Parsing as a lifting problem and the Chomsky-Schützenberger representation theorem

Paul-André Melliès, Noam Zeilberger

Submitted to MFPS 22

Parsing as a lifting problem and the Chomsky-Schützenberger representation theorem

Paul-André Melliès[1]

IRIF, Université Paris Cité, CNRS, Inria, Paris, France

Noam Zeilberger[2]

LIX, École Polytechnique, Palaiseau, France

Abstract

Building on our work on type refinement systems, we continue developing the thesis that many kinds of deductive systems may be usefully modelled as functors and derivability as a lifting problem, focusing in this work on derivability in context-free grammars. We begin by explaining how derivations in any context-free grammar may be naturally encoded by a functor of operads from a freely generated operad into a certain "operad of spliced words". This motivates the introduction of a more general notion of context-free grammar over any category, defined as a finite species $\mathcal{S}$ equipped with a color denoting the start symbol and a functor of operads $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[C]$ into the *operad of spliced arrows* in C, generating a context-free language of arrows. We show that many standard properties of context-free grammars can be formulated within this framework, thereby admitting simpler analysis, and that usual closure properties of context-free languages generalize to context-free languages of arrows. One advantage of considering parsing as a lifting problem is that it enables a dual fibrational perspective on the functor p via the notion of *displayed operad*, corresponding to a lax functor of operads $\mathcal{W}[C] \to \mathrm{Span}(\mathrm{Set})$. We show that displayed free operads admit an explicit inductive definition, using this to give a reconstruction of Leermakers' generalization of the CYK parsing algorithm. We then turn to the Chomsky-Schützenberger Representation Theorem. We start by explaining that a non-deterministic finite state automaton over words, or more generally over arrows of a category, can be seen as a category Q equipped with a pair of objects denoting initial and accepting states and a functor of categories $Q \to C$ satisfying the unique lifting of factorizations (ULF) property and the finite fiber property, recognizing a regular language of arrows. Then, we explain how to extend this notion of automaton to functors of operads, which generalize tree automata, allowing us to lift an automaton over a category to an automaton over its operad of spliced arrows. We show that every context-free grammar over a category can be pulled back along a non-deterministic finite state automaton over the same category, and hence that context-free languages are closed under intersection with regular languages. The last and important ingredient is the identification of a left adjoint $C[-] : \mathrm{Operad} \to \mathrm{Cat}$ to the operad of spliced arrows functor $\mathcal{W}[-] : \mathrm{Cat} \to \mathrm{Operad}$. This construction builds the contour category $C[O]$ of any operad O, whose arrows have a geometric interpretation as "oriented contours" of operations. A direct consequence of the contour / splicing adjunction is that every pointed finite species induces a universal context-free grammar, generating a language of tree contour words. Finally, we prove a generalization of the Chomsky-Schützenberger Representation Theorem, establishing that any context-free language of arrows over a category C is the functorial image of the intersection of a C-chromatic tree contour language and a regular language.

Keywords: context-free languages, parsing, finite state automata, category theory, operads, representation theorem

1 Introduction

In "Functors are Type Refinement Systems" [25], we argued for the idea that rather than being modelled merely as categories, type systems should be modelled as functors $p : \mathcal{D} \to \mathcal{T}$ from a category $\mathcal{D}$ whose morphisms are

[1] Email: paul-andre.mellies@cnrs.fr. Partially supported by ANR ReciProg (ANR-21-CE48-0019).
[2] Email: noam.zeilberger@lix.polytechnique.fr. Partially supported by ANR LambdaComb (ANR-21-CE48-0017).

typing derivations to a category $\mathcal{T}$ whose morphisms are the terms corresponding to the underlying *subjects* of those derivations. One advantage of this fibrational point of view is that the notion of typing judgment receives a simple mathematical status, as a triple (R, f, S) consisting of two objects R, S in $\mathcal{D}$ and a morphism f in $\mathcal{T}$ such that $p(R) = \mathrm{dom}(f)$ and $p(S) = \mathrm{cod}(f)$. The question of finding a typing derivation for a typing judgment (R, f, S) then reduces to the lifting problem of finding a morphism $\alpha : R \to S$ such that $p(\alpha) = f$. We developed this perspective in a series of papers [25,27,26], and believe that it may be usefully applied to a large variety of deductive systems, beyond type systems in the traditional sense. In this work, we focus on derivability in context-free grammars, a classic topic in formal language theory with wide applications in computer science.

