A formal language for cyclic operads

Pierre-Louis Curien, Jovana Obradovic

A formal language for cyclic operads

Pierre-Louis Curien[1] and Jovana Obradović[2]
πr^2 team, IRIF, CNRS, Université Paris Diderot, and Inria, France
[1]curien@pps.univ-paris-diderot.fr [2]jovana@pps.univ-paris-diderot.fr

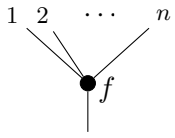
Abstract

We give a complete proof of the equivalence between the unbiased and biased definitions of cyclic operads, through a λ-calculus-style formal language, called the μ-syntax, and a new formalism of trees that provides a crisp description of the monad of unrooted trees (whose nodes are decorated with operadic operations).

Introduction

An operad is a collection of abstract operations of different arities, equipped with a notion of how to compose them and an action of permuting their inputs. An n-ary operation f should be thought of as a single-node rooted tree, whose node is decorated with the symbol f and that has n inputs, labeled either by natural numbers from 1 to n (in which case the operad is characterised as *skeletal*), or, equivalently, by elements of an arbitrary finite set of cardinality n (in which case the operad is *non-skeletal*).

Formally, in the skeletal approach, the set $\mathcal{O}(n)$ of n-ary operations is determined by a functor $\mathcal{O} : \mathbf{\Sigma}^{op} \to \mathbf{Set}$, where $\mathbf{\Sigma}$ is the skeleton of the category **Bij** of finite sets and bijections, formed by the sets $n = \{1, 2, \ldots, n\}$, $n \in \mathbb{N}$, and **Set** is the category of sets and functions. Then, for any permutation σ of n, the induced map $\mathcal{O}(\sigma)$ determines a permutation of inputs of an operation



and this constitutes the action of the symmetric group $\mathbb{S}_n$ on $\mathcal{O}(n)$.

As for the formal description of operadic composition, the *unbiased* and *biased* frameworks provide two ways to complete the characterisation of an operad.

In the unbiased framework, an operad is defined as an *algebra over a monad of rooted trees*. These trees act as pasting schemes, and the operations decorating their nodes are "composed in one shot" through the structure morphism of the algebra.

In the biased approach, the definition of an operad is biased towards "local" operadic compositions, in the sense that these are the only explicitely defined concepts. The various ways to derive a global operadic composition are then equated by the appropriate associativiy axioms. In the original definition of an operad, given by May in [M72], the operadic composition structure is specified by morphisms

$$\gamma_{n,k_1,\ldots,k_n} : \mathcal{O}(n) \otimes \mathcal{O}(k_1) \otimes \cdots \otimes \mathcal{O}(k_n) \to \mathcal{O}(k_1 + \cdots + k_n)$$

and the unit $id \in \mathcal{O}(1)$, defined for all $n \geq 1$ and $k_i \geq 0$, which are subject to associativity, equivariance and unit axioms. This kind of composition is refered to as *simultaneous*, in reference to the underlying simultaneous grafting operation on rooted trees. The presence of operadic units allows for an equivalent biased approach for introducing operadic composition. Instead of working with simultaneous composition, one can introduce it by formulas

$$f \circ_i g = \gamma_{m,1,\ldots,1,n,1,\ldots,1}(f, id, \ldots, id, g, id, \ldots, id)$$

where g appears as the i-th argument of f, which specify individual compositions

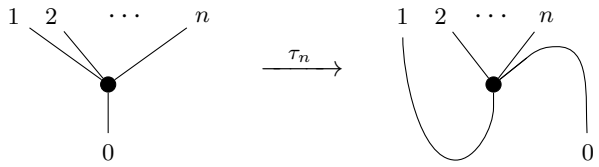
$$\circ_i : \mathcal{O}(m) \otimes \mathcal{O}(n) \to \mathcal{O}(m + n - 1)$$

for all $1 \leq i \leq m$. This definition of operadic composition, which was first formalised by Markl in [M96], is called *partial*.

Precise definitions of skeletal operads can be seen in [M06] (Definition 1, Definition 11, and Theorem 40, for the simultaneous, partial and unbiased operadic compositions, respectively). The non-skeletal version of the operad structure is obtained by passing from $\mathbf{\Sigma}$ to $\mathbf{Bij}$, i.e. by building operadic composition over a functor $\mathcal{O} : \mathbf{Bij}^{op} \to \mathbf{Set}$. Such a definition is found in [L10, Definition 1.3.2], and [L10, Theorem 1.2.7] is a theorem that sets up the equivalence between the skeletal and non skeletal definitions.

Operads encode categories of algebras whose operations have multiple inputs and one output, such as associative algebras, commutative algebras, Lie algebras, etc. The interest in encoding more general algebraic structures was a part of the *renaissance of operads* in the early nineties of the last century, when various generalizations of operads came into existence. The formalism of cyclic operads was originally introduced by Getzler and Kapranov in [GK95]. The motivation came from the framework of cyclic homology: in their paper, Getzler and Kapranov show that, in order to define a cyclic homology for $\mathcal{O}$-algebras, $\mathcal{O}$ has to be what they call a cyclic operad. Roughly speaking, an enrichment of the (ordinary) operad structure is provided by adding to the action of permuting the inputs of an operation an action of interchanging its output with one of the inputs. This feature essentially makes the distinction between the inputs and the output no longer visible, which is adequately captured by unrooted trees as pasting schemes.

The notion of a cyclic operad was originally given in the unbiased manner in [GK95, Definition 2.1], over the structure of a monad in a category of unrooted trees. Like operads, biased cyclic operads can be defined by means of simultaneous compositions [GK95, Theorem 2.2] or of partial composition [M06, Proposition 42]. In both of these definitions, the action of $\mathbb{S}_n$ is extended with the cycle $\tau_n = (0, 1, \ldots, n)$, whose action includes making the output of an operation (denoted now with 0) to be the first input and the input indexed with n to be the output, in a way that is compatible with operadic composition and preserves units. The action of τ_n can be visualised as the clockwise rotation of all wires of a tree, such that each wire takes the position of its right-hand side neighboring wire:



The fact that we can now compose two operations by grafting them along wires that "used to be outputs" leads to another point of view on cyclic operads, in which they are seen as generalisations of operads for which an operation, instead of having inputs and an (exchangeable) output, now has "entries", and it can be composed with another operation along any of them. One can find such an *entries-only* definition in [M14, Definition 48]. By contrast, we shall refer to the definitions based on describing cyclic operads as operads with extra structure as *exchangeable-output* ones.

The equivalence between the unbiased and biased definitions of a cyclic operad is formally given as a categorical equivalence that is, up to some extent, taken for granted in the literature. The issue that the construction of the structure morphism of an algebra over the monad out of the data of a biased cyclic operad should be shown independent of the way trees are decomposed has not been addressed in the proof of [GK95, Theorem 2.2], while the proof of [M06, Proposition 42] is not given. Also, the monad structure is usually not spelled out in detail, in particular for what regards the correct treatment of the identities. The primary goal of this paper is to formalise rigorously the equivalence between the unbiased and biased definitions of cyclic operads. Instead of comparing one of the two exchangeable-output biased definitions with the unbiased one, as done in [GK95, Theorem 2.2] and [M06, Proposition 42], we show that the entries-only and the unbiased definition describe the same structure. Another particularity in our approach is that the appropriate categorical equivalence will be proved in a syntactical environment: a cyclic operad with biased composition will be expressed as a model of the equational theory determined by the axioms of the entries-only definition, while the monad of unrooted trees figuring in the unbiased approach will be expressed through a formal language called *μ-syntax*, that we now introduce briefly.

The name and the language of the μ-syntax formalism are motivated by another formal syntactical tool, the $\mu\tilde{\mu}$-subsystem of the $\overline{\lambda}\mu\tilde{\mu}$-calculus, presented by Curien and Herbelin in [HC00]. In their paper, programs are described by means of expressions called commands, of the form

$$\langle \mu\beta.c_1 \,|\, \tilde{\mu}x.c_2 \rangle,$$

which exhibit a computation as the result of an interaction between a term $\mu\beta.c_1$ and an evaluation context $\tilde{\mu}x.c_2$, together with a symmetric reduction system

$$c_2[\mu\beta.c_1/x] \longleftarrow \langle \mu\beta.c_1 \,|\, \tilde{\mu}x.c_2 \rangle \longrightarrow c_1[\tilde{\mu}x.c_2/\beta],$$

reflecting the duality between call-by-name and call-by-value evaluation. In our syntactical approach, we follow this idea and view operadic composition as such a program, i.e. as an interaction between two operations f and g, where f provides an input β (selected with μ) for the output x of g (marked with $\tilde{\mu}$). By moving this concept to the entries-only setting of cyclic operads, the input/output distinction of the $\mu\tilde{\mu}$-subsystem goes away, leading to the existence of a *single binding operator* μ, whose purpose is to select the entries of two operations which are to be connected in this interaction.

The action of putting in line the characterization of the monad of unrooted trees, built upon the formalism of *unrooted trees with half-edges* commonly used in the operadic literature, together with the characterization by means of μ-syntax, i.e. of proving their equivalence, makes the greatest part of the paper. It involves setting up an intermediate formalism of unrooted trees, called the formalism of *Vernon trees*, that provides concise and lightweight pasting shemes for cyclic operads, and whose syntactical flavour reflects closely the language of μ-syntax. Roughly speaking, the formal characterisation of a Vernon tree captures precisely the information relevant for describing

the corresponding monad, which eases the verifications of the appropriate laws.

Although μ-syntax was originally designed precisely to help us carry out the proof of the equivalence between the unbiased and biased definitions of cyclic operads, it certainly has a value at the very level of encoding the (somewhat cumbersome) laws of the partial composition operation for cyclic operads. In other words, we propose it as an alternative representation of the biased structure of a cyclic operad.

Outline of the paper. Section 1 is a review of the existing definitions of cyclic operads. It finishes with the statement of the theorem that expresses the equivalence between the unbiased and the biased definition. In Section 2, we introduce the formalism of Vernon trees and we describe within it the monad of unrooted trees in full detail. Section 3 will be devoted to the introduction and analysis of the μ-syntax. We also show how to interpret the μ-syntax in an arbitrary cyclic operad. In Section 4, we deliver the proof of the main theorem. We summarise the respective merits of Vernon trees and μ-syntax in this proof in the conclusion.

Notation and conventions. This paper is about non-skeletal cyclic operads introduced in **Set**. This is just a matter of convenience: we prefer the non-skeletal setting because we prefer formulas with "named" (rather than "numbered") variables, and we chose to work in **Set** (rather than in an arbitrary symmetric monoidal category) only to be able to (correctly) speak about operadic operations in terms of elements. We assume the existence of operadic units. The proofs that "could be left to the reader" are given in the Appendix, and the corresponding lemmas are labeled with an asterix.

For a bijection $\sigma : X' \to X$ and $Y \subseteq X$, we denote with $\sigma|^Y$ the restriction of σ on $\sigma^{-1}(Y)$. For $y \notin X \cup X'$, we denote with σ_y the bijection $\sigma_y : X' \cup \{y\} \to X \cup \{y\}$, defined as σ on X', and such that $\sigma_y(y) = y$. If $\sigma(x') = x$, we denote with $\sigma^{y/x'}$ the bijection defined in the same way as σ, except that, instead of x', it contains y in its domain (the inverse image of x now being y). Finally, if $\tau : Y' \to Y$ is a bijection such that $X' \cap Y' = X \cap Y = \emptyset$, then $\sigma + \tau : X' \cup Y' \to X \cup Y$ denotes the bijection defined as σ on X' and as τ on Y'.

1 Cyclic operads

The content of this section is to a great extent a review and a gathering of material coming from [GK95], [M06] and [M14]. The main definitions are Definition 1 and Definition 2, and the main theorem is Theorem 1, which claims the equivalence of these two definitions.

1.1 Biased definition of cyclic operads

We introduce below the entries-only definition of cyclic operads, by following Markl's definition [M14, Definition 48] for a particular case when the underlying functor is $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$, and adapting it further by also demanding operadic units.

In the sequel, for $f \in \mathcal{C}(X)$ and a bijection $\sigma : X' \to X$, we write f^σ instead of $\mathcal{C}(\sigma)(f)$.

Definition 1. A **cyclic operad** is a contravariant functor $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$, together with a distinguished element $id_{x,y} \in \mathcal{C}(\{x, y\})$ for each two-element set $\{x, y\}$, and a partial composition operation

$$ {}_x\circ_y : \mathcal{C}(X) \times \mathcal{C}(Y) \to \mathcal{C}((X\backslash\{x\}) \cup (Y\backslash\{y\})), $$

defined for arbitrary non-empty finite sets X and Y and elements $x \in X$ and $y \in Y$, such that $(X\backslash\{x\}) \cap (Y\backslash\{y\}) = \emptyset$. These data satisfy the associativity, equivariance and unitality axioms given below, wherein, for each of the axioms, we assume the set disjointness that ensures that all the partial compositions involved are well-defined.

Associativity. For $f \in \mathcal{C}(X)$, $g \in \mathcal{C}(Y)$ and $h \in \mathcal{C}(Z)$, the following two equalities hold:

(A1) $(f \,{}_x\circ_y\, g) \,{}_u\circ_z\, h = f \,{}_x\circ_y\, (g \,{}_u\circ_z\, h)$, where $x \in X$, $y, u \in Y$, $z \in Z$, and

(A2) $(f \,{}_x\circ_y\, g) \,{}_u\circ_z\, h = (f_u\circ_z\, h) \,{}_x\circ_y g$, where $x, u \in X$, $y \in Y$, $z \in Z$.

Equivariance. For bijections $\sigma_1 : X' \to X$ and $\sigma_2 : Y' \to Y$, and $f \in \mathcal{C}(X)$ and $g \in \mathcal{C}(Y)$, the following equality holds:

(EQ) $f^{\sigma_1} \,{}_{\sigma_1^{-1}(x)}\circ_{\sigma_2^{-1}(y)}\, g^{\sigma_2} = (f_x\circ_y\, g)^{\sigma}$,

where $\sigma = \sigma_1|^{X\backslash\{x\}} \cup \sigma_2|^{Y\backslash\{y\}}$.

Unitality. For $f \in \mathcal{C}(X)$ and $x \in X$, the following two equalities hold:

(U1) $f \,{}_x\circ_y\, id_{y,z} = f^{\sigma}$, and

(U2) $id_{y,z} \,{}_y\circ_x\, f = f^{\sigma}$,

where $\sigma = id_{X\backslash\{x\}}$ on $X\backslash\{x\}$ and $\sigma(z) = x$. Moreover, the unit elements are preserved under the action of $\mathcal{C}(\sigma)$, i.e.

(U3) $id_{x,y}{}^{\sigma} = id_{u,v}$,

for any two two-element sets $\{x, y\}$ and $\{u, v\}$, and a bijection $\sigma : \{u, v\} \to \{x, y\}$.

Note that we impose a slightly weaker condition on the sets X and Y and elements $x \in X$ and $y \in Y$ involved in partial composition than in [M14, Definition 48]: instead of requiring X and Y to be disjoint, as Markl does, we allow the possibility that they intersect, provided that their intersection is a subset of $\{x, y\}$. This also means that we allow the possibility that $x = y$. Despite this difference, the characterizations of Definition 1 and [M14, Definition 47] are equivalent. More precisely, and under the assumption that the unit elements are already integrated in [M14, Definition 47], all partial compositions allowed in [M14, Definition 47] are obviously covered by the Definition 1. As for the other direction, if $f \,{}_x\circ_y\, g$ is such that, say, $x \in X \cap (Y\backslash\{y\})$, then we can define $f \,{}_x\circ_y\, g$ as $f^{\sigma} \,{}_{x'}\circ_y\, g$, where x' is chosen outside of Y, and $\sigma : (X\backslash\{x\}) \cup \{x'\} \to X$ is identity everywhere except on x', which is sent to x, obtaining in this way a valid definition in the sense of [M14, Definition 47].

The following remark is imported from [L10, Proposition 1.3.4].

Remark 1. *The presence of unit elements $id_{x,y} \in \mathcal{C}(\{x, y\})$ makes the partial composition operation commutative in the sense that, for $f \in \mathcal{C}(X)$, $g \in \mathcal{C}(Y)$, $x \in X$ and $y \in Y$, the following equality holds:*

(CO) $f \,{}_x\circ_y\, g = g \,{}_y\circ_x\, f$.

The above definition naturally induces the notion of simultaneous composition, as a sequence of partial compositions of the form as in the axiom (A2), that is, in which the entry involved in the next instance of a composition always comes from $f \in \mathcal{C}(X)$ and which, moreover, ends when *all* the entries of $f \in \mathcal{C}(X)$ are exhausted. In order to avoid writing explicitly such sequences, we

introduce the following notation. For $f \in \mathcal{C}(X)$, let

$$\varphi : x \mapsto (g_x, \underline{x})$$

be an assignment that associates to each $x \in X$ an operation $g_x \in \mathcal{C}(Y_x)$ and an element $\underline{x} \in Y_x$, in such a way that

$$\bigcap_{x \in X} Y_x \backslash \{\underline{x}\} = \emptyset.$$

Let, moreover, $\sigma : X' \to X$ be an arbitrary bijection such that for all $x \in X$,

$$(X' \backslash \{\sigma^{-1}(x)\}) \cap (Y_x \backslash \{\underline{x}\}) = \emptyset.$$

Under these assumptions, the composite assignment

$$\varphi \circ \sigma : x' \mapsto (g_{\sigma(x')}, \underline{\sigma(x')}),$$

defined for all $x' \in X'$, together with $f^\sigma \in \mathcal{C}(X')$, determines the composition

$$((f^\sigma \, {}_{x'}\circ_{\underline{\sigma(x')}} g_x) \, {}_{y'}\circ_{\underline{\sigma(y')}} g_y) \, {}_{z'}\circ_{\underline{\sigma(z')}} g_z \cdots,$$

consisting of a sequence of partial compositions indexed by the entries of f^σ. We will use the abbreviation $f^\sigma(\varphi \circ \sigma)$ to denote such a composition. Thanks to the axiom (A2), this abbreviation is well-defined, i.e. $f^\sigma(\varphi \circ \sigma)$ does not depend on the order in which the partial compositions were carried out. We finally set

$$f(\varphi) = f^\sigma(\varphi \circ \sigma),$$

and refer to $f(\varphi)$ as *the total composition determined by f and φ*. That $f(\varphi)$ does not depend on the choice of σ is a consequence of the axiom (EQ).

Notice that without the renaming role of σ, $f(\varphi)$ is not necessarily well-defined. For example, $f(\varphi) = (f \, {}_x\circ_{\underline{x}} g_x) \, {}_y\circ_{\underline{y}} g_y$, where $f \in \mathcal{C}(\{x, y\})$, $g_x \in \mathcal{C}(\{\overline{x}, y\})$ and $g_y \in \mathcal{C}(\{\overline{y}, v\})$, is not well-defined, although φ satisfies the required disjointness condition.

In relation to the above construction, the statements of the following lemma are easy consequences of the axioms from Definition 1.

Lemma 1. *The total composition $f(\varphi)$ has the following properties.*

a) Let $\psi : Z \to \bigcup_{x \in X}(Y_x \backslash \{\underline{x}\})$ be a bijection such that for all $x \in X$, $\underline{x} \notin \psi^{-1}(Y_x \backslash \{\underline{x}\})$. Denote with $\psi_{\underline{x}}$ the extension on Y_x of the bijection $\psi|^{Y_x \backslash \{\underline{x}\}}$, which is identity on $\underline{x}$, and let φ_ψ be defined as $\varphi_\psi : x \mapsto (g_x^{\psi_{\underline{x}}}, \underline{x})$, for all $x \in X$. Then $f(\varphi)^\psi = f(\varphi_\psi)$.

b) Let $\psi : y \mapsto (h_y, \underline{y})$ be an assignment that associates to each $y \in \bigcup_{x \in X}(Y_x \backslash \{\underline{x}\})$ an operation $h_y \in \mathcal{C}(Z_y)$ and $\underline{y} \in Z_y$, in such a way that $f(\varphi)(\psi)$ is defined. If φ_ψ is the assigment defined as $\varphi_\psi : x \mapsto (g_x^{\psi_{\underline{x}}}, \underline{x})$, where $\psi_{\underline{x}}$ denotes the extension on Y_x of the assignment $\psi|_{Y_x \backslash \{\underline{x}\}}$, which is identity on $\underline{x}$, then $f(\varphi)(\psi) = f(\varphi_\psi)$.

In what follows, when switching from $f \in \mathcal{C}(X)$ to $f^\sigma \in \mathcal{C}(X')$ under the action of $\sigma : X' \to X$, we will refer to σ as the *renaming* (of the variables) of X to (the appropriate variables of) X', and we will often omit stating explicitly its domain and codomain, always assuming that all the partial compositions defined in the context including $f \in \mathcal{C}(X)$ remain well-defined after the renaming has been done, i.e. in the context with $f^\sigma \in \mathcal{C}(X')$.

The nature of Definition 1 allows us to easily formalise the cyclic operad structure in a syntactic manner. Starting from a functor $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$, we take as formal terms

$$s, t ::= f \mid id_{x,y} \mid s \,{}_x\circ_y\, t$$

typed as

$$\frac{t \in \mathcal{C}(X)}{t : X} \qquad \frac{}{id_{x,y} : \{x, y\}} \qquad \frac{s : X \quad t : Y}{s \,{}_x\circ_y\, t : (X\backslash\{x\}) \cup (Y\backslash\{y\})}$$

where x and y are distinct variables in the second rule, while in the third rule $x \in X$, $y \in Y$ and $(X\backslash\{x\}) \cap (Y\backslash\{y\}) = \emptyset$. For the set of equations we take the one determined by the axioms from the definition. We refer to this syntax as the *combinator syntax* for cyclic operads, and call the terms $t : X$ *combinators*. The set of all combinators induced by $\mathcal{C}$ will be denoted by $\mathtt{cTerm}_{\mathcal{C}}$, and, for a finite set X, $\mathtt{cTerm}_{\mathcal{C}}(X)$ will denote the set of all combinators of type X.

