences such as (1).

REFERENCES

- [1] Jonathan Bennett. *Events and Their Names*, Hackett, 1988.

- [2] A. Carboni and R.F.C. Walters, *Cartesian bicategories I* Journal of Pure and Applied Algebra 49 (1987) 11–32.
- [3] Claudia Casadio and Joachim Lambek, *Codic pronouns in Italian*. In P. de Groote, G. Morrill, and C. Retoré (editors) *LACL 2001*, number 2099 in LNAI, pages 110–124, Berlin, Heidelberg 2001, Springer-Verlag.
- [4] Claudio Casadio and Joachim Lambek, *A tale of four grammars*, Studia Logica 71 (2002) 315–329.
- [5] Donald Davidson, *Essays on Actions and Events*, Oxford University Press 1980.
- [6] Donald Davidson, *The individuation of events*, in: *Essays on Actions and Events* [5], pages 163–180. Originally published in [16, 216–34].
- [7] Donald Davidson, *The logical form of action sentences*, in: *Essays on Actions and Events* [5], pages 105–148. Originally published in N. Rescher (ed.), *The Logic of Decision and Action*, University of Pittsburgh Press, 1967.
- [8] C. Hermida, *Some properties of fib as a fibred 2-category*, Journal of Pure and Applied Algebra 134 (1999) 83–109.
- [9] Jaegwon Kim, *Supervenience and Mind: Selected Philosophical Essays*, Cambridge Studies in Philosophy, Cambridge University Press, Cambridge, 1993.
- [10] Joachim Lambek, *Programs, grammars and arguments: A personal view of some connections between computation, language, and logic*, Bulletin of Symbolic Logic 3 (1997) 312–328.
- [11] Joachim Lambek, *Type grammar revisited*. In A. Lecomte, F. Lamarche, and G. Perrier (editors) *LACL 1997*, number 1582 in LNAI, pages 1–27, Berlin, Heidelberg 1999, Springer-Verlag.
- [12] Joachim Lambek, *From word to sentence: A pregroup analysis of the object pronoun who(m)*, Journal of Logic, Language and Information 16 (2007) 303–323.
- [13] F.W. Lawvere, *Adjointness in foundations*, Dialectica 23 (1969) 281–296.
- [14] F.W. Lawvere, *Equality in hyperdoctrines and the comprehension scheme as an adjoint functor*, in: A. Heller (editor) *Applications of Categorical Algebra*, pages 1–14. AMS, Providence RI 1970.
- [15] Terence Parsons, *Events in the Semantics of English: A Study in Subatomic Semantics*, volume 19 of *Current Studies in Linguistics*, MIT Press 1990.
- [16] Nicholas Rescher, *Essays in Honor of Carl G. Hempel*, Reidel, Dordrecht 1969.

Cartesian bicategories as symmetric monoidal bicategories

R.J. Wood *

Cartesian *locally ordered* bicategories were defined and studied in [1]. It was clear from the outset that the restriction to locally ordered bicategories could, and should, be removed but in 1987 that seemed to be a technically forbidding task. Subsequent work on equipments, first in the locally ordered case of [2] and culminating in the *cartesian equipments* of [3], suggested that the technicalities could be handled by careful exploitation of the universal properties that define cartesian bicategories. At CT06, reporting on [4], we presented our definition and used it to characterize cartesian bicategories of the form $\text{Span}(\mathcal{E})$ and $\text{Rel}(\mathcal{E})$, for suitable categories \mathcal{E} . Here we show that the canonical tensor product of a cartesian bicategory endows it with the structure of a symmetric monoidal bicategory.

REFERENCES

- [1] A. Carboni and R.F.C. Walters, *Cartesian Bicategories I*, J. Pure Appl. Algebra 49 (1987) 11–32.
- [2] A. Carboni, G.M. Kelly, and R.J. Wood, *A 2-categorical approach to change of base and geometric morphisms I*, Cahiers Top. et Géom Diff. XXXII-1 (1991) 47–95.
- [3] A. Carboni, G.M. Kelly, D. Verity, and R.J. Wood, *A 2-Categorical Approach to Change of Base and Geometric Morphisms II*, TAC 4:5 (1998) 73–136.
- [4] A. Carboni, G.M. Kelly, R.F.C. Walters, and R.J. Wood, *Cartesian Bicategories II*, to appear.