To set the stage and motivate the overall approach, let us begin by quickly explaining how context-free grammars naturally give rise to certain functors of colored operads $\mathcal{D} \to \mathcal{T}$. We will assume that the reader is already familiar with context-free grammars and languages [30] as well as with operads or multicategories [23, Ch. 2]. Note that "multicategory" and "colored operad" are two different names in the literature for the same concept, and in this paper we will often just use the word operad, it being implicit that operads always carry a (potentially trivial) set of colors. We write $f \circ (g_1, \ldots, g_n)$ for parallel composition of operations in an operad, and $f \circ_i g$ for the partial composition of g into f after the first i inputs.

Classically, a context-free grammar is defined as a tuple $G = (\Sigma, N, S, P)$ consisting of a finite set Σ of terminal symbols, a finite set N of non-terminal symbols, a distinguished non-terminal $S \in N$ called the start symbol, and a finite set P of production rules of the form $R \to \sigma$ where $R \in N$ and $\sigma \in (N \cup \Sigma)^*$ is a string of terminal or non-terminal symbols. Observe that any sequence σ on the right-hand side of a production can be factored as $\sigma = w_0 R_1 w_1 \ldots R_n w_n$ where $w_0, \ldots, w_n$ are words of terminals and $R_1, \ldots, R_n$ are non-terminal symbols. We will use this simple observation in order to capture derivations in context-free grammars by functors of operads $\mathcal{D} \to \mathcal{T}$ from an operad $\mathcal{D}$ whose colors are non-terminals to a certain monochromatic operad $\mathcal{T} = \mathcal{W}[\Sigma]$ that we like to call the **operad of spliced words** in Σ. The n-ary operations of $\mathcal{W}[\Sigma]$ consist of sequences $w_0 - w_1 - \ldots - w_n$ of $n+1$ words in Σ^* separated by n *gaps* notated with the $-$ symbol, with composition defined simply by "splicing into the gaps" and interpreting juxtaposition by concatenation in Σ^*. For example, the parallel composition of the spliced word $a{-}b{-}c$ with the pair of spliced words $d{-}e{-}f$ and $\epsilon{-}a$ is defined as $(a{-}b{-}c) \circ (d{-}e{-}f, \epsilon{-}a) = ad{-}e{-}fb{-}ac$. The identity operation is given by the spliced word $\epsilon{-}\epsilon$, and it is routine to check that the operad axioms are satisfied.

Now, to any context-free grammar G we can associate a free operad $\mathcal{D}[G]$ that we call the (colored) **operad of derivations** in G. Its colors are the non-terminal symbols $R \in N$ of the grammar, while its operations are freely generated by the production rules, with each rule $r \in P$ of the form $R \to w_0 R_1 w_1 \ldots R_n w_n$ giving rise to an n-ary operation $r : R_1, \ldots, R_n \to R$. These basic operations freely generate the operad $\mathcal{D}[G]$ whose general operations $R_1, \ldots, R_n \to R$ can be regarded as (potentially incomplete) parse trees with root label R and free leaves labelled $R_1, \ldots, R_n$, and with each node labelled by a production rule of G. Moreover, this free operad comes equipped with an evident forgetful functor $\mathcal{D}[G] \to \mathcal{W}[\Sigma]$ that sends every non-terminal symbol R to the unique color of $\mathcal{W}[\Sigma]$, and every generating operation $r : R_1, \ldots, R_n \to R$ as above to the spliced word $w_0 - \ldots - w_n$, extending to parse trees homomorphically. See Fig. 1 for an illustration.