The combinator syntax gives rise to a multi-sorted equational theory: the signature of this theory is determined by taking as sorts all finite sets, while, having denoted with $(s_1, \ldots, s_n; s)$ the sort of an n-ary function symbol, for the set of function symbols we take the collection consisting of

- constants $f : (\,; X)$ (of arity 0),
- identities $id_{x,y} : (\,; \{x, y\})$ (of arity 0),
- actions $\sigma : (X; X')$ (of arity 1), and
- partial compositions ${}_x\circ_y : (X, Y; (X\backslash\{x\}) \cup (Y\backslash\{y\}))$ (of arity 2),

the equations of the theory being derived from the axioms of Definition 1. We can then turn the syntactic representation of a cyclic operad into a semantic one, by reformulating Definition 1 as follows.

Definition 1'. *Given $\mathcal{C}$, a cyclic operad structure over $\mathcal{C}$ is a model of the equational theory above, in which every operation $f \in \mathcal{C}(X)$ is interpreted by itself.*

Based on this definition, $[\_]_{\mathcal{C}} : \mathtt{cTerm}_{\mathcal{C}} \to \mathcal{C}$ will denote the appropriate interpretation function.

1.2 Unbiased definition of cyclic operads

We now revisit the original unbiased definition of cyclic operads, [GK95, Definition 2.1], on slightly adapted grounds: besides switching to a non-skeletal setting, we will reconstruct it within the formalism of trees that incorporates edges as pairs of half-edges (rather then the usual "indivisible" edges), due to Kontsevich and Manin [KM94].

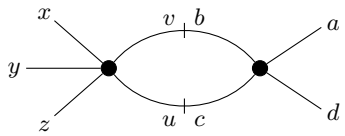
1.2.1 Graphs and unrooted trees

We start by recalling the notion of graph that we will use as the basis for describing unrooted trees, that we took from [M14, Section 4]: a graph Γ is a triple $(\mathit{Flag}(\Gamma), \sigma, \lambda)$, where $\mathit{Flag}(\Gamma)$ is a finite set of *flags* or *half-egdes*, $\sigma : \mathit{Flag}(\Gamma) \to \mathit{Flag}(\Gamma)$ is an involution, and λ is a partition of $\mathit{Flag}(\Gamma)$.

The set of *vertices* $\mathit{Vert}(\Gamma)$ of Γ is determined by λ: vertices are precisely the blocks of the partition λ. We denote by $\mathit{Leg}(v)$ the set of half-edges adjacent to the vertex v, i.e. the flags

belonging to the block v. The set of *edges* of Γ, $edge(\Gamma)$, consists of all pairs of flags forming a two-cycle of σ, with respect to the decomposition of σ into disjoint cycles. The *legs* of Γ are the fixpoints of σ, the set of which will be denoted by $Leg(\Gamma)$. In simple terms, the edges in this formalism are egdes between two vertices in the usual sense, made by connecting two half-edges coming from these two vertices, while the legs are half-edges which are not connected with some other half-edge to form an edge.

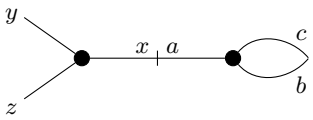
For example, the graph given by the triple $(\{x, y, z, u, v, a, b, c, d\}, \sigma, \{\{x, y, z, u, v\}, \{a, b, c, d\}\})$, where $\sigma = (u\ c)(v\ b)$, should be depicted as



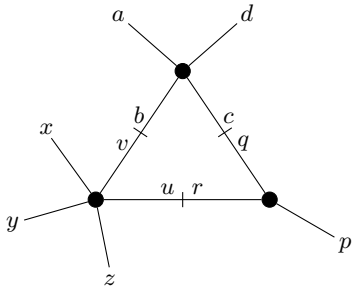
This graph has two vertices, $v_1 = \{x, y, z, u, v\}$ and $v_2 = \{a, b, c, d\}$, two edges, (u, c) and (v, b), and five legs, x, y, z, a, d.

Starting from this notion of graph, we define an *unrooted tree*[1] as a *connected graph without loops, multiple edges and cycles.*

The above graph is not an unrooted tree, since it has two edges between v_1 and v_2. The graph given by the triple $(\{x, y, z, a, b, c\}, \sigma, \{\{x, y, z\}, \{a, b, c\}\})$, where $\sigma = (x\ a)(b\ c)$, is not an unrooted tree either, since the edge (b, c) connects the vertex $\{a, b, c\}$ with itself, i.e. it is a loop:

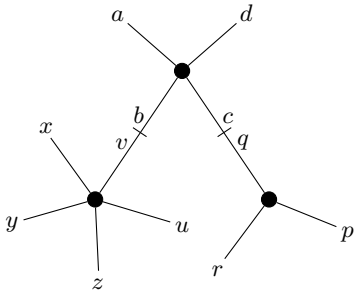


The graph $(\{x, y, z, u, v, a, b, c, d, p, q, r\}, \sigma, \{\{x, y, z, u, v\}, \{a, b, c, d\}, \{p, q, r\}\})$, with $\sigma = (v\ b)(c\ q)$ $(r\ u)$, is another example of a graph which is not an unrooted tree, this time because of the presence of a cycle that connects its three vertices:



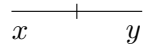
Finally, we get an example of a graph which is an unrooted tree by changing the involution σ of the previous graph to, say, $\sigma' = (v\ b)(c\ q)$, producing in this way the unrooted tree whose graphical representation is

[1] In [M06], Markl uses the attribute *cyclic* when refering to unrooted trees, pointing out in this way the fact that they are pasting schemes for cyclic operads. On the other hand, the same attribute is also used in the literature for characterizing graphs which contain at least one cycle, i.e. which *have* the property *not allowed* by the definition of an unrooted tree. In order to avoid the confusion caused by this ambiguity, the word *cyclic* will not be used in the context of trees in this paper.



If T is an unrooted tree and $l : Leg(T) \to X$ is a bijection between the legs of T and a finite set X, we call a pair (T, l) an *unrooted X-tree* and we refer to the bijection l itself as an *X-labeling* of T. For a finite set X, let $\mathbf{UTree}_X$ denote the category of unrooted X-trees and their isomorphisms, where by an isomorphism between X-trees T_1 and T_2 we mean a bijection $\phi : Flag(T_1) \to Flag(T_2)$ that preserves vertices and commutes with the involutions and X-labelings. For the graphical representation of an unrooted X-tree we take the one of its underlying *unlabeled* tree in which we replaced the names of the legs with the corresponding elements of the set X.

Observe that this definition of a graph implies that each unrooted X-tree has at least one vertex. In what follows, in order to describe the pasting scheme for the identity $id_{x,y}$, we shall need the *exceptional unrooted tree*



without vertices, made only with two half-edges (labeled with x and y in this case), connected into an edge. Whenever X is a two-element set, we assume that among the objects of $\mathbf{UTree}_X$ there is also the exceptional tree determined by X.

The reason to go for exceptional trees, instead of taking explicit vertices for identities, is that, otherwise, in the definition of the free cyclic operad (coming up next), we would have to quotient $\mathbf{UTree}_X$ by more than the vertex-preserving isomorphisms, as the validation of the unit laws would involve the identification of trees that are not isomorphic.

1.2.2 The free cyclic operad

An arbitrary functor $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$ induces a functor $\overline{\mathcal{C}} : {\mathbf{UTree}_X}^{op} \to \mathbf{Set}$, as

$$\overline{\mathcal{C}}(T) := \bigotimes_{v \in Vert(T)} \mathcal{C}(Leg(v)),$$

where T is an unrooted X-tree and $\otimes$ is the unordered tensor product over the set of vertices of T. The functor $F(\mathcal{C}) : \mathbf{Bij}^{op} \to \mathbf{Set}$, underlying the free cyclic operad structure, is then defined as

$$F(\mathcal{C})(X) := \operatorname*{colim}_{T \in \mathbf{UTree_X}} \overline{\mathcal{C}}(T),$$

which, once the definitions of $\overline{\mathcal{C}}$ and of the colimit over it are unrolled, becomes

$$F(\mathcal{C})(X) = \bigoplus_{T \in \mathbf{UTree_X}} \Big(\bigotimes_{v \in Vert(T)} \mathcal{C}(Leg(v)) \Big) / \sim,$$

where $\sim$ is the smallest equivalence relation generated by

$$\overline{\mathcal{C}}(T_2) \ni (f_1, f_2, \ldots, f_n) \sim \overline{\mathcal{C}}(\varphi)((f_1, f_2, \ldots, f_n)) \in \overline{\mathcal{C}}(T_1),$$

for any two X-labeled trees T_1 and T_2, a tree isomorphism $\varphi : T_1 \to T_2$ and $(f_1, f_2, \ldots, f_n) \in \overline{\mathcal{C}}(T_2)$. Therefore, each isomorphism class of $F(\mathcal{C})(X)$ is determined by a $\mathcal{C}$-*decorated unrooted X-tree* T, i.e. by an unrooted X-tree T and an element $(f_1, \ldots, f_n) \in \overline{\mathcal{C}}(T)$, also called a $\mathcal{C}$-*decoration* of T. In particular, for a two-element set, the isomorphism class arising with respect to the corresponding exceptional tree (in which case the tensor product is taken over the empty set) is determined by the unit of the tensor product in **Set**.

As for the rest of the operad structure, the compositions

$$_x\circ_y : F(\mathcal{C})(X) \otimes F(\mathcal{C})(Y) \to F(\mathcal{C})((X\backslash\{x\}) \cup (Y\backslash\{y\}))$$

are induced by grafting of the underlying graphs, and the actions of bijections $\sigma : X' \to X$ by relabeling of their legs. Before composing $[T_1, (f_1, \ldots, f_n)]_\sim \in F(\mathcal{C})(X)$ and $[T_2, (g_1, \ldots, g_m)]_\sim \in F(\mathcal{C})(Y)$, a "precautionary renaming" of the flags of T_1 and T_2 has to be done. We shall detail the relevant renaming issues in 2.4, where we will use Vernon trees to redefine the free cyclic operad in a technically simpler way.

Let **CycOp** denote the category of cyclic operads (and structure-preserving natural transformations). The functor $F : \mathbf{Set}^{\mathbf{Bij}^{op}} \to \mathbf{CycOp}$ described above is left adjoint to the forgetful functor $U : \mathbf{CycOp} \to \mathbf{Set}^{\mathbf{Bij}^{op}}$, which indeed makes $F(\mathcal{C})$ the free cyclic operad built over $\mathcal{C}$. For a cyclic operad $\mathcal{D}$, a hom-set bijection

$$\phi : \mathbf{CycOp}(F(\mathcal{C}), \mathcal{D}) \to \mathbf{Set}^{\mathbf{Bij}^{op}}(\mathcal{C}, U(\mathcal{D}))$$

is determined as follows.

If $\alpha : F(\mathcal{C}) \to \mathcal{D}$ is a morphism of cyclic operads, then, for a finite set X and $f \in \mathcal{C}(X)$, $\phi(\alpha_X) : \mathcal{C}(X) \to U(\mathcal{D})(X)$ is defined as

$$\phi(\alpha_X)(f) = \alpha_X([T, f]_\sim),$$

where T is a single-vertex X-tree, whose legs are given precisely by the set X and whose X-labeling is the identity.

Conversely, if $\beta : \mathcal{C} \to U(\mathcal{D})$ is an element of $\mathbf{Set}^{\mathbf{Bij}^{op}}(\mathcal{C}, U(\mathcal{D}))$ and T is an unrooted X-tree with the labeling l, then the map

$$\bigotimes_{v\in Vert(T)} \mathcal{C}(Leg(v)) \xrightarrow{\otimes_{v\in Vert(T)}\beta} \bigotimes_{v\in Vert(T)} \mathcal{D}(Leg(v)) \xrightarrow{\gamma} \mathcal{D}(Leg(T)) \xrightarrow{\mathcal{D}(l)} \mathcal{D}(X),$$

where γ stands for iterated application of the operadic compositions of $\mathcal{D}$ (coordinated by T), determines the corresponding morphism $\phi^{-1}(\beta)_X : F(\mathcal{C})(X) \to \mathcal{D}(X)$. Notice that this definition has to be shown independent on the order in which iterations behind γ are carried out. We will come back to this issue in Section 4, where we will take profit from the correspondence between the classes of $F(\mathcal{C})(X)$ and the μ-syntax to redefine $\phi^{-1}(\beta)$ in a purely syntactical way and prove this independence.

The definitions of $\phi(\alpha_X)$ and $\phi^{-1}(\beta)_X$ are adapted naturally for the unit element, and, in the other direction, the exceptional tree.

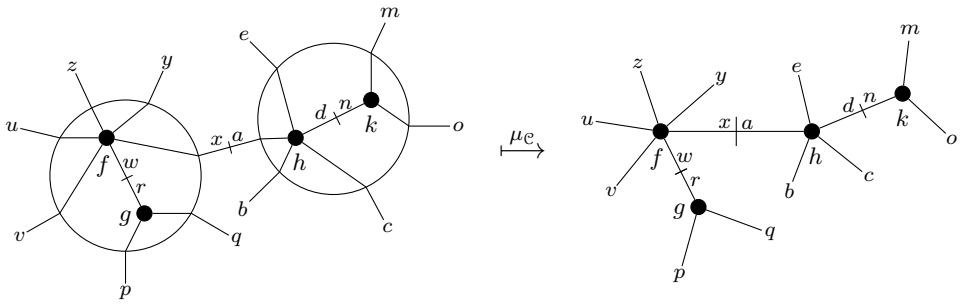
1.2.3 The monad of unrooted trees

The monad of unrooted trees is the monad $\mathbb{M} = (M, \mu, \eta)$ on the category $\mathbf{Set}^{\mathbf{Bij}^{op}}$ arising from the adjunction $F \dashv U$ described above. At this point, we follow the tradition from the literature and give below only its intuitive description.

For a functor $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$, the elements of

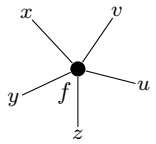
$$MM(\mathcal{C})(X) := \operatorname*{colim}_{T \in \mathbf{UTree}_X} M(\mathcal{C})(T)$$

should be imagined as unrooted X-trees with nodes decorated by the unrooted trees of $M(\mathcal{C})$. With this intuition, one can say that the multiplication $\mu_{\mathcal{C}} : MM(\mathcal{C}) \to M(\mathcal{C})$ "flattens" the trees of $MM(\mathcal{C})$, turning them into "ordinary" $\mathcal{C}$-decorated X-trees:



The monad unit $\eta_{\mathcal{C}} : \mathcal{C} \to M(\mathcal{C})$ associates to each $f \in \mathcal{C}(X)$ the isomorphism class determined by the single-node unrooted X-tree with the node decorated by f:

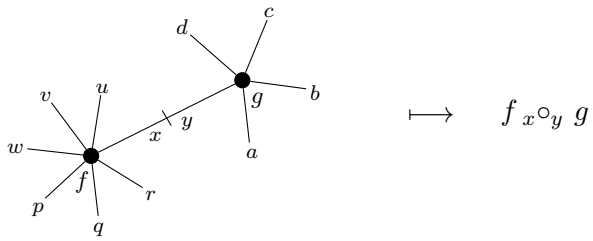
$$\mathcal{C}(X) \ni f \quad \overset{\eta_{\mathcal{C}}}{\longmapsto} \quad \in M(\mathcal{C})(X)$$



Here is the original definition [GK95, Definition 2.1] of a cyclic operad.

Definition 2. A cyclic operad is an algebra over the monad $\mathbb{M} = (M, \mu, \eta)$.

The link between this one-sentence definition and Definition 1 becomes intuitively clear from the mapping given pictorially below, which shows how the structure morphism of such an algebra induces the individual composition operations of a cyclic operad:



The equivalence of these two definitions is formally given by the following theorem.

Theorem 1. *A functor $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$ is a cyclic operad (in the sense of Definition 1) if and only if it is an $\mathbb{M}$-algebra (where $\mathbb{M}$ is the monad from Definition 2), this correspondence establishing the categorical equivalence*

$$\mathbf{CycOp} \cong \mathbf{Alg}_{\mathbb{M}}(\mathbf{Set}^{\mathbf{Bij}^{op}}).$$

The missing ingredient in the proof of this equivalence in the literature is the proof that the laws satisfied by an $\mathbb{M}$-algebra indeed come down to the axioms from Definition 1, and vice-versa. This is what our Section 4 provides.

2 Vernon trees

Recall that, in the definition of the free cyclic operad, each isomorphism class of $F(\mathcal{C})(X)$ is determined by an *unrooted, $\mathcal{C}$-decorated X-tree* T. With respect to the formalism of trees given in 1.2.1, such a tree consists of an unrooted tree T, an X-labeling $l : Leg(T) \to X$ and an element $(f_1, \dots, f_n) \in \overline{\mathcal{C}}(T)$. In this section we introduce an alternative way to describe the isomorphism classes of $F(\mathcal{C})(X)$. The formalism we introduce integrates the leg-labeling and vertex-decorations of unrooted trees a priori, in the very definition of a tree, while leaving out the information unnecessary for describing the corresponding monad. More precisely, as opposed to the approach presented before, which

- involves an explicit X-labeling $l : Leg(T) \to X$ of the legs of a tree, and
- does not carry any information about the names of the vertices of a tree,

in the formalism of Vernon trees

- the labeling of the legs of a tree is handled implicitely, in the sense that the set of the legs of a Vernon tree will be precisely X, and
- the labels of the vertices of a tree are an integral part of the tree structure.

2.1 The syntax

Let $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$ be a functor. We define the *collection of parameters* (of $\mathcal{C}$) as

$$P_{\mathcal{C}} = \{f \mid f \in \mathcal{C}(X) \text{ for some finite set } X\}.$$

An *ordinary corolla* is a term

$$f(x, y, z, \dots),$$

where $f \in \mathcal{C}(X)$ and $X = \{x, y, z, \dots\}$. We call the elements of X the *free variables* of $f(x, y, z, \dots)$, and we shall write $FV(f(x, y, z, \dots)) = X$ (or, shortly, $FV(f) = X$) to denote this set. In addition to ordinary corollas, we define *special corollas* to be terms of the shape

$$(x, y),$$

i.e. terms which do not have a parameter as a head symbol and which consist only of two distinct variables $x, y \in V$. For a special corolla (x, y), we define $FV((x, y)) = \{x, y\}$.

A *Vernon graph* $\mathcal{V}$ is a non-empty, finite set of corollas with mutually disjoint free variables, together with an involution σ on the set

$$V(\mathcal{V}) = \bigcup_{i=1}^{k} FV(f_i) \cup \bigcup_{j=1}^{p} FV((u_j, v_j))$$

of all variables occuring in $\mathcal{V}$. We write

$$\mathcal{V} = \{f_1(x_1, \ldots, x_n), \ldots, f_k(y_1, \ldots y_m), \ldots, (u_1, v_1), \ldots, (u_p, v_p); \sigma\}.$$

We will denote with $Cor(\mathcal{V})$ the set of all corollas of $\mathcal{V}$, and we will refer to an ordinary corolla by its parameter, while we will denote special corollas with $s_1, s_2, \ldots$. As in the case of graphs with half-edges, the set of edges $Edge(\mathcal{V})$ of $\mathcal{V}$ consists of pairs (x, y) of variables such that $\sigma(x) = y$. Finally, $FV(\mathcal{V})$ will denote the set of fixpoints of σ.

The properties needed to make a Vernon graph an unrooted tree are the same as for graphs with half-edges: it has to be connected and it must not contain loops, multiple edges and cycles. However, in moving from graphs to trees, we will now additionally differentiate the classes of trees with respect to the shape of corollas they contain. Let $\mathcal{T}$ be a connected Vernon graph with no loops, multiple edges and cycles.

- If $Cor(\mathcal{T})$ consists only of ordinary corollas, then $\mathcal{T}$ is an *ordinary Vernon tree*.
- If $Cor(\mathcal{T})$ is a singleton with a special corolla, then $\mathcal{T}$ is an *exceptional Vernon tree*.
- A *Vernon tree* is either an ordinary Vernon tree or an exceptional Vernon tree.
- If there are no requirements about the type of corollas in $Cor(\mathcal{T})$, $\mathcal{T}$ will be called an *extended Vernon tree*.

Remark 2. *In general, an arbitrary connected Vernon graph with no loops, multiple edges and cycles is an extended Vernon tree. In particular, every Vernon tree is an extended Vernon tree. On the other hand, every tree containing at least one ordinary and one special corolla (or at least two special corollas) is an extended Vernon tree, but not a Vernon tree.*

We now define α-conversion on extended Vernon trees. Suppose first that

$$\mathcal{T} = \{f(x_1, \ldots, x_n), \ldots; \sigma\}$$

is an ordinary Vernon tree, with $f \in \mathcal{C}(X)$, $x_i \in FV(f) \backslash FV(\mathcal{T})$ and $\sigma(x_i) = y_j$. Let $\tau : X' \to X$ be a bijection that is identity on $X' \backslash \{z\}$, while $\tau(z) = x_i$, where z is fresh with respect to $V(\mathcal{T}) \backslash \{x_i\}$. The α-conversion (for ordinary Vernon trees) is the smallest equivalence relation generated by equalities

$$(f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n), \ldots; \sigma) =_\alpha (f^\tau(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_n), \ldots; \sigma'),$$

where $\sigma' = \sigma$ on $V(\mathcal{T}) \backslash \{x_i, y_j\}$ and $\sigma'(z) = y_j$. This generalises in a natural way to extended Vernon trees: to the set of generators from above we add the clauses

$$\{(x, y), \ldots; \sigma\} =_\alpha \{(x, z), \ldots; \sigma'\},$$

where, for some variable x_i, $\sigma(y) = x_i$ (i.e. y is not a free variable of the tree on the left), z is fresh in the same sense as above, and σ' is the obvious modification of σ. In simple terms, we consider equivalent any two trees such that we can obtain one from another only by renaming variables which are not fixed points of the corresponding involution.