*Joint work with A. Carboni, G.M. Kelly and R.F.C Walters.

Kan extensions, models of sketches and the coequalizer of the kernel pair process

João J. Xarez

Let $K : \mathbf{B} \rightarrow \mathbf{A}$ be a functor such that the image $K(\mathbf{B}_0)$ of the objects in \mathbf{B} is a cogenerating set of objects for \mathbf{A} . It will be shown that the coequalizer of the kernel pair process applied to the adjunction $\mathbf{Set}^K \dashv \text{Ran}_K : \mathbf{Set}^{\mathbf{B}} \rightarrow \mathbf{Set}^{\mathbf{A}}$ produces a reflection $\mathbf{Set}^{\mathbf{A}} \rightarrow \text{Mono}(\mathbf{Set}^K)$ with stable units and monotone-light factorization. Such is the case if \mathbf{A} is the opposite category of the category of positive ordinals $[n]$, and \mathbf{B} is its full subcategory with just the object $[0]$, giving rise to the (non-trivial) monotone-light factorization for simplicial sets via ordered simplicial complexes.

The result above follows from the more general fact that an adjunction $\mathbf{C} \rightarrow \mathbf{X}$ equipped with a (pre)factorization system on \mathbf{C} and satisfying some conditions, produces a reflection with stable units and monotone-light factorization. E.g., the (non-trivial) monotone-light factorization for categories via preorders.

The latter factorization is just the restriction of the former. Why this is so will be studied replacing \mathbf{A} and \mathbf{B} by sketches, i.e., categories with distinguished families of cones.

REFERENCES

- [1] Carboni, A., Janelidze, G., Kelly, G. M., Paré, R., *On localization and stabilization for factorization systems.*, App. Cat. Struct. 5 (1997) 1–58.
- [2] Xarez, J. J. *A Galois theory with stable units for simplicial sets*, Theory Appl. Categories 15 (2006) 178–193.

Combinatorial description of many-to-one computads

Marek Zawadowski

The definition of multitopic categories the weak ω -categories in the sense of Makkai contains two ingredients. The first constitutes a description of shapes of cells that are considered (this includes the relation between cells and their domains and codomains) and the second constitutes a mechanism of composition. This work is meant to be a contribution to a better understanding of the first ingredient of the M. Makkai's definition of multitopic category.

In my talk I want to present a combinatorial description of many-to-one computads (see [3]). The main device to do this is the notion of an *ordered face structure*. Ordered face structures are to many-to-one computads are like simple ω -graphs, c.f. [1], to the ono-to-one computads i.e. the free ω -categories over ω -graphs. In other words ordered face structures represent the shapes of arbitrary cells in such many-to-one computads. They are axiomatized using operations of domain δ , codomain γ , and a strict order $<^\sim$. The most important axiom is the axiom of *globularity*

$$\gamma\gamma(a) = \gamma\delta(a) - \delta\dot{\delta}^{-\lambda}(a), \quad \delta\gamma(a) \equiv_1 \delta\delta(a) - \gamma\dot{\delta}^{-\lambda}(a),$$

which is the many-to-one version of the usual globularity condition $cc = cd$ and $dc = dd$. The category of ordered face structures is a monoidal globular category in the sense of M. Batanin, with tensor given by local pushouts with a suitable extended order relation $<^\sim$. This observation plays the central role in the construction of many-to-one computads.

REFERENCES

- [1] M. Makkai and M. Zawadowski, *Diski and duality*. TAC 8(7), 2001, 114–243.
- [2] M. Zawadowski, *On positive face structures and positive-to-one computads*, preprint, 2006, <http://www.mimuw.edu.pl/zawado/PCI.pdf>.
- [3] M. Zawadowski, *On ordered face structures and many-to-one computads*, preprint, 2007, www.mimuw.edu.pl/zawado/MCI.pdf.