In the rest of the paper, we will see how this point of view may be generalized to define a notion of context-free language of arrows between two objects A and B in any category C, by first introducing a certain operad $\mathcal{W}[C]$ of *sliced arrows in* C. We will see that many standard concepts and properties of context-free grammars and languages can be formulated within this framework, thereby admitting simpler analysis, and that parsing may indeed be profitably considered from a fibrational perspective, as a lifting problem along a functor from a freely generated operad. We will also develop a notion of non-deterministic finite state automaton and regular language of arrows in a category, and show that context-free languages are closed under intersection with regular languages. Finally, we will establish a categorical generalization of the Chomsky-Schützenberger representation theorem, relying on a fundamental adjunction between categories and operads that we call the contour / splicing adjunction.

Related work.

The functorial perspective on context-free grammars that we just sketched and take as a starting point for this article (§2) is very similar to that of Walters in his brief "note on context-free languages" [32], with the main difference that we generalize it to context-free languages over any category by considering the operad of sliced arrows construction. It is also closely related to de Groote's treatment of CFGs in his paper introducing *abstract categorial grammars* [8] and in a later article with Pogodalla [9], which were developed within a λ-calculus framework rather than a categorical / operadic one. The contour category construction and the contour / splicing adjunction between

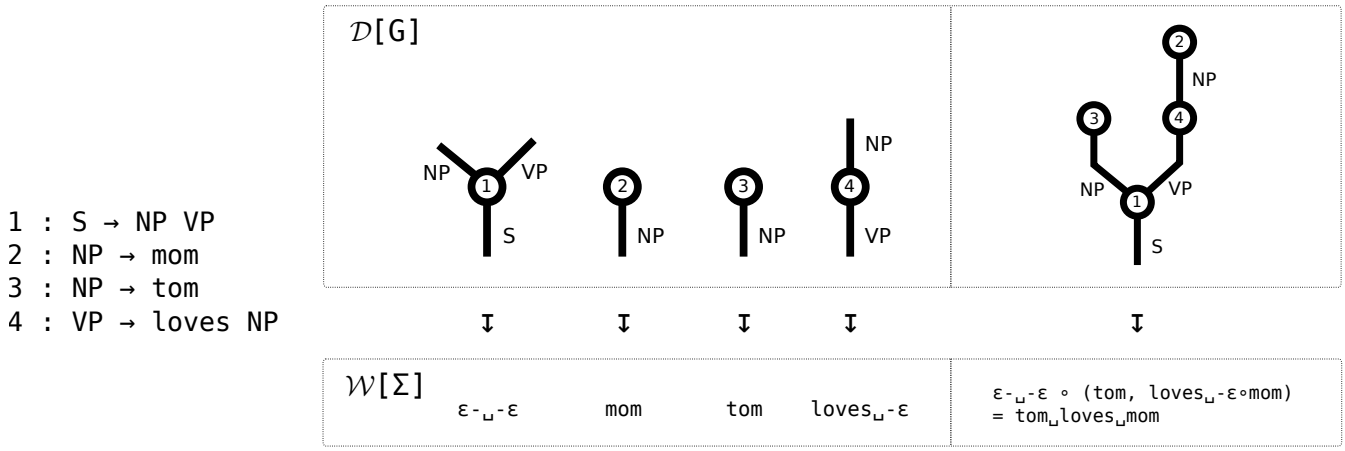


Fig. 1. Example of a context-free grammar and the corresponding functor $\mathcal{D}[G] \to \mathcal{W}[\Sigma]$, indicating the action of the functor on the generating operations of $\mathcal{D}[G]$ as well the induced action on a closed derivation.

operads and categories is fundamental to our treatment of the Chomsky-Schützenberger representation theorem (§4), and provides an unexpected geometric lens on context-free grammars, evocative of the geometry of interaction [13]. Although the adjunction is not identified, this geometric perspective is also apparent in Slavnov's recent work [31], inspired both by abstract categorial grammars and by proof-nets for classical linear logic [12], wherein he constructs a compact closed monoidal category of *word cobordisms* reminiscent of the operad of spliced words.