Let us finally denote with $\mathtt{VT}_{\mathcal{C}}(X)$ (resp. $\mathtt{eVT}_{\mathcal{C}}(X)$) the set of all α-equivalence classes of Vernon trees (resp. extended Vernon trees) whose parameters belong to $P_{\mathcal{C}}$ and whose free variables are given by the set X. If X is a two-element set, this definition includes the possibility that a Vernon tree has 0 parameters, in which case the corresponding equivalence class is determined by the appropriate exceptional Vernon tree. We will write $\mathtt{VT}_{\mathcal{C}}$ (resp. $\mathtt{eVT}_{\mathcal{C}}$) for the collection of all Vernon trees (resp. extended Vernon trees) determined by $\mathcal{C}$.

2.2 Vernon trees as pasting schemes for cyclic operads

Due to the presence of the labeling bijections in the formalism of trees with half-edges, the equivalence classes of $F(\mathcal{C})(X)$ are bulkier than the ones of $\mathtt{VT}_{\mathcal{C}}(X)$. However, as we show in the following lemma, $\mathtt{VT}_{\mathcal{C}}(X)$ corresponds precisely to the basis spanning $F(\mathcal{C})(X)$.

Lemma 2. *The isomorphism classes determined by $\mathtt{VT}_{\mathcal{C}}(X)$ are in one-to-one correspondence with the ones of $F(\mathcal{C})(X)$.*

Proof. Since the correspondence between exceptional trees of the two formalisms is obvious, we assume below that the isomorphism classes are determined by ordinary Vernon trees on one hand, and non-exceptional trees with half-edges, on the other.

Let $[\mathcal{T}]_\alpha \in \mathtt{VT}_{\mathcal{C}}(X)$ be the isomorphism class of Vernon trees with representative

$$\mathcal{T} = \{f_1(x_1, \dots, x_n), \dots, f_k(y_1, \dots y_m); \sigma_{\mathcal{T}}\}.$$

The corresponding equivalence class $\overline{\pi}[\mathcal{T}]_\alpha := [\pi_1(\mathcal{T}), \pi_2(\mathcal{T})]_\sim \in F(\mathcal{C})(X)$ is determined by the X-tree $\pi_1(\mathcal{T})$ defined by

$$Flag(\pi_1(\mathcal{T})) = V(\mathcal{T}),\ \ \sigma_{\pi_1(\mathcal{T})} = \sigma_{\mathcal{T}},\ \ \lambda_{\pi_1(\mathcal{T})} = \{FV(f_i) \,|\, 1 \leq i \leq k\},\quad \text{and}\quad l_{\pi_1(\mathcal{T})} = id,$$

and its $\mathcal{C}$-decoration $\pi_2(\mathcal{T}) = (f_1, \dots f_k)$.

Conversely, given an isomorphism class $[T, (f_1, \dots, f_k)]_\sim \in F(\mathcal{C})(X)$, where

$$T = (Flag(T), \sigma_T, \lambda_T = \{v_1, \dots, v_k\}, l_T : Leg(T) \to X),$$

we can first associate to it a "pattern"

$$(\_(x_1, \dots, x_n), \dots, \_(y_1, \dots, y_m); \sigma'),$$

where $\{\{x_1, \dots, x_n\}, \dots, \{y_1, \dots, y_m\}\}$ is determined by partitioning $Flag(T)$ with λ_T followed by renaming each $z \in Leg(T)$ to $l_T(z)$, and σ' is obtained by modifying σ so that it agrees with the in-lined action of l_T. We set $\overline{\nu}[T, (f_1, f_2 \dots, f_k)]_\sim := [\nu(T, (f_1, f_2 \dots, f_k))]_\alpha$, with

$$\nu(T, (f_1, \dots, f_k)) = (f_1^{\tau_1}(x_1, \dots, x_n), \dots, f_k^{\tau_k}(y_1, \dots y_m); \sigma'),$$

obtained by filling in this pattern with $(f_1^{\tau_1}, \dots, f_k^{\tau_k})$, where each τ_i is the renaming of variables from the block v_i of λ_T induced by l_T.

We show that $\overline{\pi}$ and $\overline{\nu}$ are mutually inverse. The equality $\nu(\pi_1(\mathcal{T}), \pi_2(\mathcal{T})) =_\alpha \mathcal{T}$ holds trivially:

the Vernon trees on the left and the right-hand side are equal. On the other hand, verifying that $(\pi_1(\nu(T,(f_1,\ldots,f_k)),\pi_2(\nu(T,(f_1,\ldots,f_k)))) =_\sim (T,(f_1,\ldots,f_k))$ amounts to exhibiting a bijection $\phi : Flag(\pi_1(\nu(T))) \to Flag(T)$ that is an isomorphism between $\pi_1(\nu(T))$ and T. If $l : Leg(T) \to X$ is the labeling of T, then ϕ arises by reversing the action of l, as

$$\phi(z) = \begin{cases} l^{-1}(z) & \text{if } z \in Leg(\pi_1(\nu(T))) \\ z & \text{otherwise}\,. \end{cases}$$

That ϕ satisfies the requirements follows straightforwardly. □

2.3 The monad of Vernon trees

We now detail the monad structure of unrooted trees in the language of Vernon trees.

By the previous lemma, it is clear that the endofunctor $\mathcal{M}$ on $\mathbf{Set}^{\mathbf{Bij}^{op}}$ will be given as

$$\mathcal{M}(\mathcal{C})(X) = \mathtt{VT}_{\mathcal{C}}(X).$$

As for the rest of the structure, it is straightforward to translate in the formalism of Vernon trees the action of the unit of the monad $\mathbb{M}$ from before: to $f \in \mathcal{C}(X)$ it will associate the isomorphism class of the Vernon tree $(\{f(x_1,\ldots,x_n)\}, id)$, where $X = \{x_1,\ldots,x_n\}$. However, the action of the monad multiplication, described in 1.2.3 as "flattening", hasn't been completely determined back then: we have omitted the part that deals with the action on trees that, as some vertices, have the exceptional trees. In order to obtain a complete description, we will translate this action to ordinary Vernon trees and *extend it to extended Vernon trees*. It will turn out that this action is indeed more than just "flattening".

To this end, we build a rewriting system on extended Vernon trees. Apart from α-conversion, we will quotient the trees by instances of the following two equalities:

$$(f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n),(y,z),\ldots;\sigma) = (f^\tau(x_1,\ldots,x_{i-1},z,x_{i+1},\ldots,x_n),\ldots;\sigma')\,,$$

where $\sigma(x_i) = y$, τ is as in 2.1, and σ' is the obvious restriction of σ, and

$$((x,y),(u,v),\ldots;\sigma) = ((x,v),\ldots;\sigma')\,,$$

where $\sigma(y) = u$, and σ' is again the obvious restriction of σ.

Let $\to$ denote (the reflexive and transitive closure of) the union of reductions obtained by orienting these equations from left to right.

Lemma 3*. *The rewriting system* $(\mathtt{eVT}_{\mathcal{C}}, \to)$ *is confluent and terminating.*

As a consequence, an arbitrary normal form $nf(\mathcal{T})$ of an extended Vernon tree $\mathcal{T}$ determines a unique α-equivalence class $[nf(\mathcal{T})]_\alpha$ in $\mathtt{VT}_{\mathcal{C}}$, i.e. there is an assignment $\mathcal{T} \mapsto [nf(\mathcal{T})]_\alpha$, defined for every $\mathcal{T} \in \mathtt{eVT}_{\mathcal{C}}$.

We now formally define the flattening on $\mathcal{M}\mathcal{M}(\mathcal{C})(X)$. Observe that, analogously as before, the isomorphism classes of

$$\mathcal{M}\mathcal{M}(\mathcal{C})(X) = \mathcal{M}(\mathtt{VT}_{\mathcal{C}})(X) = \mathtt{VT}_{\mathtt{VT}_{\mathcal{C}}}(X)$$

are determined by Vernon trees whose parameters are Vernon trees themselves (with parameters from $P_{\mathcal{C}}$), and whose set of free variables is X. Syntactically, a two-level tree of $\mathtt{VT}_{\mathtt{VT}_{\mathcal{C}}}$ can be either

- an exceptional Vernon tree $\{(x,y);id\}$, in which case we trivially have 0 parameters coming from $VT_{\mathcal{C}}$, or
- an ordinary Vernon tree

$$\{\{f(x,y,\dots),g(u,\dots),\dots;\sigma_1\}(x,y,u,\dots),\dots,\{(z,t);id\}(z,t),\dots;\sigma\},$$

whose parameters can be both ordinary and exceptional Vernon trees of $\mathtt{VT}_{\mathcal{C}}$.

Remark 3. *Let $\mathcal{T}$ be a two-level tree in $\mathtt{VT}_{\mathtt{VT}_{\mathcal{C}}}$. Suppose that, for $1 \leq i \leq n$, $\mathcal{T}_i \in \mathtt{VT}_{\mathcal{C}}(Y_i)$ are the parameters of $\mathcal{T}$ and let C_i be their corresponding corollas. We then have $FV(C_i) = FV(\mathcal{T}_i) = Y_i$. The fact that the set of free variables of each corolla is recorded by the data of the corresponding parameter allows us to shorten the notation by writing $\mathcal{T}_i$ without listing explicitly the elements of $FV(\mathcal{T}_i)$. For example, for the tree from the latter case above, we shall write $\{\{f(x,y,\dots),g(u,\dots),\dots;\sigma_1\},\dots,\{(z,t);id\},\dots;\sigma\}$. We shall extend this abbreviation to trees in $\mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}$, and when the form of the parameters of a two-level tree is irrelevant, we shall write $\{\mathcal{T}_1,\dots,\mathcal{T}_n,\, s_1,\dots,s_m;\sigma\}$.*

The flattening is defined by:

- $\mathit{flat}(\{(x,y);id\}) = \{(x,y);id\}$, and
- if $\mathcal{T} = \{\{f(x,y,\dots),g(u,\dots),\dots;\sigma_1\},\dots,\{(z,t);id\},\dots;\sigma\}$, then

$$\mathit{flat}(\mathcal{T}) = \{f(x,y,\dots),g(u,\dots),\dots,(z,t),\dots;\underline{\sigma}\},$$

where, having denoted with $\mathcal{T}_i$, $1 \leq i \leq n$, the corollas of $\mathcal{T}$, and with σ_i the corresponding involutions,

$$\underline{\sigma}(x) = \begin{cases} \sigma(x) & \text{if } x \in \bigcup_{i=1}^{n} FV(\mathcal{T}_i) \\ \sigma_i(x) & \text{if } x \in V(\mathcal{T}_i) \backslash FV(\mathcal{T}_i)\,. \end{cases}$$

In simple terms, the flattening of a tree in obtained by removing the brackets that delimit its corollas, and expanding the domain of the involution in the natural way.

Remark 4. *Observe that $\mathit{flat}(\mathcal{T})$ is an extended Vernon tree whenever $\mathcal{T}$ contains a corolla that is an exceptional Vernon tree. These are the cases that make a gap between the flattening function and the action of the monad multiplication.*

The complete characterisation of the monad multiplication is obtained by combining the flattening and the assignment $\mathcal{T} \mapsto [nf(\mathcal{T})]_\alpha$, as follows. If $[\mathcal{T}]_\alpha \in \mathcal{MM}(\mathcal{C})(X)$, then $\mu_{\mathcal{C}X} : \mathcal{MM}(\mathcal{C})(X) \to \mathcal{M}(\mathcal{C})(X)$ first flattens $\mathcal{T}$, and then takes the unique α-equivalence class determined by a normal form of the obtained tree, i.e.

$$\mu_{\mathcal{C}X} : [\mathcal{T}]_\alpha \mapsto [nf(\mathit{flat}(\mathcal{T}))]_\alpha.$$

The domain of the above definition of flattening can be extended in a natural way so that it covers all the trees from $\mathcal{M}'\mathcal{M}'(\mathcal{C})$, where $\mathcal{M}'(\mathcal{C})(X) = \mathtt{eVT}_{\mathcal{C}}(X)$. The clause that needs to be added to encompass $\mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}(X)$ concerns two-level trees of the form

$$\{\{f(x,y,\dots),g(u,\dots),(v,w)\dots;\sigma_1\},\dots,\{(z,t);id\},\dots,(a,b),\dots;\sigma\},$$

i.e. Vernon trees whose set of corollas allows special corollas and extended Vernon trees. Let us denote with $\mathcal{T}$ the above tree, and let $Cor_s(\mathcal{T})$ be the set of its special corollas. The flattening of $\mathcal{T}$ is defined simply as

$$\mathit{flat}(\mathcal{T}) = \{f(x, y, \dots), g(u, \dots), (v, w) \dots, (z, t), \dots, (a, b), \dots; \underline{\sigma}\},$$

with $\underline{\sigma}$ being defined exactly like before for the variables coming from $Cor(\mathcal{T}) \backslash Cor_s(\mathcal{T})$, while we set $\underline{\sigma}(x) = \sigma(x)$ for all variables $x \in \bigcup_{s \in Cor_s(\mathcal{T})} FV(s)$.

For Vernon trees $\mathcal{T}$ and $\mathcal{T}'$ of $\mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}$, the following two lemmas give conditions that ensure that $\mathit{flat}(\mathcal{T}) \to \mathit{flat}(\mathcal{T}')$, in the instance $(\mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}, \to)$ of the rewriting system defined before.

Lemma 4*. *For $\mathcal{T}, \mathcal{T}' \in \mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}$, if $\mathcal{T} \to \mathcal{T}'$, then $\mathit{flat}(\mathcal{T}) \to \mathit{flat}(\mathcal{T}')$.*

Lemma 5*. *For $\{\mathcal{T}_1, \dots, \mathcal{T}_n, s_1, \dots, s_m; \sigma\} \in \mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}$ and $1 \leq j \leq n$, if $\mathcal{T}_j \to \mathcal{T}'_j$, then*

$$\mathit{flat}(\{\mathcal{T}_1, \dots, \mathcal{T}_j, \dots, \mathcal{T}_n, s_1, \dots, s_m; \sigma\}) \to \mathit{flat}(\{\mathcal{T}_1, \dots, \mathcal{T}'_j, \dots, \mathcal{T}_n, s_1, \dots, s_m; \sigma\}).$$

Lemma 6. *For $\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_n, s_1, \dots, s_m; \sigma\} \in \mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}$ the following claims hold:*

a) $\mathit{nf}(\mathit{flat}(\mathcal{T})) =_\alpha \mathit{nf}(\mathit{flat}(\mathit{nf}(\mathcal{T})))$,

b) $\mathit{nf}(\mathit{flat}(\mathcal{T})) =_\alpha \mathit{nf}(\mathit{flat}(\{\mathit{nf}(\mathcal{T}_1), \dots, \mathit{nf}(\mathcal{T}_n), s_1, \dots, s_m; \sigma\}))$.

Proof. By the termination of $(\mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}, \to)$, we have $\mathcal{T} \to \mathit{nf}(\mathcal{T})$, and then, by Lemma 4 and the termination of $(\mathtt{eVT}_{\mathcal{C}}, \to)$, we know that, in $(\mathtt{eVT}_{\mathcal{C}}, \to)$,

$$\mathit{flat}(\mathcal{T}) \to \mathit{flat}(\mathit{nf}(\mathcal{T})) \to \mathit{nf}(\mathit{flat}(\mathit{nf}(\mathcal{T}))).$$

On the other hand, by the termination of $(\mathtt{eVT}_{\mathcal{C}}, \to)$, we also have that $\mathit{flat}(\mathcal{T}) \to \mathit{nf}(\mathit{flat}(\mathcal{T}))$. Therefore, the first claim follows by the confluence of $(\mathtt{eVT}_{\mathcal{C}}, \to)$.

As for the second claim, by the termination of $(\mathtt{eVT}_{\mathcal{C}}, \to)$, we have $\mathcal{T}_i \to \mathit{nf}(\mathcal{T}_i)$, for all $i \in I$. Hence, by Lemma 5, and then again by the termination of $(\mathtt{eVT}_{\mathcal{C}}, \to)$, we get that

$$\begin{array}{rcl} \mathit{flat}(\mathcal{T}) & \to & \mathit{flat}(\{\mathit{nf}(\mathcal{T}_1), \dots, \mathit{nf}(\mathcal{T}_n), s_1, \dots, s_m; \sigma\}) \\ & \to & \mathit{nf}(\mathit{flat}(\{\mathit{nf}(\mathcal{T}_1), \dots, \mathit{nf}(\mathcal{T}_n), s_1, \dots, s_m; \sigma\})) \end{array}$$

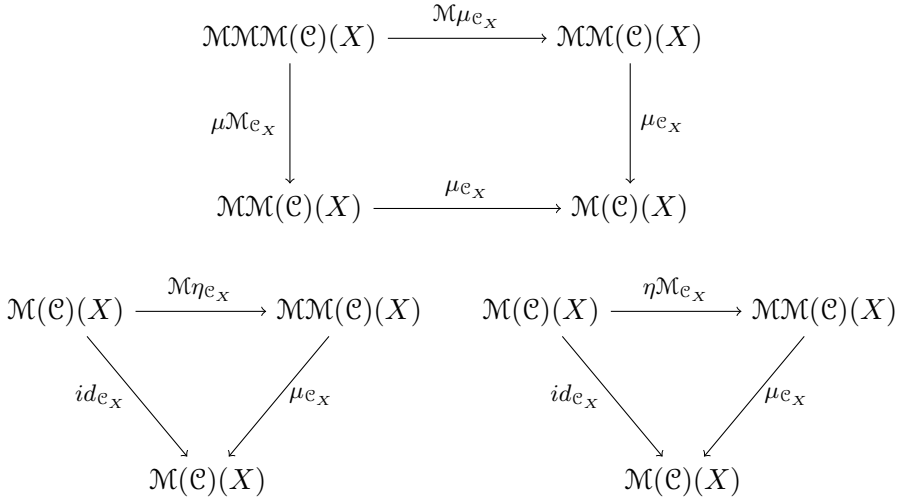
is a reduction sequence of $(\mathtt{eVT}_{\mathcal{C}}, \to)$. The conclusion follows as in the previous claim. □

On the other hand, by the very definition of flattening on extended Vernon trees, we have the following property.

Lemma 7. *For $\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_n; \sigma\} \in \mathtt{VT}_{\mathtt{eVT}_{\mathtt{eVT}_{\mathcal{C}}}}$ the following equality holds:*

$$\mathit{flat}(\mathit{flat}(\mathcal{T})) = \mathit{flat}(\{\mathit{flat}(\mathcal{T}_1), \dots, \mathit{flat}(\mathcal{T}_n); \sigma\}).$$

Theorem 2. *For natural transformations $\mu : \mathcal{MM} \to \mathcal{M}$ and $\eta : id \to \mathcal{M}$, the following diagrams commute for every functor $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$ and a finite set X:*



Proof. We first verify the commutation of the upper diagram. Chasing the associativity of multiplication includes treating several cases, according to the shape of the member of

$$\mathcal{MMM}(\mathcal{C})(X) = \mathtt{VT}_{\mathtt{VT}_{\mathtt{VT}_{\mathcal{C}}}}(X)$$

we start from. The most interesting is the one starting from (a class determined by) an ordinary Vernon tree with corollas given by ordinary Vernon trees build upon $\mathtt{VT}_{\mathcal{C}}$ and we prove the associativity only for this case. Let, therefore, $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n; \sigma\}$.

By chasing the diagram to the right, the action of $\mathcal{M}\mu_{\mathcal{C}_X}$ corresponds to corolla per corolla flattening of $\mathcal{T}$, followed by taking the respective normal forms. Then μ flattens additionally the resulting tree and reduces it to a normal form. Therefore, we have the following sequence on the right hand side:

$$\begin{array}{rcl} \mathcal{T} & \mapsto & \{\mathit{flat}(\mathcal{T}_1), \ldots, \mathit{flat}(\mathcal{T}_n); \sigma\} \\ & \mapsto & \{\mathit{nf}(\mathit{flat}(\mathcal{T}_1)), \ldots, \mathit{nf}(\mathit{flat}(\mathcal{T}_n)); \sigma\} \\ & \mapsto & \mathit{flat}(\{\mathit{nf}(\mathit{flat}(\mathcal{T}_1)), \ldots, \mathit{nf}(\mathit{flat}(\mathcal{T}_n)); \sigma\}) \\ & \mapsto & \mathit{nf}(\mathit{flat}(\{\mathit{nf}(\mathit{flat}(\mathcal{T}_1)), \ldots, \mathit{nf}(\mathit{flat}(\mathcal{T}_n)); \sigma\})) = R \end{array}$$

The action $\mu\mathcal{M}_{\mathcal{C}_X}$ on the left hand side corresponds to the action of μ on the tree $\mathcal{T}$ itself, which flattens it and reduces it to a normal form. Followed by μ again, this gives us the following sequence:

$$\begin{array}{rcl} \mathcal{T} & \mapsto & \mathit{flat}(\mathcal{T}) \\ & \mapsto & \mathit{nf}(\mathit{flat}(\mathcal{T})) \\ & \mapsto & \mathit{flat}(\mathit{nf}(\mathit{flat}(\mathcal{T}))) \\ & \mapsto & \mathit{nf}(\mathit{flat}(\mathit{nf}(\mathit{flat}(\mathcal{T})))) = L \end{array}$$

Let $R' = \mathit{nf}(\mathit{flat}(\{\mathit{flat}(\mathcal{T}_1), \ldots, \mathit{flat}(\mathcal{T}_n); \sigma\}))$ and $L' = \mathit{nf}(\mathit{flat}(\mathit{flat}(\mathcal{T})))$. By the claims of Lemma 6, we have that $R = R'$ and $L = L'$, and by Lemma 7 we have $R' = L'$.

We now verify the unit laws for the case when $[\mathcal{T}]_\alpha \in \mathcal{M}(\mathcal{C})(X)$ is determined by an ordinary Vernon tree. Let, therefore, $\mathcal{T} = \{f_1(x_1, \ldots, x_k), \ldots, f_n(y_1, \ldots, y_r); \sigma\}$.

By going to the right in the first diagram, the action of $\mathcal{M}_{\eta \mathcal{C}_X}$ turns each corolla f_i into a single-corolla Vernon tree $\mathcal{T}_i$, leading to a two-level Vernon tree, which is then flattened and reduced to a normal form by μ. Therefore, the right-hand side sequence is as follows:

$$\begin{array}{rcl} \mathcal{T} & \mapsto & \{\{f_1(x_1,\ldots,x_k),id\},\ldots,\{f_n(y_1,\ldots,y_r);id\};\sigma\} \\ & \mapsto & \{f_1(x_1,\ldots,x_k),\ldots,f_n(y_1,\ldots,y_r);\underline{\sigma}\} \\ & \mapsto & \{f_1(x_1,\ldots,x_k),\ldots f_n(y_1,\ldots,y_r);\underline{\sigma}'\} \end{array}$$

the resulting tree being exactly $\mathcal{T}$, since

$$\underline{\sigma}'(x) = \underline{\sigma}(x) = \begin{cases} \sigma(x) & \text{if } x \in \bigcup_{i=1}^n FV(\mathcal{T}_i) \\ x & \text{if } x \in V(\mathcal{T}_i)\backslash FV(\mathcal{T}_i) \end{cases} = \begin{cases} \sigma(x) & \text{if } x \in V(\mathcal{T}) \\ x & \text{if } x \in V(\mathcal{T}_i)\backslash FV(\mathcal{T}_i) \end{cases} = \sigma(x),$$

the last equality holding because $V(\mathcal{T}_i)\backslash FV(\mathcal{T}_i) = \emptyset$, for all $1 \leq i \leq n$.

By chasing the second diagram to the right, $\mathcal{T}$ will first be turned, by the action of $\eta\mathcal{M}_{\mathcal{C}_X}$, into a single-corolla two-level tree, which will then be flattened and reduced to a normal form by the action of μ. Therefore, we have the sequence

$$\begin{array}{rcl} \mathcal{T} & \mapsto & \{\{f_1(x_1,\ldots,x_k),\ldots,f_n(y_1,\ldots,y_r);\sigma\},id\} \\ & \mapsto & \{f_1(x_1,\ldots,x_k),\ldots,f_n(y_1,\ldots,y_r);\underline{id}\} \\ & \mapsto & \{f_1(x_1,\ldots,x_k),\ldots,f_n(y_1,\ldots,y_r);\underline{id}'\} \end{array}$$

For the resulting involution $\underline{id}'$ we have

$$\underline{id}'(x) = \underline{id}(x) = \begin{cases} x & \text{if } x \in FV(\mathcal{T}) \\ \sigma(x) & \text{if } x \in V(\mathcal{T})\backslash FV(\mathcal{T}) \end{cases} = \sigma(x).$$

Therefore, the resulting tree is exactly $\mathcal{T}$. □

2.4 The cyclic operad structure of Vernon trees

We shall now exhibit the *biased* cyclic operad structure on α-equivalence classes of Vernon trees. This structure is the free cyclic operad built over the functor $\mathtt{VT}_{\mathcal{C}} : \mathbf{Bij}^{op} \to \mathbf{Set}$, defined on a finite set X as the set of all α-equivalence classes of Vernon trees whose corollas come from $\mathcal{C}$ and whose free variables are given by the set X (cf. end of 2.1).

In the sequel, for a Vernon tree $\mathcal{T}$ and a bijection $\vartheta : V \to V(\mathcal{T})$, we will denote with $\mathcal{T}^\vartheta$ a Vernon tree obtained from $\mathcal{T}$ by renaming its variables in a way dictated by ϑ and adapting its corollas accordingly. More precisely, if $C \in Cor(\mathcal{T})$, $\mathcal{T}^\vartheta$ will, instead of C, contain the corolla C^ϑ defined as follows.

- If C is an ordinary corolla, say $f \in \mathcal{C}(X)$, then C^ϑ is the ordinary corolla $f^{\vartheta|^X}$.
- If C is a special corolla, say (x, y), then C^ϑ is the special corolla $(\vartheta^{-1}(x), \vartheta^{-1}(y))$.

The involution σ^ϑ of $\mathcal{T}^\vartheta$ is defined as $\sigma^\vartheta(v) = \vartheta^{-1}(\sigma(\vartheta(v)))$, $v \in V$. If $Y \subseteq V(\mathcal{T})$ and $\vartheta : U \to Y$ is a bijection such that $U \cap (V(\mathcal{T})\backslash Y) = \emptyset$, we define $\mathcal{T}^\vartheta$ to be $\mathcal{T}^{\vartheta + id_{V(\mathcal{T})\backslash Y}}$.

For an arbitrary bijection $\kappa : X' \to X$, the image $[\mathcal{T}]_\alpha^\kappa$ of $[\mathcal{T}]_\alpha \in \mathtt{VT}_\mathcal{C}(X)$ under $\mathtt{VT}_\mathcal{C}(\kappa) : \mathtt{VT}_\mathcal{C}(X) \to \mathtt{VT}_\mathcal{C}(X')$ will be the equivalence class $[\mathcal{T}^{\kappa+\varepsilon}]_\alpha$, where $\varepsilon : V \to V(\mathcal{T})\backslash X$ is an arbitrary bijection such that $X' \cap V = \emptyset$. Notice that any two choices of ε (and V) satisfying the required disjointness condition lead to the same equivalence class.

The functoriality of $\mathtt{VT}_\mathcal{C}$ follows easily: for $[\mathcal{T}]_\alpha \in \mathtt{VT}_\mathcal{C}(X)$, we have

$$\mathtt{VT}_\mathcal{C}(id_X)([\mathcal{T}]_\alpha) = [\mathcal{T}^{id_X+\varepsilon}]_\alpha = [\mathcal{T}]_\alpha = id_{\mathtt{VT}_\mathcal{C}(X)}([\mathcal{T}]_\alpha),$$

while for $\tau : X'' \to X'$ and $\kappa : X' \to X$ we have

$$\mathtt{VT}_\mathcal{C}(\kappa \circ \tau)([\mathcal{T}]_\alpha) = [\mathcal{T}^{\kappa\circ\tau+\varepsilon}]_\alpha = [(\mathcal{T}^{\kappa+\varepsilon_1})^{\tau+\varepsilon_2}]_\alpha = \mathtt{VT}_\mathcal{C}(\tau)([\mathcal{T}^{\kappa+\varepsilon_1}]_\alpha) = (\mathtt{VT}_\mathcal{C}(\tau) \circ \mathtt{VT}_\mathcal{C}(\kappa))([\mathcal{T}]_\alpha),$$

where $\varepsilon : V \to V(\mathcal{T})\backslash X$, $\varepsilon_1 : V_1 \to V(\mathcal{T})\backslash X$ and $\varepsilon_2 : V_2 \to V(\mathcal{T}^\kappa)\backslash X'$ are arbitrary bijections such that $V \cap X'' = \emptyset$, $V_1 \cap X' = \emptyset$ and $V_2 \cap X'' = \emptyset$. The equality $[\mathcal{T}^{\kappa\circ\tau+\varepsilon}]_\alpha = [(\mathcal{T}^{\kappa+\varepsilon_1})^{\tau+\varepsilon_2}]_\alpha$ is proved easily by taking V_2 to be precisely V and defining $\varepsilon_2 = \varepsilon_1^{-1} \circ \varepsilon$.

The rest of the cyclic operad structure is built as follows. Let X and Y be non-empty finite sets such that for some $x \in X$ and $y \in Y$ we have $(X\backslash\{x\}) \cap (Y\backslash\{y\}) = \emptyset$, and let $[\mathcal{T}_1]_\alpha \in \mathtt{VT}_\mathcal{C}(X)$, $[\mathcal{T}_2]_\alpha \in \mathtt{VT}_\mathcal{C}(Y)$. The partial composition operation

$$ {}_x\bullet_y : \mathtt{VT}_\mathcal{C}(X) \times \mathtt{VT}_\mathcal{C}(Y) \to \mathtt{VT}_\mathcal{C}((X\backslash\{x\}) \cap (Y\backslash\{y\}))$$

is given as

$$[\mathcal{T}_1]_\alpha \, {}_x\bullet_y \, [\mathcal{T}_2]_\alpha = [nf(\mathcal{T})]_\alpha,$$

where $\mathcal{T}$ is determined as follows.

The set of corollas of $\mathcal{T}$ is obtained by taking the union of the sets of corollas of $\mathcal{T}_1$ and $\mathcal{T}_2$, after having previously adapted them in a way that makes this union disjoint with respect to the variables occuring in it. More precisely, if $\vartheta_1 : V_1 \to (V(\mathcal{T}_1)\backslash X) \cup \{x\}$ and $\vartheta_2 : V_2 \to (V(\mathcal{T}_2)\backslash Y) \cup \{y\}$ are bijections such that $V_1 \cap V_2 = \emptyset$, then

$$Cor(\mathcal{T}) = \{C^{\vartheta_1} \mid C \in Cor(\mathcal{T}_1)\} \cup \{D^{\vartheta_2} \mid D \in Cor(\mathcal{T}_2)\}.$$

The involution σ of $\mathcal{T}$ is defined as follows, wherein σ_1 and σ_2 denote the involutions of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively:

$$\sigma(v) = \begin{cases} \vartheta_1^{-1}(\sigma_1(\vartheta_1(v))) & \text{if } v \in V_1\backslash\vartheta_1^{-1}(x) \\ \vartheta_2^{-1}(y) & \text{if } v = \vartheta_1^{-1}(x) \\ \vartheta_2^{-1}(\sigma_2(\vartheta_2(v))) & \text{if } v \in \vartheta_2^{-1}(y) \\ \vartheta_1^{-1}(x) & \text{if } v = \vartheta_2^{-1}(y) \\ v & \text{if } v \in (X\backslash\{x\}) \cup (Y\backslash\{y\})\,. \end{cases}$$

For an arbitrary two-element set $\{y, z\}$, the distinguished unit element of $\mathtt{VT}_\mathcal{C}(\{y, z\})$ will be the class determined by the exceptional Vernon tree $\{(y, z); id\}$.

Lemma 8*. *The functor $\mathtt{VT}_\mathcal{C}$ together with the operation ${}_x\bullet_y$ and the identities $id_{x,y}$, is a cyclic operad (in the sense of Definition 1).*

The definition of the partial composition operation on classes of Vernon trees formalises what we have so far informally called the action of tree grafting: the *grafting of $\mathcal{T}_1$ and $\mathcal{T}_2$ along $x \in FV(\mathcal{T}_1)$ and $y \in FV(\mathcal{T}_2)$* is precisely $[\mathcal{T}_1]_\alpha \, {}_x\bullet_y \, [\mathcal{T}_2]_\alpha$. Similarly, for $f \in \mathcal{C}(X)$ and $\varphi : x \mapsto (\mathcal{T}_x, \underline{x})$, the *(simultaneous) grafting of the corolla f and the surrounding trees (determined by φ)* is $[\mathcal{T}]_\alpha(\varphi)$, where $\mathcal{T} = \{f(x, y, z, \dots); id\}$, and we will write $f(\varphi)$ instead of $[\mathcal{T}]_\alpha(\varphi)$.

3 μ-syntax

Backed up with the graphical ingredient of the biased cyclic operad structure on classes of Vernon trees, in this section we introduce μ-syntax. For the proof of the main theorem we will be primarily interested in the normal forms of μ-syntax, which we examine in 3.2.

3.1 The language and the equations

Unlike the combinator syntax, which has only one kind of expressions, the μ-syntax features two different kinds of typed expressions

COMMANDS	TERMS
$c ::= \langle s \,|\, t \rangle \;\mid\; \underline{f}\{t_{x_i} \,|\, i \in \{1,\dots,n\}\}$	$s,t ::= x \;\mid\; \mu x.c$

whose respective types we denote with $c : X$ and $X \,|\, s$. Commands mimick operations $f \in \mathcal{C}(X)$, and a judgement $c : X$ should be thought of as a Vernon tree whose free variables are precisely the elements of X. On the other hand, terms represent operations with one selected entry and the role of the set X in a judgement $X \,|\, s$ is to label all entries *except* the selected one. From the tree-wise perspective, this is represented by a Vernon tree whose set of free variables is $X \cup \{x\}$, where x is precisely the variable bound by μ.

The typing rules for terms and commands are as follows:

$$\frac{}{\{x\} \,|\, x} \qquad \frac{f \in \mathcal{C}(\{x_1,\dots,x_n\}) \quad Y_{x_i} \,|\, t_{x_i} \text{ for all } i \in \{1,\dots,n\}}{\underline{f}\{t_{x_i} \,|\, i \in \{1,\dots,n\}\} : \bigcup_{i=1}^{n} Y_{x_i}} \qquad \frac{X \,|\, s \quad Y \,|\, t}{\langle s \,|\, t \rangle : X \cup Y} \qquad \frac{c : X \quad x \in X}{X \backslash \{x\} \,|\, \mu x.c}$$

where, in the second rule, the sets Y_{x_i} are pairwise disjoint, as are X and Y in the third rule. We shall also denote the commands introduced by the second rule as $\underline{f}\{t_x \,|\, x \in X\}$ (with $f \in \mathcal{C}(X)$), or as $\underline{f}\{\sigma\}$, where σ assigns to every $x \in X$ a term t_x. Whenever we use the notation, say $\underline{f}\{t,u\}$, for $f \in \mathcal{C}(\{x,y\})$, it will be clear from the context whether we mean $\underline{f}\{t,u\} = \underline{f}\{\sigma\}$, with $\sigma(x) = t$ and $\sigma(y) = u$, or with σ defined in the other way around.

The way commands are constructed is motivated by the action of the simultaneous and partial tree grafting that we formalised in the last section. The command $\underline{f}\{t_x \,|\, x \in X\}$, introduced by the second rule, should be imagined as the simultaneous grafting of the corolla f and the surrounding trees, determined by $\sigma : x \mapsto t_x$ and the variables bound by μ in each t_x. In the special case when, for some $x \in X$, the corresponding term t_x is a variable, say u, this process of grafting reduces to the renaming of the variable x of the corolla f as u. Therefore, if all terms associated to the elements of X by the rule are variables, say $u, v, w, ...$, then the corresponding command is $\underline{f}\{u,v,w,\dots\}$ and it describes the Vernon tree $\{f^{\sigma}(u,v,w,\dots); id\}$, where $\sigma : \{u,v,w,\dots\} \to X$ is induced by the rule. The command $\langle s \,|\, t \rangle$ describes the grafting of Vernon trees represented by the terms s and t along their variables bound by μ. Therefore, the pattern $\langle \mu x.\_ \,|\, \mu y.\_ \rangle$ corresponds to the partial composition operation ${}_x\bullet_y$ on classes of Vernon trees.

The equations of the μ-syntax are

$\langle s \,|\, t \rangle = \langle t \,|\, s \rangle$	(MU1)	$\mu x.c = \mu y.c[y/x]$	(MU3)
$\langle \mu x.c \,|\, s \rangle = c[s/x]$	(MU2)	$\underline{f}\{t_x \,|\, x \in X\} = \underline{f^{\sigma}}\{t_{\sigma(y)} \,|\, y \in Y\}$	(MU4)

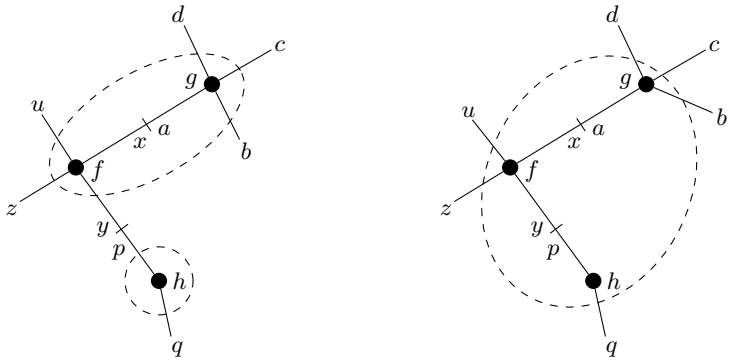
where in (MU2) $c[s/x]$ denotes the command c in which the unique occurrence of x has been replaced by s, in (MU3) y has to be fresh with respect to all variables of c except x, and in (MU4) $\sigma : Y \to X$ is an arbitrary bijection.

The first equation stipulates the symmetry of grafting of Vernon trees, i.e. the commutativity of the partial composition operation (cf. Remark 1), and the last two are α-conversions.

The second equation, defined in terms of substitution, is quite evidently reminescent of the β-reduction of λ-calculus, when considered as a rewriting rule $\langle \mu x.c \,|\, s\rangle \to c[s/x]$, and it essentially captures the same idea of function application as λ-calculus. The intuition becomes more tangible from the point of view of trees: the commands $\langle \mu x.c \,|\, s\rangle$ and $c[s/x]$, equated with (MU2), describe two ways to build (by means of grafting) the same Vernon tree. To see this on a simple example, consider the Vernon tree

$$\mathcal{T} = \{f(x, y, z, u), g(a, b, c, d), h(p, q); \sigma\},$$

where $\sigma = (x\ \ a)(y\ \ p)$. One way to build $\mathcal{T}$ is to graft along y and p Vernon trees $\mathcal{T}_1 = \{f(x, y, z, u), g(a, b, c, d); \sigma_1\}$, where $\sigma_1 = (x\ \ a)$, and $\mathcal{T}_2 = \{h(p, q); id\}$, singled out with dashed lines in the left picture below:



Vernon tree $\mathcal{T}_1$ (in the upper part of the same picture) can itself be seen as a grafting, namely the simultaneous grafting of the corolla f and its surrounding trees: in this case this involves explicit grafting only with the corolla g (along the free variables x and a). This way of constructing $\mathcal{T}$ is described by the command

$$\langle \mu y.\underline{f}\{\mu a.\underline{g}\{a, b, c, d\}, y, z, w\} \,|\, \mu p.\underline{h}\{p, q\}\rangle \qquad (*)$$

that witnesses the fact that $\mathcal{T}_1$ and $\mathcal{T}_2$ are connected along their (selected) free variables y and p, respectively: y and p are bound with μ in the terms corresponding to these two trees. The subterm $\underline{f}\{\mu a.\underline{g}\{a, b, c, d\}, y, z, w\}$ on the left-hand side is the command that accounts for the simultaneous grafting of the corolla f and its surrounding trees, while $\underline{h}\{p, q\}$ on the right-hand side stands for the corolla h. On the other hand, we could have chosen to build this tree by simply taking the simultaneous grafting of the corolla f and its surrounding trees, *surrounded* in the picture on the right. This way of building $\mathcal{T}$ is described with the command $\underline{f}\{\mu a.\underline{g}\{a, b, c, d\}, \mu p.\underline{h}\{p, q\}, z, w\}$, which is, up to substitution, exactly the command

$$\underline{f}\{\mu a.\underline{g}\{a, b, c, d\}, y, z, w\}[\mu p.\underline{h}\{p, q\}/y]$$

to which (*) reduces by applying the rewriting rule $\langle \mu x.c \,|\, s\rangle \to c[s/x]$.

Alternatively, we can think of a command as a prescription for a *decomposition* of a Vernon tree

(more on this at the end of 3.2).

We will denote with $\mu\mathtt{Exp}_{\mathcal{C}}$ the set of all expressions of the μ-syntax induced by $\mathcal{C}$, while its subsets of terms and commands will be denoted by $\mu\mathtt{Term}_{\mathcal{C}}$ and $\mu\mathtt{Comm}_{\mathcal{C}}$, respectively. As in the case of the combinator syntax, the set of expressions (resp. terms and commands) of type X will be denoted by $\mu\mathtt{Exp}_{\mathcal{C}}(X)$ (resp. $\mu\mathtt{Term}_{\mathcal{C}}(X)$ and $\mu\mathtt{Comm}_{\mathcal{C}}(X)$).

3.2 μ-syntax as a rewriting system

Let $\to$ be the rewriting relation defined on expressions of the μ-syntax as (the reflexive, transitive and congruent closure of) the union of rewriting rules

(MU1') $\quad \langle s \,|\, t\rangle \to \langle t \,|\, s\rangle$

(MU2') $\quad \langle \mu x.c \,|\, s\rangle \to c[s/x]$

corresponding to equations (MU1) and (MU2), respectively, which is, moreover, congruent with respect to (MU3) and (MU4).

The non-confluence of the term rewriting system $(\mu\mathtt{Exp}_{\mathcal{C}}, \to)$ shows up immediately: non-joinable critical pairs

$$c_2[\mu x.c_1/y] \longleftarrow \langle \mu x.c_1 \,|\, \mu y.c_2\rangle \longrightarrow c_1[\mu y.c_2/x]$$

arise due to the commutativity rule (MU1'), which makes the whole reduction system symmetric. Nevertheless, all three commands above describe the same Vernon tree.

On the other hand, modulo the trivial commuting conversion, this rewriting system is terminating: as a consequence of linearity of terms, the number of μ-binders in an expression is strictly decreasing at each (MU2') reduction step. It is straightforward to prove that the set $\mu\mathtt{Exp}_{\mathcal{C}}^{nf} = \mu\mathtt{Comm}_{\mathcal{C}}^{nf} \cup \mu\mathtt{Term}_{\mathcal{C}}^{nf}$ of normal forms is generated by the following rules:

$$\frac{}{x \in \mu\mathtt{Term}_{\mathcal{C}}^{nf}} \qquad \frac{f \in \mathcal{C}(X) \quad t_x \in \mu\mathtt{Term}_{\mathcal{C}}^{nf} \text{ for all } x \in X}{\underline{f}\{t_x \,|\, x \in X\} \in \mu\mathtt{Comm}_{\mathcal{C}}^{nf}} \qquad \frac{c \in \mu\mathtt{Comm}_{\mathcal{C}}^{nf}}{\mu x.c \in \mu\mathtt{Term}_{\mathcal{C}}^{nf}}$$

Let us examine the normal forms in relation with Vernon trees. By "unrolling" the recursive definition, we generate the following list of commands in normal form that decompose the tree from the previous axample:

$$\underline{f}\{\mu a.\underline{g}\{a,b,c,d\}, \mu p.\underline{h}\{p,q\}, z, w\},$$
$$\underline{g}\{\mu x.\underline{f}\{x,w,z,\mu p.\underline{h}\{p,q\}\}, b, c, d\}, \quad \underline{g}\{\mu x.\underline{h}\{\mu y.\underline{f}\{x,y,z,w\}, q\}, b, c, d\},$$
$$\underline{h}\{\mu y.\underline{f}\{\mu a.\underline{g}\{a,b,c,d\}, y, z, w\}, q\}, \quad \underline{h}\{\mu y.\underline{g}\{\mu x.\underline{f}\{x,y,z,w\}, b, c, d\}, q\}.$$

Each of the commands records completely the free variables and corollas of this tree: free variables are those not bound with μ (u, z, q, b, c and d), while underlined symbols correspond to the corollas (g, f and h). The bound variables, i.e., variables involved in edges, i.e., here x, a, y, p, can also be recovered. For example, in the first command we get, a, p explicitly bound by μ and x, y implicitly bound by their replacement by a non-variable term.