The fibrational perspective on non-deterministic finite state automata as finitary ULF functors that we take in the middle of this article (§3) is also similar in spirit to (and roughly dual to) Colcombet and Petrişan's proposal [6] for modelling various forms of automata as functors. Our approach is motivated by the desire to place both context-free grammars and non-deterministic finite state automata within a common framework, facilitating for example taking the intersection of a context-free language with a regular language. Our main goal is to develop a unified framework for type systems and other deductive systems, which would benefit from the classical body of work on context-free languages and automata theory, and in a future article we intend to consider parsing from left to right [17,10].

2 Context-free languages of arrows in a category

In this section we explain how the functorial formulation of context-free grammars discussed in the Introduction extends naturally to context-free grammars over any category, which at the same time leads to a simplification of the classical treatment of context-free languages while also providing a useful generalization. First, we need to explain how the operad $\mathcal{W}[\Sigma]$ of spliced words mentioned in the Introduction generalizes to define an operad $\mathcal{W}[C]$ of spliced arrows over any category C.

2.1 *The operad of spliced arrows of a category*

Definition 2.1 Let C be a category. The **operad $\mathcal{W}[C]$ of spliced arrows in C** is defined as follows:

- its colors are pairs (A, B) of objects of C;
- its n-ary operations $(A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ consist of sequences $w_0 - w_1 - \ldots - w_n$ of $n+1$ arrows in C separated by n gaps notated $-$, where each arrow must have type $w_i : B_i \to A_{i+1}$ for $0 \leq i \leq n$, under the convention that $B_0 = A$ and $A_{n+1} = B$;
- composition of spliced arrows is performed by "splicing into the gaps": formally, the partial composition $f \circ_i g$ of a spliced arrow $g = u_0 - \ldots - u_m$ into another spliced arrow $f = w_0 - \ldots - w_n$ is defined by substituting g for the ith occurrence of $-$ in f (starting from the left using 0-indexing) and interpreting juxtaposition by sequential composition in C (see Fig. 2 for an illustration);
- the identity operation on (A, B) is given by $id_A - id_B$.

It is routine to check that $\mathcal{W}[C]$ satisfies the associativity and neutrality axioms of an operad, these reducing to associativity and neutrality of composition of arrows in C. Indeed, the spliced arrows operad construction defines a