More generally, the μ-normal forms describe decompositions of Vernon trees of the following kind: pick a node f (i.e., a corolla) of the tree, and then proceed recursively so in all the connected components of the graph resulting from the removal of f . (We provide in 4.1 an algorithmic

computation of these connected components).

Amusingly, one can show that if we instead decide to orient (MU1) the other way around, then the normal forms with respect to this other orientation are in one-to-one correspondence with the combinators of Section 1, and thus describe decompositions of Vernon trees of the following kind: pick an edge e of the tree, and then proceed recursively so in the two connected components of the graph resulting from the removal of e.

These two extremes substantiate our informal explanation of the μ-syntax as a mix of partial composition and simultaneous composition styles.

3.3 The cyclic operad interpretation of μ-syntax

We next formalise the semantic aspect of the μ-syntax relative to Vernon trees, that we brought up in 3.1 and 3.2, by defining an interpretation function of μ-syntax into an arbitrary cyclic operad. We will ascribe meaning to the μ-syntax by first translating it to the combinator syntax given in Section 1.

The translation function

$$[[\_]] : \mu\mathtt{Exp}_{\mathcal{C}} \to \mathtt{cTerm}_{\mathcal{C}}$$

is defined recursively as follows, wherein the assignment of a combinator to a term $t \in \mu\mathtt{Term}_{\mathcal{C}}$ is indexed by a variable that is fresh relative to t.

- $[[x]]_y = id_{x,y}$,
- if, for each $x \in X$, $[[t_x]]_{\overline{x}}$ is a translation of the term t_x, then the translation of the command $\underline{f}\{t_x \,|\, x \in X\}$ is given as
$$[[\underline{f}\{t_x \,|\, x \in X\}]] = f(\varphi),$$
where $f(\varphi)$ denotes the total composition determined by $f \in \mathcal{C}(X)$ and $\varphi : x \mapsto ([[t_x]]_{\overline{x}}, \overline{x})$ (cf. page 6),
- $[[\mu x.c]]_y = [[c[y/x]]]$, and
- $[[\langle s \,|\, t\rangle]] = [[s]]_{x\ x}\circ_y [[t]]_y$.

In order to show that $[[\_]]$ is well-defined, we introduce the following notational conventions. For a command $c : X$ (resp. term $X \,|\, t$) and a bijection $\sigma : X' \to X$, we define

$$c^\sigma := c[\ldots, \sigma^{-1}(x)/x, \ldots] \text{ (resp. } t^\sigma := t[\ldots, \sigma^{-1}(x)/x, \ldots])$$

as a simultaneous substitution (renaming) of the variables from the set X (guided by σ). One of the basic properties of the introduced substitution is the equality $(\mu a.c)^\sigma = \mu a.c^{\sigma_a}$.

The way c^σ is defined clearly indicates that its translation should be the combinator $\mathcal{C}(\sigma)([[c]]) : X'$. The following lemma ensures that this is exactly the case. In its statement, $[[\_]]_X$ will denote the restriction of $[[\_]]$ on the set $\mu\mathtt{Exp}_{\mathcal{C}}(X)$. Furthermore, for a bijection $\sigma : X' \to X$, $(\_)^\sigma : \mathtt{cTerm}_{\mathcal{C}}(X) \to \mathtt{cTerm}_{\mathcal{C}}(X')$ will be the mapping of combinators canonically induced by $\mathcal{C}(\sigma) : \mathcal{C}(X) \to \mathcal{C}(X')$.

Lemma 9*. *For an arbitrary bijection $\sigma : X' \to X$ the following diagram commutes:*

$$\begin{array}{ccc} \mu\mathtt{Exp}_{\mathcal{C}}(X) & \xrightarrow{(\_)^\sigma} & \mu\mathtt{Exp}_{\mathcal{C}}(X') \\ \Big\downarrow{\scriptstyle [[\_]]_X} & & \Big\downarrow{\scriptstyle [[\_]]_{X'}} \\ \mathtt{cTerm}_{\mathcal{C}}(X) & \xrightarrow{(\_)^\sigma} & \mathtt{cTerm}_{\mathcal{C}}(X') \end{array}$$

In other words,

a) for $t \in \mu\mathtt{Term}_{\mathcal{C}}(X)$, $[[t^\sigma]]_y = [[t]]_y^{\sigma_y}$, *and*

b) for $c \in \mu\mathtt{Comm}_{\mathcal{C}}(X)$, $[[c^\sigma]] = [[c]]^\sigma$.

To verify the soundness of $[[\_]]$ we also need the following result.

Lemma 10*. *Let* $X \cap Y = \emptyset$, $t \in \mu\mathtt{Term}_{\mathcal{C}}(Y)$ *and* $x \in X$. *Then*

a) for $s \in \mu\mathtt{Term}_{\mathcal{C}}(X)$, $[[s[t/x]]]_u = [[s]]_{u\ x}\circ_v [[t]]_v$, *and*

b) for $c \in \mu\mathtt{Comm}_{\mathcal{C}}(X)$, $[[c[t/x]]] = [[c]]_{\ x}\circ_v [[t]]_v$.

The equation (MU1) is obviously valid in the world of combinators (see Remark 1). As for (MU2), we have, using the previous lemma:

$$\begin{array}{rcccccl} [[\langle \mu x.c \,|\, t\rangle]] & = & [[\mu x.c]]_{u\ u}\circ_v [[t]]_v & = & [[c[u/x]]]_{\ u}\circ_v [[t]]_v & & \\ & = & [[c]]^{id^{u/x}}_{\ \ u}\circ_v [[t]]_v & = & [[c]]_{\ x}\circ_v [[t]]_v & = & [[c[t/x]]], \end{array}$$

For the two equations expressing the α-convertion laws, we have

$$[[\mu x.c]]_u = [[c[u/x]]] = [[c[y/x][u/y]]] = [[\mu y.c[y/x]]]_u$$

and

$$[[\underline{f}^\sigma\{t_{\sigma(y)} \,|\, y \in Y\}]] = f^\sigma(\varphi') = f^\sigma(\varphi \circ \sigma) = f(\varphi) = [[\underline{f}\{t_x \,|\, x \in X\}]],$$

where $\varphi' : y \mapsto ([[t_{\sigma(y)}]]_{\overline{\sigma(y)}}, \overline{\sigma(y)})$ and $\varphi : \sigma(y) \mapsto ([[t_{\sigma(y)}]]_{\overline{\sigma(y)}}, \overline{\sigma(y)})$. This proves the following theorem.

Theorem 3. *The translation function* $[[\_]] : \mu\mathtt{Exp}_{\mathcal{C}} \to \mathtt{cTerm}_{\mathcal{C}}$ *is well-defined, i.e., it induces a map from* $\mu\mathtt{Exp}_{\mathcal{C}}/_{=_\mu}$ *to* $\mathtt{cTerm}_{\mathcal{C}}/_{=}$, *where* $=_\mu$ *(resp.* $=$*) is the smallest equality relation on* $\mu\mathtt{Exp}_{\mathcal{C}}$ *(resp.* $\mathtt{cTerm}_{\mathcal{C}}$*) generated by the equations of* μ*-syntax (resp. by the equations of Definition 1).*

We now define the interpretation of μ-syntax into an arbitrary (biased) cyclic operad $\mathcal{C}$ as the composition $[\,[[\_]]\,]_{\mathcal{C}} : \mu\mathtt{Exp}_{\mathcal{C}} \to \mathcal{C}$.

It can be actually shown that the translation function induces a bijection from $\mu\mathtt{Exp}_{\mathcal{C}}/_{=_\mu}$ to $\mathtt{cTerm}_{\mathcal{C}}/_{=}$. An inverse translation from combinators to commands is obtained via the correspondence

$$\_\ {}_x\circ_y\ \_ \quad \mapsto \quad \langle \mu x.\_ \,|\, \mu y.\_ \rangle.$$

4 Putting everything together

In 4.1 we prove that the free cyclic operad can be described in terms of the μ-syntax, which is the basis of the proof of Theorem 1 in 4.2. We finish by taking profit of the main result to complement the intuitive description of the monad of unrooted trees from 1.2.2.

4.1 μ-syntax does the job!

The theorem below, together with Lemma 2, puts the μ-syntax in line with already established approaches for defining a cyclic operad.

Theorem 4. *The quotient set of the commands of the μ-syntax relative to the relation $=_\mu$, is in one-to-one correspondence with the one of Vernon trees relative to the α-conversion. In other words, for every finite set X, there exists a categorical isomorphism*

$$\mu\mathtt{Comm}_{\mathcal{C}}(X)/_{=_\mu} \simeq \mathtt{VT}_{\mathcal{C}}(X)/_{\alpha}.$$

The proof goes through a new equality $='$ on normal forms of the μ-syntax, which will be the key for establishing the injectivity of the correspondence. Namely, the proof of the injectivity involves certain decompositions of Vernon trees and their equivalence and $\mu\mathtt{Comm}^{nf}_{\mathcal{C}}(X)/_{='}$ (which is better understood in the broader context of the full μ-syntax) provides a crisp way to encompas them. We first describe these decompositions and the new equality and we then prove the theorem.

4.1.1 "Pruning" of Vernon trees

We shall describe an algorithm that takes an ordinary Vernon tree $\mathcal{T} \in \mathtt{VT}_{\mathcal{C}}(X)$, a corolla $C \in Cor(\mathcal{T})$ and a variable $v \in FV(C)\backslash X$ and returns a proper subtree $\mathcal{T}_v$ of $\mathcal{T}$, the subtree "plucked" from C at the junction of v and $\sigma(v)$. Here, by a proper subtree of $\mathcal{T}$, we mean a proper connected subgraph of $\mathcal{T}$. In the sequel, for an arbitrary corolla $D \in Cor(\mathcal{T})$ and $x \in FV(D)\backslash X$, $S_x(D)$ will denote the corolla adjacent to D along the edge $(x, \sigma(x))$.

We first specify how to generate a set $Cor(\mathcal{T}_v)^+$ of pairs of a corolla of $\mathcal{T}_v$ and one of its free variables, by the following formal rules:

$$\frac{}{(S_v(C), \sigma(v)) \in Cor(\mathcal{T}_v)^+} \qquad \frac{(D,u) \in Cor(\mathcal{T}_v)^+ \quad x \in FV(D)\backslash(X \cup \{u\})}{(S_x(D), \sigma(x)) \in Cor(\mathcal{T}_v)^+}$$

Remark 5. *This system has the following properties.*

1. *Each element $(S_x(D), \sigma(x)) \in Cor(\mathcal{T}_v)^+$ is such that $S_x(D)$ is adjacent to D in $\mathcal{T}$.*
2. *For each $(D, u) \in Cor(\mathcal{T}_v)^+$ we have that $D \neq C$.*

We obtain the set of corollas of $\mathcal{T}_v$ by erasing from the elements of the set $Cor(\mathcal{T}_v)^+$ the data about the distinguished free variables, i.e. we define

$$Cor(\mathcal{T}_v) = \{D \mid (D,u) \in Cor(\mathcal{T}_v)^+ \text{ for some } u \in FV(D)\}.$$

The involution $\sigma_{\mathcal{T}_v}$ of $\mathcal{T}_v$ is defined as

$$\sigma_{\mathcal{T}_v}(x) = \begin{cases} \sigma(x) & \text{if } x \in \big(\bigcup_{D \in Cor(\mathcal{T}_v)} FV(D)\big) \backslash \sigma(v) \\ x & \text{if } x = \sigma(v)\,. \end{cases}$$

We will denote the algorithm with $\mathcal{P}$, and the result $\mathcal{P}(\mathcal{T}, C, v)$ of instatiating $\mathcal{P}$ on a tree $\mathcal{T}$, a corolla $C \in Cor(\mathcal{T})$, and a variable $v \in FV(C) \backslash FV(\mathcal{T})$ will often be denoted as $\mathcal{T}_v$, as we have just done above. The following claim guarantees that $\mathcal{P}$ is correct.

Lemma 11. *$\mathcal{T}_v$ is a proper subtree of $\mathcal{T}$.*

Proof. By the construction we have that $Cor(\mathcal{T}_v) \subseteq Cor(\mathcal{T})$ and that $\mathcal{T}_v$ is connected. By the claim (2) of Remark 5, it follows that $Cor(\mathcal{T}_v)$ is a proper subset of $Cor(\mathcal{T})$. Finally, since $\sigma_{\mathcal{T}_v} = \sigma$ on $V(\mathcal{T}_v) \backslash FV(\mathcal{T}_v)$, we can conclude that $\mathcal{T}_v$ is indeed a subtree of $\mathcal{T}$. □

Corollary 1. *For a Vernon tree $\mathcal{T} \in \mathtt{VT}_{\mathcal{C}}(X)$ and a corolla $C \in Cor(\mathcal{T})$,*

$$\mathcal{P}(\mathcal{T}, C) := \{C; id\} \cup \{\mathcal{T}_v \mid v \in FV(C) \backslash X\}$$

is a decomposition of the tree $\mathcal{T}$ into disjoint subtrees.

Proof. We trivially have that $\{C; id\}$ is a subtree of $\mathcal{T}$ and, by the previous lemma, we know that this is also true for all $\mathcal{T}_v$, where $v \in FV(C) \backslash X$.

By the claim (2) of Remark 5, it follows that $\{C; id\}$ and $\mathcal{T}_v$ are disjoint, for all $v \in FV(C) \backslash X$. Since $\mathcal{T}$ does not contain any cycles, this is also true for arbitrary $\mathcal{T}_u$ and $\mathcal{T}_v$, with $u, v \in FV(C) \backslash X$. Namely, if D would be a corolla of both $\mathcal{T}_u$ and $\mathcal{T}_v$, then the concatenation of the path from C to D in $\mathcal{T}_u$ (starting with the edge $(u, \sigma(u))$) and the path from D to C in $\mathcal{T}_v$ (ending with the edge $(v, \sigma(v))$) would be a cycle of $\mathcal{T}$.

What remains to be shown is that for any corolla D of $\mathcal{T}$ we have that either $D = C$, or there exists $v \in FV(C) \backslash X$ such that $D \in Cor(\mathcal{T}_v)$. Suppose that $D \neq C$. By the connectedness of $\mathcal{T}$, we know that there exist a path p between C and D. Let $v \in FV(C) \cap p$ and $u \in FV(D) \cap p$ be the ending half-edges of p. We prove by induction on the length n of p that $(D, u) \in Cor(\mathcal{T}_v)^+$. If $n = 1$, then $(D, u) = (S_v(C), \sigma(v)) \in Cor(\mathcal{T}_v)^+$. Suppose that the claim holds for all pairs $(D', u') \in Cor(\mathcal{T}_v)^+$ such that the length of the path p' between C and D' is less than n, where $u' \in FV(D') \cap p'$ is the ending half-edge of p'. By the induction hypothesis, we have that $(S_u(D), w) \in Cor(\mathcal{T}_v)^+$, where $w \in FV(S_u(D)) \cap p$. But then, since $\sigma(u) \in FV(S_u(D)) \backslash (X, \{w\})$, we also have that

$$(D, u) = (S_{\sigma(u)}(S_u(D)), \sigma(\sigma(u))) \in Cor(\mathcal{T}_v)^+.$$

Therefore, D is indeed a corolla of $Cor(\mathcal{T}_v)$. □

Lemma 12. *Let $C = f(x_1, \ldots, x_n)$ and $I = \{i_1, \ldots i_k\} = \{i \in \{1, \ldots, n\} \mid x_i \in FV(C) \backslash X\}$. Then, if $\{C; id\} \cup \{\mathcal{T}_{x_i} \mid i \in I\}$ is the decomposition of $\mathcal{T}$ obtained by the algorithm, we have that*

$$[\mathcal{T}]_\alpha = (([\{f(x_1, \ldots, x_n); id\}]_\alpha \, {}_{x_{i_1}}\bullet_{\sigma(x_{i_1})} [\mathcal{T}_{x_{i_1}}]_\alpha) \cdots) \, {}_{x_{i_k}}\bullet_{\sigma(x_{i_k})} [\mathcal{T}_{x_{i_k}}]_\alpha\,.$$

Proof. By induction on the size of $\mathcal{T}$. If C is the only corolla of $\mathcal{T}$, then the decomposition obtained by the algorithm is $\{f(x_1, \ldots, x_n); id\}$ and the claim holds trivially.

Suppose that $\mathcal{T}$ has k corollas, $k \geq 2$, and that the claim holds for all proper subtrees of $\mathcal{T}$ that

contain the corolla C. Since there exists at least one corolla other than C in $\mathcal{T}$, we know that there exists $1 \leq j \leq n$ such that $x_j \in FV(C)\backslash X$. Let $\mathcal{T}'$ be a Vernon tree whose set of corollas is

$$Cor(\mathcal{T}') = \{C\} \cup \{\, Cor(\mathcal{T}_{x_i}) \,|\, i \in I\backslash\{j\}\}$$

and whose involution σ' is defined as

$$\sigma'(x) = \begin{cases} \sigma(x) & \text{if } x \in FV(C)\backslash\{x_j\} \cup \bigcup_{D \in Cor(\mathcal{T}')\backslash\{C\}} FV(D) \\ x & \text{if } x = x_j\,. \end{cases}$$

Clearly, $\mathcal{T}'$ is a proper subtree of $\mathcal{T}$, and, by the induction hypothesis, we have

$$[\mathcal{T}']_\alpha = (([\{f(x_1, \ldots, x_n); id\}]_\alpha \, {}_{x_{i_1}}\bullet_{\sigma(x_{i_1})} [\mathcal{T}_{x_{i_1}}]_\alpha) \cdots) \, {}_{x_{i_k}}\bullet_{\sigma(x_{i_k})} [\mathcal{T}_{x_{i_k}}]_\alpha,$$

where $i_1, \ldots, i_k \in I\backslash\{j\}$. The claim holds since $[\mathcal{T}_\alpha] = [\mathcal{T}']_\alpha \, {}_{x_j}\bullet_{\sigma(x_j)} [\mathcal{T}_{x_j}]_\alpha$. □

Lemma 13. *If $\mathcal{T}$ is a Vernon tree that has at least two corollas, then there exists $D \in Cor(\mathcal{T})$ such that $FV(D)\backslash FV(\mathcal{T})$ is a singleton.*

Proof. Let σ be the involution of $\mathcal{T}$. We prove the claim by induction on the number n of corollas of $\mathcal{T}$. For the base case, suppose that $Cor(\mathcal{T}) = \{C_1, C_2\}$. Then, by the connectedness and the absence of cycles in $\mathcal{T}$, we know that there exist $x \in FV(C_1)$ and $y \in FV(C_2)$ such that $\sigma(x) = y$, while all other variables of $\mathcal{T}$ are fixpoints of σ. Hence, $FV(C_1)\backslash FV(\mathcal{T}) = \{x\}$ and $FV(C_1)\backslash FV(\mathcal{T}) = \{x\}$, i.e. C_1 and C_2 both satisfy the claim.

Assume now that $\mathcal{T}$ has n corollas, where $n > 2$. Let C be a corolla of $\mathcal{T}$ such that there exists $v \in FV(C)\backslash FV(X)$. If v is the unique such element, we are done. If not, let $\{C; id\} \cup \{\mathcal{T}_v \,|\, v \in FV(C)\backslash FV(\mathcal{T})\}$ be the decomposition of $\mathcal{T}$ obtained by applying $\mathcal{P}$ on $\mathcal{C}$. Let $v \in FV(C)\backslash FV(\mathcal{T})$ be fixed. Then, if $Cor(\mathcal{T}_v) = \{S_v(C)\}$, by the definition of $\mathcal{P}$ we know that $FV(S_v(C))\backslash(FV(\mathcal{T}) \cup \{\sigma(v)\}) = \emptyset$, i.e. that $FV(S_v(C))\backslash FV(\mathcal{T}) = \{\sigma(v)\}$. Therefore, since $Cor(\mathcal{T}_v) \subseteq Cor(\mathcal{T})$, $S_v(C)$ is a corolla that satisfies the claim. On the other hand, if $\mathcal{T}_v$ contains more than one corolla, by the induction hypothesis on $\mathcal{T}_v$, we get $D \in Cor(\mathcal{T}_v)$ such that $FV(D)\backslash FV(\mathcal{T}_v) = \{u\}$. Since $FV(D)\backslash FV(\mathcal{T}) \subseteq FV(D)\backslash FV(\mathcal{T}_v)$, we know that either $FV(D)\backslash FV(\mathcal{T}) = \{u\}$, or $FV(D)\backslash FV(\mathcal{T}) = \emptyset$. The latter is impossible because D would be the only corolla of $\mathcal{T}$. □

Let $\mathcal{T}$ and D be as in the previous lemma, and let $\{v\} = FV(D)\backslash FV(\mathcal{T})$. We will denote with $\mathcal{T}_{/D}$ a Vernon tree whose set of corollas is $Cor(\mathcal{T}_{/D}) = Cor(\mathcal{T})\backslash\{D\}$ and whose involution $\sigma_{/D}$ agrees with the involution σ of $\mathcal{T}$ everywhere, except on $\sigma(v)$, which is a fixpoint of $\sigma_{/D}$. Notice that the previous lemma guarantees that $\mathcal{T}_{/D}$ is well-defined.

We next establish a non-inductive characterisation of the output of the algorithm $\mathcal{P}$.

Lemma 14. *Let $\mathcal{T} \in \mathtt{VT}_{\mathcal{C}}(X)$, $C \in Cor(\mathcal{T})$ and $v \in FV(C)\backslash X$. Let σ be the involution of $\mathcal{T}$. The following are equivalent characterisations of a subtree $\mathcal{T}'$ of $\mathcal{T}$:*

1. *$\mathcal{T}' = \mathcal{P}(\mathcal{T}, C, v)$,*
2. *$\sigma(v) \in FV(\mathcal{T}')$ and $FV(\mathcal{T}')\backslash\{\sigma(v)\} \subseteq X$.*

Proof. That (1) implies (2) is clear. We prove that (2) implies (1) by induction on the number n of corollas of $\mathcal{T}'$. If $n = 1$, then, since $\sigma(v) \in FV(\mathcal{T}')$, $S_v(C)$ is the only corolla of $\mathcal{T}'$ and the conclusion follows since, by the assumption, $FV(\mathcal{T}')\backslash\{\sigma(v)\} = FV(S_v(C))\backslash\{\sigma(v)\} \subseteq X$, i.e. $FV(S_v(C))\backslash\{X \cup \{\sigma(v)\}\} = \emptyset$.

Suppose that $\mathcal{T}'$ has n corollas where $n \geq 2$, and let, by Lemma 13, D be a corolla of $\mathcal{T}'$ such that $FV(D)\backslash FV(\mathcal{T}')$ is a singleton, say $\{u\}$. If $D = S_v(C)$, then it follows easily that $\mathcal{T}' = \mathcal{T}_v$. If not, by applying the induction hypothesis on $\mathcal{T}'_{/D}$, we get that $\mathcal{T}'_{/D} = \mathcal{P}(\mathcal{T}_{/D}, C, v)$. Observe that $(S_u(D), w) \in Cor(\mathcal{T}'_{/D})^+$, for some $w \in FV(S_u(D))$ that is different from $\sigma(u)$. By instantiating $\mathcal{P}$ on $(S_u(D), w)$ and $\sigma(u)$, we get the pair (D, u), and the claim follows because $FV(D)\backslash(X \cup \{u\}) = \emptyset$ (i.e. the algorithm stops) and because $\mathcal{T}'_{/D} \cup \{D; id\}$ is a decomposition of $\mathcal{T}$. □

Lemma 15. *Let $f \in \mathcal{C}(X)$, and let for all $x \in X$, $\gamma : x \mapsto ([\mathcal{T}_x]_\alpha, \overline{x})$ and $\tau : x \mapsto ([\mathcal{T}'_x]_\alpha, \tilde{x})$ be such that $f(\gamma)$ and $f(\tau)$ are defined. Then, if $f(\gamma) = f(\tau)$, we have that $[\mathcal{T}_x]^\kappa_\alpha = [\mathcal{T}'_x]_\alpha$ for all $x \in X$, where κ renames $\overline{x}$ to $\tilde{x}$.*

Proof. By removing the corolla f from $\mathcal{T} = f(\gamma) = f(\tau)$, we get precisely the set $\mathcal{P}(\mathcal{T}, f)\backslash\{f, id\}$, where $\mathcal{P}(\mathcal{T}, f)$ is the decomposition introduced by Corollary 1. It is straightforward to show that, for each $x \in FV(f)/FV(\mathcal{T})$, $\mathcal{T}_x$ and $\mathcal{T}'_x$ both satisfy the assumptions of Lemma 14, i.e. that they are both equal (up to renaming) to the output $\mathcal{P}(\mathcal{T}, f, x)$ of the algorithm $\mathcal{P}$, which proves the claim. □

We shall need Lemmas 12 and 15 in the proof of Theorem 4.

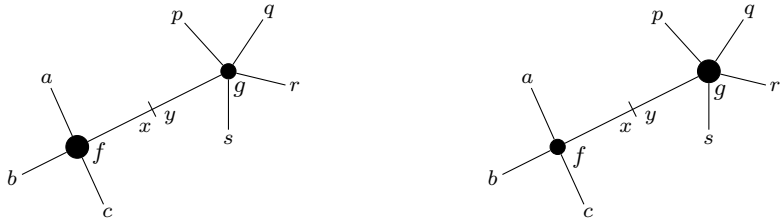
4.1.2 The equality induced on mu-normal forms

Let $f \in \mathcal{C}(X)$ and let $\sigma : x \mapsto t_x$ be an association of terms to variables from the set X such that $\underline{f}\{\sigma\}$ is defined. The equality $='$ is defined as follows:

$$\boxed{\text{if } \sigma(x) = \mu y.c, \quad \text{then } \underline{f}\{\sigma\} =' c[\mu x.f\{\sigma[x/x]\}/y],}$$

where $\sigma[x/x]$ denotes the same association as σ, except for x, to which is now associated x itself.

The intuition behind this equality is about equating commands that reflect two ways to build the same tree. Consider the two pictures below.



The difference between these two graphical realisations of the same Vernon tree is meant to reflect the two possible ways to build this tree by means of simultaneous grafting relative to f and g. As suggested by the larger black dot on f, the picture on the left represents the grafting of the corolla f and its surrounding trees, which we describe in the language of μ-syntax by the command

$\underline{f}\{\mu y.\underline{g}\{y,p,q,r,s\},a,b,c\}$, while the command that reflects the interpretation of the other picture is $\underline{g}\{\mu x.\underline{f}\{x,a,b,c\},p,q,r,s\}$. The new equality says that

$$\underline{f}\{\mu y.\underline{g}\{y,p,q,r,s\},a,b,c\} =' \underline{g}\{\mu x.\underline{f}\{x,a,b,c\},p,q,r,s\}.$$

The proof of the following lemma shows that $='$ is a "macro" for $=_\mu$'s.

Lemma 16. *If* $\underline{f}\{\sigma\} =' c[\mu x.\underline{f}\{\sigma[x/x]\}/y]$, *then* $\underline{f}\{\sigma\} =_\mu c[\mu x.\underline{f}\{\sigma[x/x]\}/y]$.

Proof. If $\underline{f}\{\sigma\} =' c[\mu x.\underline{f}\{\sigma[x/x]\}/y]$, then we know that $\sigma(x) = \mu y.c$, which justifies the following sequence of equalities:

$$\underline{f}\{\sigma\} =_\mu \langle \mu x.\underline{f}\{\sigma[x/x]\} \,|\, \mu y.c\rangle =_\mu \langle \mu y.c \,|\, \mu x.\underline{f}\{\sigma[x/x]\}\rangle =_\mu c[\mu x.\underline{f}\{\sigma[x/x]\}/y]. \qquad \square$$

In the sequel, we shall work with the reflexive and transitive closure of $='$, denoted in the same way. Clearly, the previous lemma holds for this generalisation as well.

Corollary 2. *For any* $c_1, c_2 \in \mu\mathtt{Comm}^{nf}_{\mathcal{C}}(X)$, *if* $c_1 =' c_2$, *then* $c_1 =_\mu c_2$.

The equality $='$ (with different notation) appears in a paper of Lamarche [L07], where it is called Adjunction. There, unlike in our work, it is not derived from a more primitive notion of equality.

4.1.3 Proof of Theorem 4

Let $\Phi : \mu\mathtt{Exp}_{\mathcal{C}} \to \mathtt{VT}_{\mathcal{C}}$ be the composition of the translation function $[[\_]] : \mu\mathtt{Exp}_{\mathcal{C}} \to \mathtt{cTerm}_{\mathcal{C}}$ (defined in 3.3) with the interpretation function $[\_]_{\mathtt{VT}_{\mathcal{C}}} : \mathtt{cTerm}_{\mathcal{C}} \to \mathtt{VT}_{\mathcal{C}}$ (see Lemma 8). Let us show explicitely how Φ looks like. The assignment of an α-equivalence class (of Vernon trees) to a term $t \in \mathtt{Term}_\mu$ will be indexed by a fresh variable y involved in the corresponding interpretation $[[t]]_y$.

- $\Phi_y(x) = [\{(x,y); id\}]_\alpha$,
- if, for each $x \in X$, $\Phi_{\overline{x}}(t_x) = [\mathcal{T}_x]_\alpha$, then $\Phi(\underline{f}\{t_x | x \in X\}) = [f(\varphi)]_\alpha$, where $f(\varphi)$ denotes the total composition determined by the Vernon tree $\{f(x,y,z,\dots); id\}$ and $\varphi : x \mapsto (\mathcal{T}_x, \overline{x})$ (cf. end of Section 2),
- $\Phi_y(\mu x.c) = (\Phi(c))^\kappa$, where κ renames x to y, and
- if $\Phi_x(s) = [\mathcal{T}_s]_\alpha$ and $\Phi_y(t) = [\mathcal{T}_t]_\alpha$, then $\Phi(\langle s \,|\, t\rangle) = [\mathcal{T}_s]_\alpha \, {}_x\!\bullet_y \, [\mathcal{T}_t]_\alpha$.

By Theorem 3, we have that the correspondence $\underline{\Phi} : \mu\mathtt{Comm}_{\mathcal{C}}(X)/_{=_\mu} \mapsto \mathtt{VT}_{\mathcal{C}}(X)/_\alpha$, canonically induced by Φ, is well-defined. We prove that it is both injective and surjective.

To prove the surjectivity, suppose given an α-equivalence class $[\mathcal{T}]_\alpha \in \mathtt{VT}_{\mathcal{C}}(X)/_\alpha$. We differentiate two cases, according to whether $\mathcal{T}$ is an ordinary or an exceptional Vernon tree.

If $\mathcal{T} = \{(x,y); id\}$, we have

$$\Phi(\langle x \,|\, y\rangle) = [\{(x,u); id\}]_\alpha \, {}_u\!\circ_v \, [\{(y,v); id\}]_\alpha = [nf(\{(x,u),(y,v); \sigma\})]_\alpha = [\mathcal{T}]_\alpha,$$

the last of the equalities holding since $\sigma(u) = v$ and x and y are the fixpoints of σ.

Suppose now that $\mathcal{T}$ is an ordinary Vernon tree and let $FV(\mathcal{T}) = X$. We reconstruct a

command whose equivalence class is mapped to $[\mathcal{T}]_\alpha$ by induction on the number of corollas of $\mathcal{T}$. Let $f(x_1,\dots,x_n)$ be an arbitrary corolla of $\mathcal{T}$.

For the base case, assume that $f(x_1,\dots,x_n)$ is the only corolla of $\mathcal{T}$. We then have $\Phi(\underline{f}\{x_1,\dots,x_n\}) = f(\varphi)$, where $\varphi : x_i \mapsto (\varphi_{\overline{x_i}}(x_i), \overline{x_i}) = (\{(x_i,\overline{x_i}); id\}, \overline{x_i})$. Therefore, by the axiom (U1), we have that

$$f(\varphi) = (\cdots([\{f(x_1,\dots,x_n); id\}]_\alpha^{\kappa_1})^{\kappa_2}\cdots)^{\kappa_n},$$

where, trivially, each κ_i is the renaming of x_i to x_i, i.e. the identity on $FV(f)$, and consequently, we have $\Phi(\underline{f}\{x_1,\dots,x_n\}) = [f(x_1,\dots,x_n); id]_\alpha$.

Suppose now that $f(x_1,\dots,x_n)$ is not the only corolla of $\mathcal{T}$, i.e. that $\mathcal{T}$ has k corollas, with $k \geq 2$, and let σ be the involution of $\mathcal{T}$. Suppose also that the claim holds for all ordinary Vernon trees whose number of corollas is less than k. In order to exhibit an appropriate command in this case, we will apply the algorithm $\mathcal{P}$ on $C = f(x_1,\dots,x_n)$ and all $v \in FV(C)\backslash X$ and then apply the induction hypothesis on the resulting subtrees. Let $I_C = \{i \in \{1,\dots,n\} \mid x_i \in FV(C)\backslash X\}$ and $J_C = \{1,\dots,n\}\backslash I_C$. The assignments $(C, x_i) \mapsto \mathcal{T}_{x_i}$ determined by the algorithm for all $i \in I_C$, together with the induction hypothesis for each $\mathcal{T}_{x_i}$, provide us with a set

$$\{c_i \mid i \in I_C \text{ and } \Phi(c_i) = [\mathcal{T}_{x_i}]_\alpha\}.$$

We now set for all $i \in I_C$, $t_{x_i} = \mu\sigma(x_i).c_i$, and for all $j \in J$, $t_{x_j} = x_j$, and we claim that $\Phi(f\{t_x \mid x \in X\}) = [\mathcal{T}]_\alpha$. We have $\Phi(f\{t_x \mid x \in X\}) = f(\varphi)$, where

$$\varphi : x \mapsto \begin{cases} ([\mathcal{T}_{x_i}]_\alpha^{\kappa_i}, z_i) & \text{if } x = x_i \text{ for some } i \in I \\ ([\{(x_j, \underline{x_j}); id\}]_\alpha, \underline{x_j}) & \text{if } x = x_j \text{ for some } j \in J \end{cases}$$

with $[\mathcal{T}_{x_i}]_\alpha^{\kappa_i} = \Phi_{z_i}(\mu\sigma(x_i).c_i)$ being the class associated to the term $\mu\sigma(x_i).c_i$ with respect to the interpretation under the fresh variable z_i. Therefore, if $I_C = \{i_1,\dots,i_{k_I}\}$ and $J_C = \{j_1,\dots,j_{k_J}\}$, then, by the axiom (U1), $\Phi(f\{t_x \mid x \in X\})$ is equal to

$$(\cdots([\{f(x_1,\dots,x_n); id\}]_\alpha^{\kappa_{j_1}\kappa_{j_2}\cdots\kappa_{j_{k_J}}}\, {}_{x_{i_1}}\!\bullet_{z_{i_1}} [\mathcal{T}_{x_{i_1}}]_\alpha^{\kappa_{i_1}})\cdots)\, {}_{x_{i_{k_I}}}\!\bullet_{z_{i_{k_I}}} [\mathcal{T}_{x_{i_{k_I}}}]_\alpha^{\kappa_{i_{k_I}}}$$

where each κ_{j_k}, $1 \leq k \leq k_J$ is the renaming of x_{j_k} to x_{j_k}, i.e. the identity on $FV(f)$, and each κ_{i_k}, $1 \leq k \leq k_I$, is the renaming of $\sigma(x_{i_k})$ to z_{i_k}. Finally, by (EQ), we have that

$$\Phi(f\{t_x \mid x \in X\}) = (([\{f(x_1,\dots,x_n); id\}]_\alpha\, {}_{x_{i_1}}\!\bullet_{\sigma(x_{i_1})} [\mathcal{T}_{x_{i_1}}]_\alpha)\cdots)\, {}_{x_{i_k}}\!\bullet_{\sigma(x_{i_k})} [\mathcal{T}_{x_{i_k}}]_\alpha,$$

and, consequently, by Lemma 12, that $\Phi(f\{t_x \mid x \in X\}) = [\mathcal{T}]_\alpha$.

Notice that, in order to establish the injectivity of $\underline{\Phi}$, it suffices to prove it for commands $c_1, c_2 \in \mu\mathtt{Comm}_{\mathtt{C}}^{nf}(X)$. Indeed, since for an arbitrary command c, by Theorem 3, we know that $[[c]] = [[nf(c)]]$, and consequently that $\Phi(c) = \Phi(nf(c))$, then, from $\Phi(nf(c_1)) = \Phi(c_1) = \Phi(c_2) = \Phi(nf(c_2))$, by the injectivity for commands that are normal form we can conclude that $c_1 =_\mu nf(c_1) =_\mu nf(c_2) =_\mu c_2$. By Corollary 2, the injectivity for normal forms follows if we show that for arbitrary commands $c_1, c_2 \in \mu\mathtt{Comm}_{\mathtt{C}}^{nf}(X)$, if $\Phi(c_1) = \Phi(c_2)$, then $c_1 =' c_2$. We prove the last implication by case analysis with respect to the shapes of c_1 and c_2.

If c_1 and c_2 have the same head symbol, we proceed by induction on the structure of c_1 and c_2. Let us spell out the induction hypothesis here: it tells that, if $\mu x.c_1'$ and $\mu y.c_2'$ are subterms of c_1

and c_2 that are normal forms and if $\Phi_z(\mu x.c_1') = \Phi_z(\mu y.c_2')$, then $c_1'^{\tau_1} =' c_2'^{\tau_2}$, where τ_1 renames x to z and τ_2 renames y to z.

Suppose that $c_1 = \underline{f}\{s_x | x \in X\} = \underline{f}\{\sigma\}$ and $c_2 = \underline{f}\{t_x | x \in X\} = \underline{f}\{\sigma'\}$. The assumption $\Phi(c_1) = \Phi(c_2)$ means that $f(\varphi) =_\alpha f(\psi)$, where $\varphi : x \mapsto (\Phi_{\tilde{x}}(s_x), \tilde{x})$ and $\psi : x \mapsto (\Phi_{\overline{x}}(t_x), \overline{x})$, and consequently, by Lemma 15, that for all $x \in X$, $\Phi_{\tilde{x}}(s_x)^\kappa = \Phi_{\overline{x}}(t_x)$, where κ renames $\tilde{x}$ to $\overline{x}$. Now, the claim holds by the reflexivity of $='$ if all s_x and t_x are variables: if $s_x = u$ and $t_x = v$, then

$$[\{(u, \overline{x}); id\}]_\alpha = (\Phi_{\tilde{x}}(u))^\kappa = \Phi_{\overline{x}}(v) = [\{(v, \overline{x}); id\}]_\alpha,$$

and, therefore, it must be the case that $u = v$.

Suppose, therefore, that $s_x = \mu u.c_x$ and $t_x = \mu v.c_x'$. Then we have

$$[[c_x^{\tau_1}]] = [[c_x]]^{\tau_1} = [[s_x]]^\kappa_{\tilde{x}} = [[t_x]]_{\overline{x}} = [[c_x']]^{\tau_2} = [[c_x'^{\tau_2}]],$$

and consequently $\Phi(c_x^{\tau_1}) = \Phi(c_x'^{\tau_2})$, where τ_1 renames u to $\overline{x}$ and τ_2 renames v to $\overline{x}$. By the induction hypothesis we now have $c_x^{\tau_1} =' c_x'^{\tau_2}$ and, consequently, we get that

$$\begin{array}{rcccccccccc} c_1 & = & \underline{f}\{\sigma\} & =' & c_x[\mu x.\underline{f}\{\sigma[x/x]\}/u] & = & c_x^{\tau_1}[\mu x.\underline{f}\{\sigma[x/x]\}/\overline{x}] & & & & \\ & & & =' & c_x'^{\tau_2}[\mu x.\underline{f}\{\sigma[x/x]\}/\overline{x}] & = & c_x'[\mu x.\underline{f}\{\sigma[x/x]\}/v] & =' & \underline{f}\{\sigma'\} & = & c_2. \end{array}$$

Suppose now that c_1 and c_2 do not have the same head symbol, i.e. that $c_1 = \underline{f}\{s_x | x \in X\} = \underline{f}\{\sigma_1\}$ and $c_2 = \underline{g}\{t_y | y \in Y\} = \underline{g}\{\sigma_2\}$, and let $\Phi(c_1) = [\mathcal{T}_{c_1}]_\alpha$ and $\Phi(c_2) = [\mathcal{T}_{c_2}]_\alpha$. Notice first that, since $\mathcal{T}_{c_1} =_\alpha \mathcal{T}_{c_2}$, there exist a bijection

$$\vartheta : V(\mathcal{T}_{c_2}) \backslash FV(\mathcal{T}_{c_2}) \to V(\mathcal{T}_{c_1}) \backslash FV(\mathcal{T}_{c_1})$$

such that $\mathcal{T}_{c_1}^\vartheta = \mathcal{T}_{c_2}$. Then, since $g \in Cor(\mathcal{T}_{c_2})$, there exists $C \in Cor(\mathcal{T}_{c_1})$ such that $C^\vartheta = g$. Clearly, C must have the shape g^τ, for some bijection $\tau : Y' \to Y$. We can conclude, by the construction of $\mathcal{T}_{c_1}$, that $\underline{g}$ appears in the command c_1. Let $x \in X$ be such that $\sigma_1(x) = \mu z.c$ contains $\underline{g}$. We define the *distance between* $\underline{f}$ *and* $\underline{g}$ *in* c_1 as the natural number $d_{c_1}(f, g)$ determined as follows.

- If $\underline{g}$ is the head symbol of c, then $d_{c_1}(f, g) = 1$.
- If $\underline{h}$ is the head symbol of c, $h \neq g$, then $d_{c_1}(f, g) = d_c(h, g) + 1$.

We prove that $c_1 =' c_2$ by induction on $d_{c_1}(f, g)$. If $d_{c_1}(f, g) = 1$, then, for some $x \in X$ and $y \in Y$, we have that $\sigma_1(x) = \mu y.\underline{g}\{\sigma_2[y/y]\}$ and $\sigma_2(y) = \mu x.\underline{f}\{\sigma_1[x/x]\}$ (since, symmetrically, $\underline{f}$ appears in c_2 and its distance from $\underline{g}$ clearly must also be 1). Therefore,

$$\underline{f}\{\sigma_1\} =' \underline{g}\{\sigma_2[y/y]\}[\mu x.\underline{f}\{\sigma_1[x/x]\}/y] = \underline{g}\{\sigma_2[\mu x.\underline{f}\{\sigma_1[x/x]\}/y]\} = \underline{g}\{\sigma_2\}.$$

If $d_{c_1}(f, g) \geq 2$, then, since $d_{c_1}(f, h) = 1$ (where h is as above), we have that $c_1 =' c[\mu x.f\{\sigma_1[x/x]\}/z]$. On the other hand, by the induction hypothesis for $d_c(h, g) < n$, we have that $c_2 =' c[\mu x.f\{\sigma_1[x/x]\}/z]$, and the conclusion follows by the transitivity of $='$. This completes the proof of Theorem 4.

Note that we have in fact proved *two* categorical isomorphisms:

$$\mu\mathtt{Comm}_{\mathcal{C}}(X)/_{=_\mu} \simeq \mu\mathtt{Comm}_{\mathcal{C}}^{nf}(X)/_{='} \simeq \mathtt{VT}_{\mathcal{C}}(X)/_\alpha,$$

the first isomorphim being induced by nf: indeed, we have that $nf(c_1) =' nf(c_2)$ implies $c_1 = c_2$, and conversely, if $c_1 = c_2$, then $\Phi(nf(c_1) = \Phi(nf(c_2))$ implies $nf(c_1) =' nf(c_2)$.