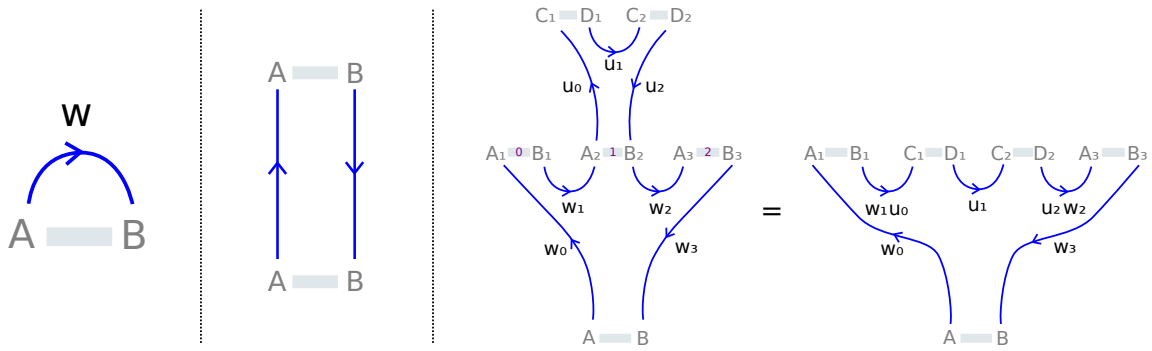


Fig. 2. Left: a constant of $\mathcal{W}[C]$. Middle: an identity operation. Right: illustration of partial composition. Here we compose an operation $g = u_0{-}u_1{-}u_2 : (C_1, D_1), (C_2, D_2) \to (A_2, B_2)$ into $f = w_0{-}w_1{-}w_2{-}w_3 : (A_1, B_1), (A_2, B_2), (A_3, B_3) \to (A, B)$ at the gap labelled 1 to obtain the operation $f \circ_1 g = w_0{-}w_1u_0{-}u_1{-}u_2w_2{-}w_3 : (A_1, B_1), (C_1, D_1), (C_2, D_2), (A_3, B_3) \to (A, B)$.

functor $\mathcal{W}[-] : \mathrm{Cat} \to \mathrm{Operad}$ since any functor of categories $F : C \to \mathcal{D}$ induces a functor of operads $\mathcal{W}[F] : \mathcal{W}[C] \to \mathcal{W}[\mathcal{D}]$, acting on colors by $(A, B) \mapsto (FA, FB)$ and on operations by $w_0{-}\ldots{-}w_n \mapsto Fw_0{-}\ldots{-}Fw_n$.

Example 2.2 Words $w \in \Sigma^*$ may be regarded as the arrows $w : * \to *$ of a one-object category that we notate $\mathcal{B}_\Sigma$, with sequential composition of arrows in $\mathcal{B}_\Sigma$ given by concatenation. The operad $\mathcal{W}[\Sigma]$ of spliced words in Σ described in the Introduction is identical to the operad $\mathcal{W}[\mathcal{B}_\Sigma]$ of spliced arrows in $\mathcal{B}_\Sigma$, and more generally, any monoid seen as a one-object category induces a corresponding operad of spliced words of that monoid.

Remark 2.3 Although the one-object category $\mathcal{B}_\Sigma$ is a free category (being freely generated by the arrows $a : * \to *$ ranging over letters $a \in \Sigma$), this property of being freely generated does not extend to its operad of spliced words. Indeed, an operad of spliced arrows $\mathcal{W}[C]$ is almost *never* a free operad. That's because any pair of objects A and B induces a binary operation $id_A{-}id_A{-}id_B : (A, A), (A, B) \to (A, B)$, and any arrow $w : A \to B$ of C induces a corresponding constant $w : (A, B)$. Since $id_A{-}id_A{-}id_B \circ (id_A, w) = w$, $\mathcal{W}[C]$ cannot be a free operad except in the trivial case where C has no objects and no arrows.

Example 2.4 The *ordinal sum* [20] of two categories C and $\mathcal{D}$ may be constructed as a category $C +_\sigma \mathcal{D}$ whose objects are the disjoint union of the objects of both categories, and whose arrows are the disjoint union of the arrows of both categories with an additional arrow $A \to B$ freely adjoined for every pair of objects $A \in C$, $B \in \mathcal{D}$. The operations of the spliced arrows operad $\mathcal{W}[C +_\sigma \mathcal{D}]$ may be described accordingly as consisting of a (possibly empty) sequence of arrows of C followed by a (possibly empty) sequence of arrows of $\mathcal{D}$. As a special case, consider spliced arrows over the ordinal sum $\mathcal{B}_\Sigma^\top = \mathcal{B}_\Sigma +_\sigma \mathbf{1}$, which is the two-object category obtained from $\mathcal{B}_\Sigma$ by freely adjoining an object $\top$ and an arrow $\$: * \to \top$. The operad $\mathcal{W}[\mathcal{B}_\Sigma^\top]$ includes operations of the form $f = w_0{-}\ldots{-}w_n\$: (*, *), \ldots, (*, *) \to (*, \top)$ which may be seen as spliced words with an explicit "end of input" marker, since it is impossible to concatenate anything after the last word w_n using only substitution in the operad. (See Example 2.8 below for an application of this construction.)

Remark 2.5 The operad $\mathcal{W}[\mathbf{1}]$ of spliced arrows over the terminal category is isomorphic to the terminal operad, with a single color $(*, *)$, and a single n-ary operation $id{-}\ldots{-}id : (*, *), \ldots, (*, *) \to (*, *)$ of every arity n. Likewise, the operad of spliced arrows over the product of two categories decomposes as a product of spliced arrow operads $\mathcal{W}[C \times \mathcal{D}] \cong \mathcal{W}[C] \times \mathcal{W}[\mathcal{D}]$. This might suggest that the functor $\mathcal{W}[-] : \mathrm{Cat} \to \mathrm{Operad}$ is a right adjoint, and we will see in §4.2 that this is indeed the case.

2.2 *Context-free grammars and context-free derivations over a category*

We already sketched in the Introduction how an ordinary context-free grammar $G = (\Sigma, N, P, S)$ gives rise to a freely generated operad $\mathcal{D}[G]$ equipped with a functor to the operad of spliced words $\mathcal{W}[\Sigma]$, where $\mathcal{D}[G]$ has the set of non-terminals N as objects and operations freely generated by the productions in P. To make this more precise and to generalize to context-free grammars over arbitrary categories, we first need to recall the notion of a (colored non-symmetric) species, and how one gives rise to a free operad.

A *colored non-symmetric species,* which we abbreviate to "species" [3] for short, is a tuple $\mathcal{S} = (C, V, i, o)$ con-

[3] Species in this sense are also called "multigraphs" [18] since they bear a precisely analogous relationship to multicategories

sisting of a span of sets $C^* \xleftarrow{i} V \xrightarrow{o} C$ with the following interpretation: C is a set of "colors", V is a set of "nodes", and the functions $i : V \to C^*$ and $o : V \to C$ return respectively the list of input colors and the unique output color of each node. Adopting the same notation as we use for operations of an operad, we write $x : R_1, \ldots, R_n \to R$ to indicate that $x \in V$ is a node with list of input colors $i(x) = (R_1, \ldots, R_n)$ and output color $o(x) = R$. However, it should be emphasized that a species by itself only contains bare coloring information about the nodes, and does not say how to compose them as operations.

We say that a species is *finite* (also called *polynomial* [16]) just in case both sets C and V are finite.

A map of species $\phi : \mathcal{S} \to \mathcal{R}$ from $\mathcal{S} = (C, V, i, o)$ to $\mathcal{R} = (D, W, i', o')$ is given by a pair $\phi = (\phi_C, \phi_V)$ of functions $\phi_C : C \to D$ and $\phi_V : V \to W$ making the diagram commute:

$$\begin{array}{ccccc} C^* & \xleftarrow{i} & V & \xrightarrow{o} & C \\ \downarrow{\scriptstyle \phi_C^*} & & \downarrow{\scriptstyle \phi_V} & & \downarrow{\scriptstyle \phi_C} \\ D^* & \xleftarrow{i'} & W & \xrightarrow{o'} & D \end{array}$$

Equivalently, overloading ϕ for both ϕ_C and ϕ_V, every node $x : R_1, \ldots, R_n \to R$ of $\mathcal{S}$ must be sent to a node $\phi(x) : \phi(R_1), \ldots, \phi(R_n) \to \phi(R)$ of $\mathcal{R}$. Every operad $\mathcal{O}$ has an underlying species with the same colors and whose nodes are the operations of $\mathcal{O}$, and this extends to a forgetful functor $\mathsf{Forget} : \mathrm{Operad} \to \mathrm{Species}$ from the category of operads and functors of operads to the category of species and maps of species. Moreover, this forgetful functor has a left adjoint