4.2 The equivalence established

We finally show how all this allows us to prove in full rigor the main theorem of the first section, which, thanks to Lemma 2, can be reformulated as follows.

> *A functor* $\mathcal{C} : \mathbf{Bij}^{op} \to \mathbf{Set}$ *is a cyclic operad (in the sense of the Definition 1) if and only if it is an* $\mathcal{M}$*-algebra (where* $\mathcal{M}$ *is the monad of Vernon trees).*

Suppose first that $(\mathcal{C}, \delta)$ is an $\mathcal{M}$-algebra. The partial composition operation

$$ {}_x\circ_y : \mathcal{C}(X) \times \mathcal{C}(Y) \to \mathcal{C}((X\backslash\{x\}) \cup (Y\backslash\{y\})) $$

is characterised via $\delta : \mathcal{M}(\mathcal{C})((X\backslash\{x\}) \cup (Y\backslash\{y\})) \to \mathcal{C}((X\backslash\{x\}) \cup (Y\backslash\{y\}))$ as

$$ f \,{}_x\circ_y\, g = \delta(\eta(f) \,{}_x\bullet_y\, \eta(g)), $$

where $f \in \mathcal{C}(X)$, $g \in \mathcal{C}(Y)$, $x \in X$, $y \in Y$, η is is the unit of the monad $\mathcal{M}$, and ${}_x\bullet_y$ is the operation on (classes of) Vernon trees defined in 2.4.

As a structure morphism of $\mathcal{M}$-algebra $(\mathcal{C}, \delta)$, δ satisfies the coherence conditions given by commutations of the following two diagrams:

$$ \begin{array}{ccc} \mathcal{M}\mathcal{M}(\mathcal{C}) & \xrightarrow{\mathcal{M}\delta} & \mathcal{M}(\mathcal{C}) \\ {\scriptstyle \mu_{\mathcal{C}}}\downarrow & & \downarrow{\scriptstyle \delta} \\ \mathcal{M}(\mathcal{C}) & \xrightarrow{\delta} & \mathcal{C} \end{array} \qquad \begin{array}{ccc} \mathcal{C} & \xrightarrow{\eta_{\mathcal{C}}} & \mathcal{M}(\mathcal{C}) \\ & {\scriptstyle id_{\mathcal{C}}}\searrow \quad \swarrow{\scriptstyle \delta} & \\ & \mathcal{C} & \end{array} $$

which allows us to verify the axioms from Definition 1 as follows.

As for the proof of the associativity axiom (A1), let f and g be as above, $h \in \mathcal{C}(Z)$ and let $z \in Z$ and $u \in Y$. Suppose also that f, g and h are all different from identity and that X, Y and Z are mutually disjoint (only to avoid the renaming technicalities). We will chase the left diagram from above two times, starting with two-level Vernon trees $\mathcal{T}_1 = \{\{f(x,\dots), g(y,u,\dots); \sigma_1'\}, \{h(z,\dots); id\}; \sigma_1\}$ and $\mathcal{T}_2 = \{\{f(x,\dots); id\}, \{g(y,u,\dots), h(z,\dots); \sigma_2'\}; \sigma_2\}$. Here, $\sigma_1' = (x\,y)$, $\sigma_1 = (u\,z)$, $\sigma_2' = (u\,z)$ and $\sigma_2 = (x\,y)$. If we start with $\mathcal{T}_1$, then, by chasing the diagram to the right, the action of $\mathcal{M}\delta$ corresponds to the action of δ on $\{f(x,\dots), g(y,u,\dots); \sigma_1'\}$ and $\{h(z,\dots); id\}$ separately. Followed by the action of δ again, we get the following sequence

$$ \mathcal{T}_1 \overset{\mathcal{M}\delta}{\longmapsto} \{(f \,{}_x\circ_y\, g)(u,\dots), h(z,\dots); \sigma\} \overset{\delta}{\longmapsto} (f \,{}_x\circ_y\, g) \,{}_u\circ_z\, h. $$

In the other direction, the action of the monad multiplication flattens $\mathcal{T}_1$, the resulting tree already being in normal form. Followed by the action of δ, we obtain the sequence:

$$ \mathcal{T}_1 \overset{\mu_{\mathcal{C}}}{\longmapsto} \{f(x,\dots), g(y,u,\dots), h(z,\dots); \underline{\sigma}\} \overset{\delta}{\longmapsto} \delta(\{f(x,\dots), g(y,u,\dots), h(z,\dots); \underline{\sigma}\}). $$

Hence, $(f \,{}_x\circ_y\, g) \,{}_u\circ_z\, h = \delta(\{f(x,\dots), g(y,u,\dots), h(z,\dots); \underline{\sigma}\})$. The diagram chasing with respect to $\mathcal{T}_2$ gives us that $f \,{}_x\circ_y (g \,{}_u\circ_z h) = \delta(\{f(x,\dots), g(y,u,\dots), h(z,\dots); \underline{\sigma}\})$. Therefore, $(f \,{}_x\circ_y\, g) \,{}_u\circ_z\, h = f \,{}_x\circ_y (g \,{}_u\circ_z h)$. The axiom (A2) is proved similarly.

The axiom (EQ) holds by the equivariance of ${}_x\bullet_y$ and the naturality of η and δ:

$$
\begin{array}{rcl@{\qquad}rcl}
f^{\sigma_1}{}_{\sigma_1^{-1}(x)\circ\sigma_2^{-1}(y)}\, g^{\sigma_2} & = & \delta(\eta(f^{\sigma_1})_{\sigma_1^{-1}(x)\bullet\sigma_2^{-1}(y)}\, \eta(g^{\sigma_2})) & = & \delta(\eta(f)^{\sigma_1}{}_{\sigma_1^{-1}(x)\bullet\sigma_2^{-1}(y)}\, \eta(g)^{\sigma_2}) \\
 & = & \delta((\eta(f)\,{}_x\!\bullet_y \eta(g))^{\sigma}) & = & \delta(\eta(f)\,{}_x\!\bullet_y \eta(g))^{\sigma} \\
 & & & = & (f\,{}_x\!\circ_y g)^{\sigma}.
\end{array}
$$

We prove the unit axioms (U1) and (U3) by the corresponding laws for Vernon trees and naturality of η ((U2)is proved analogously as (U1)):

$$f\,{}_x\!\circ_y id_{y,z} = \delta(\eta(f)\,{}_x\!\bullet_y \eta(id_{y,z})) = \delta(\eta(f)^{\kappa}) = \delta(\eta(f^{\kappa})) = f^{\kappa},$$

$$id^{\sigma}_{x,y} = \delta(\eta(id^{\sigma}_{x,y})) = \delta(\eta(id_{x,y})^{\sigma}) = \delta(\eta(id_{u,v})) = id_{u,v}\,.$$

In the other direction, we define $\delta : \mathcal{M}(\mathcal{C}) \to \mathcal{C}$ as the map induced by the composition of $[[\_]] : \mu\mathtt{Exp}_{\mathcal{C}} \to \mathtt{cTerm}_{\mathcal{C}}$ and $[\_]_{\mathcal{C}} : \mathtt{cTerm}_{\mathcal{C}} \to \mathcal{C}$, i.e. (with Φ as in the proof of Theorem 4):

$$\delta(\mathcal{T}) = [\,[[c]]\,]_{\mathcal{C}}, \quad \text{where } c \text{ is any command such that } \Phi(c) = \mathcal{T}.$$

Note that *this definition is sound, by Theorem 4.* Let us check that δ indeed satisfies the equations for an $\mathcal{M}$-algebra. We do this on simple examples, but the general case follows naturally. Let

$$\mathcal{T} = \{\{f(x,\dots), g(y,u,\dots); \sigma_1\}, \{h(z,\dots); id\}; \sigma\}$$

be a two-level Vernon tree such that $\sigma_1(x) = y$, and $\sigma(u) = z$. and suppose, say, that $\Phi(\underline{f}\{t_x \,|\, x \in X\}) = \{f(x,\dots), g(y,u,\dots); \sigma_1\}$ and $\Phi(\underline{h}\{s_z \,|\, z \in Z\}) = \{h(z,\dots); \sigma_2\}$.

By chasing the first diagram to the right, the action of $\mathcal{M}\delta$ provides the interpretations of the commands that correspond to each of the corollas of $\mathcal{T}$. Thus, setting $[[\underline{f}\{t_x \,|\, x \in X\}]] = f(\varphi)$ and $[[\underline{h}\{s_z \,|\, z \in Z\}]] = h(\tau)$, we get that

$$\mathcal{M}\delta(\mathcal{T}) = \{[f(\varphi)]_{\mathcal{C}}(x,\dots,y,u,\dots), [h(\tau)]_{\mathcal{C}}(z,\dots); \sigma\}.$$

If now $\Phi(\underline{[f(\varphi)]_{\mathcal{C}}}\{k_u \,|\, u \in FV(f(\varphi))\}) = \mathcal{M}\delta(\mathcal{T})$, then setting $[[\underline{[f(\varphi)]_{\mathcal{C}}}\{k_u \,|\, u \in FV(f(\varphi))\}]] = [f(\varphi)]_{\mathcal{C}}(\psi)$, we get

$$\delta(\mathcal{M}\delta(\mathcal{T})) = [f(\varphi)(\psi)]_{\mathcal{C}}.$$

By chasing the diagram to the left, we first get

$$\mu_{\mathcal{C}}(\mathcal{T}) = \{f(x,\dots), g(y,u,\dots), h(z,\dots); \underline{\sigma}\}\,.$$

We shall construct a command c such that $\Phi(c) = \mu_{\mathcal{C}}(\mathcal{T})$ in a way guided by the choices we made in chasing the diagram to the right. More precisely, in that direction, f was the corolla of $\{f(x,\dots), g(y,u,\dots); \sigma_1\}$ chosen in constructing the corresponding command, and h was the one for $\{h(z,\dots); \sigma_2\}$, and then, in the next step, $[f(\varphi)]_{\mathcal{C}}$ was the chosen corolla of $\{[f(\varphi)]_{\mathcal{C}}(x,\dots,y,u,\dots), [h(\tau)]_{\mathcal{C}}(z,\dots); \sigma\}$. Therefore, we set $c = \underline{f}\{\sigma\}$, where $\sigma(x) = \mu y.\underline{g}\{\mu z.\underline{h}\{z,\dots\},\dots\}$. Thus, setting $[[\underline{f}\{\sigma\}]] = f(\xi)$, we get

$$\delta(\mu_{\mathcal{C}}(\mathcal{T})) = [f(\xi)]_{\mathcal{C}}$$

as a result of chasing the diagram to the left. That $f(\varphi)(\psi) = f(\xi)$ now follows directly by the claim (b) of Lemma 1.

As for the second diagram, if $f \in \mathcal{C}(X)$, where $X = \{x_1,\dots,x_n\}$, then $\eta_{\mathcal{C}}(f) = \{f(x_1,\dots,x_n); id\}$, and, since $[\{f(x_1,\dots,x_n); id\}]_{\alpha} = \Phi(\underline{f}\{x_1,\dots,x_n\})$, we have that $\delta(\eta_{\mathcal{C}}(f)) = [\,[[\underline{f}\{x_1,\dots,x_n\}]]\,]_{\mathcal{C}} = f$. This completes the proof.

We can finally take profit of this result and define in a rigorous way the mapping

$$\bigotimes_{v\in Vert(T)} \mathcal{D}(Leg(v)) \xrightarrow{\gamma} \mathcal{D}(Leg(T)),$$

intuitively described in 1.2.2. Let $T_{\mathcal{D}} \in \bigotimes_{v\in Vert(T)} \mathcal{D}(Leg(v))$ be an unrooted, $\mathcal{D}$-decorated tree (with half-edges) and let $\mathcal{T}$ be the Vernon tree that corresponds to it as in the proof of Lemma 2. Any "sequence of iterated applications of operadic composition maps of $\mathcal{D}$ coordinated by $\mathcal{T}$" corresponds to a way of decomposing $\mathcal{T}$, codified by a combinator s. We can assign to s a command c of the μ-syntax by the translation sketched at the end of Section 3. Then the result of applying γ is precisely the interpretation of the command c in $\mathcal{D}$. Therefore, if c_1 and c_2 are commands associated to two sequences γ_1 and γ_2, then proving that γ is well-defined comes down to showing that $[\,[[c_1]]\,]_{\mathcal{D}} = [\,[[c_2]]\,]_{\mathcal{D}}$. This equality follows from $c_1 =_\mu c_2$, which in turn holds because $\Phi(c_1) = \Phi(c_2)$.

Conclusion

The correspondences exhibited in this paper are

$$F(\mathcal{C})(X) \Leftrightarrow \mathtt{VT}_{\mathcal{C}}(X)/_{\alpha}, \quad \mathtt{VT}_{\mathcal{C}}(X)/_{\alpha} \Leftrightarrow \mu\mathtt{Comm}_{\mathcal{C}}(X)/_{=_\mu} \quad \text{and} \quad \mathtt{VT}_{\mathcal{C}}(X)/_{\alpha} \Leftrightarrow \mu\mathtt{Comm}^{nf}_{\mathcal{C}}(X)/_{='}.$$

The first one links us with the literature: Vernon trees are a handy "in-lined" notation for trees-with-half-edges-whose-vertices-are-decorated-by-operadic-operations, and it eases the proof that the latter have, on one hand, the structure of a monad, and, on the other hand, the structure of a biased cyclic operad. The second correspondence gives the representation of Vernon trees in the μ-syntax formalism and it is the natural context for arriving at the last correspondence, which was our formal tool for establishing Theorem 1.

In future work, we hope to apply a similar syntactic approach to other variations of operads (like modular or wheeled operads), and wish to investigate the adjustments to be made in the case where symmetries (other than cyclic permutations) are not present.

References

[HC00] P. -L. Curien, H. Herbelin, The duality of computation, ACM SIGPLAN Notices, Volume 35 Issue 9, 233-243, September 2000.

[GK95] E. Getzler, M. Kapranov, Cyclic operads and cyclic homology, Geom., Top., and Phys. for Raoul Bott, International Press, Cambridge, MA, 167-201, 1995.

[KM94] M. Kontsevich, Y. Manin. Gromov-Witten Classes, Quantum Cohomology, and Enumerative Geometry. Comm. Math. Phys., 164:525-562, February 1994. Preprint, hep-th/9402147.

[L07] F. Lamarche, On the algebra of structural contexts, preprint available from `http://www.loria.fr/~lamarche`.

[M96] M. Markl, Models for operads, Comm. Algebra, 24(4):14711500, 1996.

[M14] M. Markl, Modular envelopes, OSFT and nonsymmetric (non-Σ) modular operads, arXiv:1410.3414.

[M06] M. Markl, Operads and PROPs, arXiv:math/0601129.

[M72] J. P. May, The geometry of iterated loop spaces, volume 271 of Lectures Notes in Mathematics. Springer-Verlag, Berlin, 1972.

[L10] A. Lukács, Cyclic operads, Dendroidal Structures, Higher Categories, PhD thesis, Utrecht University, 2010.

Appendix

Proof of Lemma 3. The termination of the system is obvious: in an arbitrary reduction sequence, each subsequent tree has one special corolla less, and the sequence finishes either when all of them are exhausted (in the case when the initial tree has at least one ordinary corolla), or when there is only one special corolla left (in the case when the initial tree consists only of special corollas). Due to the connectedness of Vernon trees, all special corollas (except one in the latter case) will indeed be exhausted. Clearly, the set of normal forms is $\mathtt{VT}_{\mathcal{C}}$.

Suppose that $\mathcal{T}_1$ and $\mathcal{T}_2$ are reduced from $\mathcal{T} \in \mathtt{eVT}_{\mathcal{C}}$ in one step, and let $\langle u_1, v_1\rangle$ and $\langle u_2, v_2\rangle$ be the pairs of corollas involved in the respective reductions, with v_1 and v_2 being special corollas. We prove local confluence by case analysis, with respect to whether v_1 and v_2 are equal or not.

Let $v_1 = v_2 = (x, y)$. Notice that, if $x, y \in FV(\mathcal{T})$, then (x, y) is the only corolla of $\mathcal{T}$, i.e. $\mathcal{T}$ is already in normal form. Also, if $\sigma(x) = x$, $\sigma(y) = x_i$ (or $\sigma(y) = y$, $\sigma(x) = x_i$) and $x_i \in FV(f)$, then $u_1 = u_2 = f(\dots, x_i, \dots)$ and $\mathcal{T}_1$ and $\mathcal{T}_2$ are trivially equal. Let us therefore assume that $x, y \notin FV(\mathcal{T})$. Let $\sigma(x) = x_i$ and $\sigma(y) = y_j$, where x_i and y_j come from u_1 and u_2, respectively. Since extended Vernon trees contain no cycles, u_1 and u_2 must be different. We proceed by analysing the shapes of u_1 and u_2.

i) If $u_1 = f(x_1, \dots, x_i, \dots, x_n)$ and $u_2 = g(y_1, \dots, y_j, \dots, y_m)$, i.e. if

$$\mathcal{T} = (f(x_1, \dots, x_i, \dots, x_n), (x, y), g(y_1, \dots, y_j, \dots, y_m), \dots; \sigma),$$

then both reductions arise from the first equation, leading to

$$\mathcal{T}_1 = (f^{\tau_1}(x_1, \dots, x_{i-1}, y, x_{i+1}, \dots, x_n), g(y_1, \dots, y_j, \dots, y_m), \dots; \sigma'_1),$$

where $\sigma'_1(y) = y_j$, on one hand, and

$$\mathcal{T}_2 = (f(x_1, \dots, x_i, \dots, x_n), g^{\tau_2}(y_1, \dots, y_{j-1}, x, y_{j+1}, y_m), \dots; \sigma'_2),$$

where $\sigma'_2(x) = x_i$, on the other hand. $\mathcal{T}_1$ and $\mathcal{T}_2$ are clearly α-equivalent with

$$\mathcal{T}_3 = (f^{\tau_1}(x_1, \dots, x_{i-1}, y, x_{i+1}, \dots, x_n), g^{\tau_2}(y_1, \dots, y_{j-1}, x, y_{j+1}, y_m), \dots; \sigma'_3),$$

where $\sigma'_3 = \sigma'_1 = \sigma'_2$ on $Flag(\mathcal{T}_1)\backslash\{y, y_j\} = Flag(\mathcal{T}_2)\backslash\{x, x_i\}$ and $\sigma'_3(x) = y$. Therefore, $\mathcal{T}_1 =_\alpha \mathcal{T}_2$.

ii) Suppose now that u_1 is like above and that u_2 is a special corolla (y_j, z_j), i.e. that

$$\mathcal{T} = (f(x_1, \dots, x_i, \dots, x_n), (x, y), (y_j, z_j), \dots; \sigma).$$

In this case, by the reduction coming from the first equation, we get

$$\mathcal{T}_1 = (f^{\tau}(x_1, \dots, x_{i-1}, y, x_{i+1}, \dots, x_n), (y_j, z_j), \dots; \sigma'_1),$$

where $\sigma'_1(y) = y_j$, and, by the other one,

$$\mathcal{T}_2 = (f(x_1, \dots, x_i, \dots, x_n), (x, z_j), \dots; \sigma'_2),$$

where $\sigma'_2(x) = x_i$. $\mathcal{T}_1$ and $\mathcal{T}_2$ can be reduced again, the respective reductions leading to

$$\mathcal{T}'_1 = ((f^{\tau})^{\tau_1}(x_1, \dots, x_{i-1}, z_j, x_{i+1}, \dots, x_n), \dots; \sigma''_1),$$

and

$$\mathcal{T}'_2 = (f^{\tau_2}(x_1, \dots, x_{i-1}, z_j, x_{i+1}, \dots, x_n), \dots; \sigma''_2).$$

It is easy to verify that $\tau_1\tau = \tau_2$ and $\sigma''_1 = \sigma''_2$, from which we conclude that $\mathcal{T}'_1 = \mathcal{T}'_2$.

iii) If both u_1 and u_2 are special corollas, say (w_i, x_i) and (y_j, z_j) respectively, i.e. if

$$\mathcal{T} = ((w_i, x_i), (x, y), (y_j, z_j), \dots; \sigma),$$

then we get

$$\mathcal{T}_1 = ((w_i, y), (y_j, z_j), \dots; \sigma'_1),$$

with $\sigma_1(y) = y_j$, and

$$\mathcal{T}_2 = ((w_i, x_i), (x, z_j), \dots; \sigma'_2),$$

with $\sigma'_2(x) = x_i$. The conclusion follows since, by the reduction arising from the second equation, both $\mathcal{T}_1$ and $\mathcal{T}_2$ can now be reduced to the tree

$$\mathcal{T}_3 = ((w_i, z_j), \dots; \sigma'_3).$$

On the other hand, if $v_1 = (a, b)$ and $v_2 = (c, d)$, we proceed by comparing u_1 and u_2.

iv) If $u_1 = u_2 = f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n)$, and if $\sigma(a) = x_i$ and $\sigma(c) = x_j$, then the corresponding reductions of

$$\mathcal{T} = (f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n), (a, b), (c, d), \ldots; \sigma)$$

lead to

$$\mathcal{T}_1 = (f^{\tau_1}(x_1, \ldots, b, \ldots, x_j, \ldots, x_n), (c, d), \ldots; \sigma_1)$$

and

$$\mathcal{T}_2 = (f^{\tau_2}(x_1, \ldots, x_i, \ldots, d, \ldots, x_n), (a, b), \ldots; \sigma_2),$$

where $\sigma_1(c) = x_j$ and $\sigma_2(a) = x_i$. This configuration of $\mathcal{T}_1$ and $\mathcal{T}_2$ is analogous to the one from *ii)* and the conclusion follows by the same argument.

v) If $u_1 = u_2 = (x, y)$, $\sigma(a) = x$ and $\sigma(c) = y$, the reasoning is the same as in *iii)*.

vi) Finally, if $u_1 \neq u_2$, then $\mathcal{T}_1$ arises by reducing $\langle u_1, v_1 \rangle$, while u_2 and v_2 remain unchanged, and, symmetrically, $\mathcal{T}_2$ arises by reducing $\langle u_2, v_2 \rangle$, while u_1 and v_1 remain unchanged. By reducing $\langle u_2, v_2 \rangle$ in $\mathcal{T}_1$ and $\langle u_1, v_1 \rangle$ in $\mathcal{T}_2$, we clearly obtain the same tree.

All the cases being covered, the local confluence of the system is established.

Proof of Lemma 4. Suppose that $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n, s_1, \ldots, s_m; \sigma\}$, and that, for some $j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m\}$, $\mathcal{T}_j = \{f(x_1, \ldots, x_i, \ldots, x_n), \ldots; \sigma_j\}$ and $s_k = (y, z)$, and let $\sigma(x_i) = y$. Let $\mathcal{T}'$ be a tree obtained from $\mathcal{T}$ by a reduction with respect to $\mathcal{T}_j$ and s_k, i.e.

$$\mathcal{T}' = \{\ldots, \{f^{\tau}(x_1, \ldots, z, \ldots, x_n), \ldots; \sigma'_j\}, \ldots; \sigma'\},$$

where $\sigma'_j = \sigma_j$ on $V(\mathcal{T}_j) \backslash \{x_i\}$ and $\sigma'_j(z) = z$, and $\sigma' = \sigma|_{V(\mathcal{T}_j) \backslash \{x_i, y\}}$.

Let $\mathcal{T}'_j = \{f^{\tau}(x_1, \ldots, z, \ldots, x_n), \ldots; \sigma'_j\}$. The flattenings of $\mathcal{T}$ and $\mathcal{T}'$ are then given as

$$\mathit{flat}(\mathcal{T}) = \{\ldots, f(x_1, \ldots, x_i, \ldots, x_n), \ldots, (y, z), \ldots; \underline{\sigma}\},$$

where

$$\underline{\sigma}(x) = \begin{cases} \sigma(x) & \text{if } x \in \bigcup_{i=1}^{n} FV(\mathcal{T}_i) \cup \bigcup_{j=1}^{m} FV(s_j) \\ \sigma_i(x) & \text{if } x \in V(\mathcal{T}_i) \backslash FV(\mathcal{T}_i) \end{cases},$$

and

$$\mathit{flat}(\mathcal{T}') = \{\ldots, f^{\tau}(x_1, \ldots, z, \ldots, x_n), \ldots; \underline{\sigma'}\},$$

where

$$\underline{\sigma'} = \begin{cases} \sigma'(x) & \text{if } x \in \bigcup_{\substack{i=1 \\ i \neq j}}^{n} FV(\mathcal{T}_i) \cup FV(\mathcal{T}'_j) \cup \bigcup_{\substack{j=1 \\ j \neq k}}^{m} FV(s_j) \\ \sigma_i(x) & \text{if } x \in V(\mathcal{T}_i) \backslash FV(\mathcal{T}_i), \ i \neq j \\ \sigma'_j(x) & \text{if } x \in V(\mathcal{T}_j) \backslash FV(\mathcal{T}_j) \end{cases}$$

Now, since $x_i \in FV(\mathcal{T}_j)$, for $\mathit{flat}(\mathcal{T})$ we have $\underline{\sigma}(x_i) = \sigma(x_i) = y$. Therefore,

$$\mathit{flat}(\mathcal{T}) \to \{\{f^{\tau}(x_1, \ldots, z, \ldots, x_n), \ldots; \underline{\sigma}'\},$$

where

$$\underline{\sigma}' = \underline{\sigma}|_{X \backslash \{x_i, y\}} = \begin{cases} \sigma(x) & \text{if } x \in \bigcup_{i=1}^{n} FV(\mathcal{T}_i) \backslash \{x_i, y\} \cup \bigcup_{\substack{j=1 \\ j \neq k}}^{m} FV(s_j) \\ \sigma_i(x) & \text{if } x \in V(\mathcal{T}_i) \backslash FV(\mathcal{T}_i) \end{cases}.$$

That $\underline{\sigma'}$ and $\underline{\sigma}'$ are equal, i.e. that $\mathit{flat}(\mathcal{T}) \to \mathit{flat}(\mathcal{T}')$, follows easily.

The same kind of analysis proves the claim for the cases when the corolla of $\mathcal{T}_j$ involved in reduction is special, and when the reduction $\mathcal{T} \to \mathcal{T}'$ is done with respect to two special corollas.

Proof of Lemma 5. Denote

$$\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_j, \ldots, \mathcal{T}_n, s_1, \ldots, s_m; \sigma\} \text{ and } \mathcal{T}' = \{\mathcal{T}_1, \ldots, \mathcal{T}_j', \ldots, \mathcal{T}_n, s_1, \ldots, s_m; \sigma\},$$

and suppose that $\mathcal{T}_j = \{\ldots, f(x_1, \ldots, x_i, \ldots, x_n), \ldots, (y, z), \ldots; \sigma_j\}$, where $\sigma_j(x_i) = y$. Let $\mathcal{T}_j'$ be a tree obtained from $\mathcal{T}_j$ by a reduction with respect to $f(x_1, \ldots, x_i, \ldots, x_n)$ and (y, z), i.e.

$$\mathcal{T}_j' = \{\ldots, f^\tau(x_1, \ldots, z, \ldots, x_n), \ldots; \sigma_j'\},$$

where $\sigma_j' = \sigma_j|_{V(\mathcal{T}_j)\setminus\{x_i, y\}}$.

By the definition of flattening, we have

$$\mathit{flat}(\mathcal{T}) = \{\ldots, f(x_1, \ldots, x_i, \ldots, x_n), \ldots, (y, z), \ldots; \underline{\sigma}\},$$

where

$$\underline{\sigma}(x) = \begin{cases} \sigma(x) & \text{if } x \in \bigcup_{i=1}^n FV(\mathcal{T}_i) \cup \bigcup_{k=1}^m FV(s_k) \\ \sigma_i(x) & \text{if } x \in V(\mathcal{T}_i)\setminus FV(\mathcal{T}_i) \end{cases}$$

with σ_i, $1 \leq i \leq n$, being the involutions corresponding to $\mathcal{T}_i$. Now, since $\sigma_j(x_i) = y$, it follows that $x_i, y \in V(\mathcal{T}_j)\setminus FV(\mathcal{T}_j)$, and, consequently, that $\underline{\sigma}(x_i) = \sigma_j(x_i) = y$. Therefore,

$$\mathit{flat}(\mathcal{T}) \to \{\ldots, f^\tau(x_1, \ldots, z, \ldots, x_n), \ldots; \underline{\sigma}'\},$$

where $\underline{\sigma}' = \underline{\sigma}|_{V(\mathit{flat}(\mathcal{T}))\setminus\{x_i, y\}}$.

On the other hand, we have

$$\mathit{flat}(\mathcal{T}') = \{\ldots, f^\tau(x_1, \ldots, z, \ldots, x_n), \ldots; \underset{\sim}{\sigma}\},$$

where

$$\underset{\sim}{\sigma}(x) = \begin{cases} \sigma(x) & \text{if } x \in \bigcup_{\substack{i=1 \\ i\neq j}}^n FV(\mathcal{T}_i) \cup FV(\mathcal{T}_j') \cup \bigcup_{k=1}^m FV(s_k) \\ \sigma_i(x) & \text{if } x \in V(\mathcal{T}_i)\setminus FV(\mathcal{T}_i),\ \ i \neq j \\ \sigma_j'(x) & \text{if } x \in V(\mathcal{T}_j')\setminus FV(\mathcal{T}_j') \end{cases}$$

Since $FV(\mathcal{T}_j') = FV(\mathcal{T}_j)$ and $\sigma_j' = \sigma_j$ on $V(\mathcal{T}_j)\setminus\{x_i, y\} = V(\mathcal{T}_j') \supseteq V(\mathcal{T}_j')\setminus FV(\mathcal{T}_j')$, it follows that $\underline{\sigma}' = \underset{\sim}{\sigma}$, i.e. that $\mathit{flat}(\mathcal{T}) \to \mathit{flat}(\mathcal{T}')$.

The proof goes analogously for the case when the reduction $\mathcal{T}_j \to \mathcal{T}_j{}'$ is done with respect to two special corollas of $\mathcal{T}_j$.

Proof of Lemma 8. It is straightforward to check that the partial composition is well defined. In order to illustrate that it also satisties the cyclic operad laws, we now show that (U1) holds. We verify the equality $[\mathcal{T}]_\alpha \,{}_x\!\bullet_y [\{(y, z); id\}]_\alpha = [\mathcal{T}]_\alpha^\kappa$, where $FV(\mathcal{T}) = X$ and κ is the renaming of x to z, by case analysis with respect to whether y and z appear in $V(\mathcal{T})$. We handle each case by choosing the renamings ϑ_1 and ϑ_2 of $(V(\mathcal{T})\setminus X) \cup \{x\}$ and y, respectively, that are *closest to the identity*.

Suppose, say, that $y \in V(\mathcal{T})$ and $z \in V(\mathcal{T})$. Then, since $(X\setminus\{x\}) \cap (\{y, z\}\setminus\{y\}) = \emptyset$, it must be the case that $z \in (V(\mathcal{T})\setminus X) \cup \{x\}$, and we take $\vartheta_1 : (((V(\mathcal{T})\setminus X) \cup \{x\})\setminus\{z\}) \cup \{w\} \to (V(\mathcal{T})\setminus X) \cup \{x\}$ to be identity everywhere, except on w, which is mapped to z, and $\vartheta_2 : \{u\} \to \{y\}$. Let $x \in FV(C_x)$, $y \in FV(C_y)$ and $z \in FV(C_z)$. Suppose also that C_x, C_y and C_z are mutually different corollas.

By the definition of ${}_x\!\bullet_y$, we have that $[\mathcal{T}]_\alpha \,{}_x\!\bullet_y [\{(y, z); id\}]_\alpha = [nf(\mathcal{T}')]_\alpha$, where

$$Cor(\mathcal{T}') \quad = \quad \{C_z^{\vartheta_1}\} \cup Cor(\mathcal{T})\setminus\{C_z\} \cup \{(u, z)\}$$

and

$$\sigma_{\mathcal{T}'}(v) = \begin{cases} \sigma(v) & \text{if } v \in V(\mathcal{T})\setminus\{x, w\} \\ \sigma(z) & \text{if } v = w \\ u & \text{if } v = x \\ x & \text{if } v = u \\ v & \text{if } v = z \end{cases}$$

with σ being the involution of $\mathcal{T}$. Clearly, $nf(\mathcal{T}')$ is obtained by the reduction with respect to (u, z) and C_x. Therefore,

$$Cor(nf(\mathcal{T}')) = \{C_z^{\vartheta_1}\} \cup Cor(\mathcal{T})\backslash\{C_z, C_x\} \cup \{C_x^{\tau}\},$$

where τ is renames x to z, and the involution $\sigma'_{\mathcal{T}'}$ of $nf(\mathcal{T}')$ is defined as

$$\sigma'_{\mathcal{T}'}(v) = \begin{cases} \sigma(v) & \text{if } v \in V(\mathcal{T})\backslash\{x, w\} \\ \sigma(z) & \text{if } v = w \\ v & \text{if } v = z \end{cases}$$

Our goal is to show that $nf(\mathcal{T}') =_\alpha \mathcal{T}^{\kappa+\varepsilon}$, for an arbitrary bijection $\varepsilon : V \to V(\mathcal{T})\backslash X$ such that $V \cap X' = \emptyset$. We choose ε to be the restriction of ϑ_1 on $(V(\mathcal{T}_1)\backslash(X \cup \{z\})) \cup \{w\}$. Therefore, ε is simply the renaming of z to w. For the tree $\mathcal{T}^{\kappa+\varepsilon}$ we now have

$$Cor(\mathcal{T}^{\kappa+\varepsilon}) = (Cor(\mathcal{T})\backslash\{C_x, C_z\}) \cup \{C_x^{\kappa+\varepsilon}, C_z^{\kappa+\varepsilon}\},$$

and the involution $\sigma^{\kappa+\varepsilon}$ of $\mathcal{T}^{\kappa+\varepsilon}$ is defined as

$$\begin{aligned} \sigma^{\kappa+\varepsilon}(v) &= \begin{cases} \sigma(\varepsilon(v)) & \text{if } v \in V(\mathcal{T}^{\kappa+\varepsilon})\backslash X' \\ v & \text{if } v \in X' \end{cases} \\ &= \begin{cases} \sigma(v) & \text{if } v \in V(\mathcal{T}^{\kappa+\varepsilon})\backslash(((X\backslash\{x\}) \cup \{z\}) \cup \{w\}) \\ \sigma(z) & \text{if } v = w \\ v & \text{if } v \in (X\backslash\{x\}) \cup \{z\} \end{cases} \\ &= \begin{cases} \sigma(v) & \text{if } v \in V(\mathcal{T}^{\kappa+\varepsilon})\backslash\{x, z, w\} \\ \sigma(z) & \text{if } v = w \\ v & \text{if } v = z \end{cases} \end{aligned}$$

Clearly, $C_x^{\tau} = C_x^{\kappa+\varepsilon}$ and $C_z^{\vartheta_1} = C_z^{\kappa+\varepsilon}$, and for the respective involutions, since $V(\mathcal{T}^{\kappa+\varepsilon})\backslash\{x, z, w\} = V(\mathcal{T})\backslash\{x, w\}$, we have $\sigma'_{\mathcal{T}'} = \sigma^{\kappa+\varepsilon}$, which completes the claim.

As for the rest of the axioms from Definition 1, the verifications are lengthy but straightforward, and are done in a similar fashion.

Proof of Lemma 9. The proof goes by structural induction on t and c.
a) If $t \equiv x$ and $\sigma(z) = x$, then

$$[[x^\sigma]]_y = [[z]]_y = id_{z,y} = {id_{x,y}}^{\sigma_y} = [[x]]_y^{\sigma_y},$$

where the third equality holds thanks to the axiom (U3).

Suppose now that the claim holds for $c : X \cup \{x\}$ and let $t \equiv \mu x.c$. Then we have

$$[[(\mu x.c)^\sigma]]_y = [[\mu x.(c^{\sigma_x})]]_y = [[c^{\sigma_x}[y/x]]] = [[(c^{\sigma_x})^{\tau_1}]] = [[c]]^{\tau_1\sigma_x},$$

where $\tau_1 = {id_{X'\cup\{x\}}}^{y/x}$, on one hand, and, on the other,

$$[[\mu x.c]]_y^{\sigma_y} = [[c[y/x]]]^{\sigma_y} = [[c^{\tau_2}]]^{\sigma_y} = ([[c]]^{\tau_2})^{\sigma_y} = [[c]]^{\sigma_y\tau_2},$$

where $\tau_2 = {id_{X\cup\{x\}}}^{y/x}$. The claim follows from the equality of composites $\tau_1\sigma_x$ and $\sigma_y\tau_2$.
b) Let $c \equiv \underline{f}\{t_z \mid z \in Z\} : \bigcup_{z\in Z} Y_z = X$ and suppose that the claim holds for all terms $Y_z \mid t_z$. Then

$$[[\underline{f}\{t_z \mid z \in Z\}^\sigma]] = [[\underline{f}\{t_z^{\sigma|^{Y_z}} \mid z \in Z\}]] = f(\varphi),$$

where $\varphi : z \mapsto ([[t_z^{\sigma|^{Y_z}}]]_{\overline{z}}, \overline{z})$, and

$$[[\underline{f}\{t_z \mid z \in Z\}]]^\sigma = f(\varphi')^\sigma,$$

where $\varphi' : z \mapsto ([[t_z]]_{\overline{z}}, \overline{z})$. The conclusion follows from the sequence of equalities

$$f(\varphi')^\sigma = f(\varphi'_\sigma) = f(\varphi),$$

that holds thanks to the Lemma 1 and the induction hypothesis:

$$([[t_z^{\sigma|^{Y_z}}]]_{\overline{z}}, \overline{z}) = ([[t_z]]_{\overline{z}}^{(\sigma|^{Y_z})_{\overline{z}}}, \overline{z}), \text{ for each } z \in Z.$$

Finally, if $c \equiv \langle t_1 \,|\, t_2 \rangle : X_1 \cup X_2 = X$, where $X_1 \,|\, t_1$ and $X_2 \,|\, t_2$ satisfy the claim, then

$$[[\langle t_1 \,|\, t_2 \rangle^\sigma]] = [[\langle t_1^{\sigma|^{X_1}} \,|\, t_2^{\sigma|^{X_2}} \rangle]] = [[t_1^{\sigma|^{X_1}}]]_u \; {}_u\circ_v \, [[t_2^{\sigma|^{X_2}}]]_v = [[t_1]]_u^{(\sigma|^{X_1})_u} \; {}_u\circ_v \, [[t_2]]_v^{(\sigma|^{X_2})_v},$$

and

$$[[\langle t_1 \,|\, t_2 \rangle]]^\sigma = ([[t_1]]_u \; {}_u\circ_v \, [[t_2]]_v)^\sigma,$$

the two combinators on the right-hand sides being equal thanks to the axiom (EQ).

Proof of Lemma 10. First of all, if t is a variable, say y, then

$$[[s[y/x]]]_u = [[s^{id^{y/x}}]]_u = [[s]]_u^{\; id^{y/x}} = [[s]]_u \; {}_x\circ_v \, id_{v,y} = [[s]]_u \; {}_x\circ_v \, [[y]]_v \,,$$

and, analogously,

$$[[c[y/x]]] = [[c^{id^{y/x}}]] = [[c]]^{id^{y/x}} = [[c]] \; {}_x\circ_z \, id_{z,y} = [[c]] \; {}_x\circ_z \, [[y]]_z.$$

Otherwise, i.e. if $t \equiv \mu y.c_1$, we proceed by induction on the structure of s, i.e. c.
a) Suppose first that $s \equiv x$. Then

$$\begin{array}{rclcrcl} [[x[\mu y.c_1/x]]]_u & = & [[\mu y.c_1]]_u & = & [[c_1[u/y]]] & = & [[c_1^{id^{u/y}}]] \\ & = & [[c_1]]^{id^{u/y}} & = & [[c_1]] \; {}_y\circ_x \, id_{x,u} & = & [[c_1]] \; {}_y\circ_x \, [[x]]_u \quad = \quad [[c_1[u/y]]] \; {}_u\circ_x \, [[x]]_u. \end{array}$$

Next, assume that $c : X \cup \{z\}$ satisfies the claim and let $s \equiv \mu z.c$. We have

$$\begin{array}{rclcrcl} [[\mu z.c[\mu y.c_1/x]]]_u & = & [[\mu z.(c[\mu y.c_1/x])]]_u & = & [[c[\mu y.c_1/x][u/z]]] & = & [[c[\mu y.c_1/x]^{id^{u/z}}]] \\ & = & [[c[\mu y.c_1/x]]]^{id^{u/z}} & = & ([[c]] \; {}_x\circ_y \, [[c_1]])^{id^{u/z}} & = & [[c]]^{id^{u/z}} \; {}_x\circ_y \, [[c_1]] \\ & = & [[c[u/z]]] \; {}_x\circ_v \, [[c_1[v/y]]] & = & [[\mu v.c]]_u \; {}_x\circ_v \, [[c_1[v/y]]]. & & \end{array}$$

b) Let $X = X_1 \cup X_2$ and suppose $c \equiv \langle t_1 \,|\, t_2 \rangle$, where $X_1 \,|\, t_1$ and $X_2 \,|\, t_2$ satisfy the claim. Without loss of generality, we can assume that $x \in X_2$. Then we have

$$\begin{array}{rclcrcl} [[\langle t_1 \,|\, t_2 \rangle[\mu y.c_1/x]]] & = & [[\langle t_1 \,|\, t_2[\mu y.c_1/x] \rangle]] & & = & [[t_1]]_a \; {}_a\circ_b \, [[t_2[\mu y.c_1/x]]]_b \\ & = & [[t_1]]_a \; {}_a\circ_b \, ([[t_2]]_b \; {}_x\circ_v \, [[\mu y.c_1]]_v) & = & ([[t_1]]_a \; {}_a\circ_b \, [[t_2]]_b) \; {}_x\circ_v \, [[\mu y.c_1]]_v \\ & = & [[\langle t_1 \,|\, t_2 \rangle]] \; {}_x\circ_v \, [[\mu y.c_1]]_v. & & & & \end{array}$$

Finally, let $X = \bigcup_{z \in Z} Y_z$ and suppose that $c \equiv \underline{f}\{t_z \,|\, z \in Z\}$, where for all $z \in Z$, $Y_z \,|\, t_z$ satisfy the claim. Suppose, moreover, that for $a \in Z$, $x \in Y_a$. Then, on one hand, we have

$$[[\underline{f}\{t_z \,|\, z \in Z\}[\mu y.c_1/x]]] = [[\underline{f}\{\{t_z \,|\, z \in Z \backslash \{a\}\} \cup \{t_a[\mu y.c_1/x]\}\}]] = f(\varphi),$$

where $\varphi : z \mapsto ([[t_z]]_{\overline{z}}, \overline{z})$, for all $z \in Z \backslash \{a\}$, and $\varphi : a \mapsto ([[t_a[\mu y.c_1/x]]]_{\overline{a}}, \overline{a})$. On the other hand,

$$[[\underline{f}\{t_z \,|\, z \in Z\}]] \; {}_x\circ_v \, [[\mu y.c_1]]_v = f(\psi_1) \; {}_x\circ_v \, [[\mu y.c_1]]_v,$$

where $\psi_1 : z \mapsto ([[t_z]]_{\overline{z}}, \overline{z})$, for all $z \in Z$. By Lemma 1,

$$f(\psi_1) \; {}_x\circ_v \, [[\mu y.c_1]]_v = f(\psi_2),$$

where $\psi_2 = \psi_1$ on $Z \backslash \{a\}$, and $\psi_2 : a \mapsto ([[t_a]]_{\overline{a}} \; {}_x\circ_v \, [[\mu y.c_1]]_v, \overline{a})$. Hence, we need to prove that

$$[[t_a[\mu y.c_1/x]]]_{\overline{a}} = [[t_a]]_{\overline{a}} \; {}_x\circ_v \, [[\mu y.c_1]]_v,$$

but this equality is exactly the induction hypothesis for the term t_a.