$$\mathrm{Species} \underset{\mathsf{Forget}}{\overset{\mathsf{Free}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathrm{Operad} \tag{1}$$

which sends any species $\mathcal{S}$ to an operad $\mathsf{Free}\,\mathcal{S}$ with the same set of colors and whose operations are freely generated from the nodes of $\mathcal{S}$. By the universal property of the adjoint pair, there is a natural isomorphism of hom-sets

$$\mathrm{Operad}(\mathsf{Free}\,\mathcal{S}, \mathcal{O}) \cong \mathrm{Species}(\mathcal{S}, \mathsf{Forget}\,\mathcal{O})$$

placing functors of operads $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{O}$ and maps of species $\phi : \mathcal{S} \to \mathsf{Forget}\,\mathcal{O}$ in one-to-one correspondence. In the sequel, we will leave the action of the forgetful functor implicit, writing $\mathcal{O}$ for both an operad and its underlying species $\mathsf{Forget}\,\mathcal{O}$.

We are now ready to introduce the main definitions of this section.

Definition 2.6 A **context-free grammar of arrows** is a tuple $G = (C, \mathcal{S}, S, \phi)$ consisting of a finitely generated category C, a finite species $\mathcal{S}$ equipped with a distinguished color $S \in \mathcal{S}$ called the *start symbol*, and a functor of operads $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[C]$. A color of $\mathcal{S}$ is then called a **non-terminal** while an operation of $\mathsf{Free}\,\mathcal{S}$ is called a **derivation**. The **context-free language of arrows** L_G generated by the grammar G is the subset of arrows in C which, seen as constants of $\mathcal{W}[C]$, are in the image of constants of color S in $\mathsf{Free}\,\mathcal{S}$, that is, $L_G = \{\, p(\alpha) \mid \alpha : S \,\}$.

As suggested in the Introduction and by Example 2.2, every context-free grammar in the classical sense $G = (\Sigma, N, S, P)$ corresponds to a context-free grammar over $\mathcal{B}_\Sigma$. For instance, for the grammar in Fig. 1, the corresponding species $\mathcal{S}$ has three colors and four nodes, and the functor p is uniquely defined by the action on the generators in $\mathcal{S}$ displayed in the middle of the figure. Conversely, any finite species $\mathcal{S}$ equipped with a color $S \in \mathcal{S}$ and a functor of operads $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[\mathcal{B}_\Sigma]$ uniquely determines a context-free grammar over the alphabet Σ. Indeed, the colors of $\mathcal{S}$ give the non-terminals of the grammar and S the distinguished start symbol, while the nodes of $\mathcal{S}$ together with the functor p give the production rules of the grammar, with each node $x : R_1, \ldots, R_n \to R$ such that $p(x) = w_0 - w_1 - \ldots - w_n$ determining a context-free production rule $x : R \to w_0 R_1 w_1 \ldots R_n w_n$.

Proposition 2.7 *A language $L \subseteq \Sigma^*$ is context-free in the classical sense if and only if it is the language of arrows of a context-free grammar over $\mathcal{B}_\Sigma$.*

An interesting feature of the general notion of context-free grammar of arrows $G = (C, \mathcal{S}, S, p)$ is that the non-terminals of the grammar are *sorted* in the sense that every color of $\mathcal{S}$ is mapped by p to a unique color of $\mathcal{W}[C]$,

as graphs do to categories, but that terminology unfortunately clashes with a different concept in graph theory. We use "species" to emphasize the link with Joyal's theory of (uncolored symmetric) species [15] and also with generalized species [11].

corresponding to a pair of objects of C. Adapting the conventions from our work on type refinement systems, we sometimes write $R \sqsubset (A, B)$ to indicate that $p(R) = (A, B)$ and say that R refines the "gap type" (A, B). The language L_G generated by a grammar with start symbol $S \sqsubset (A, B)$ is a subset of the hom-set $C(A, B)$.

Example 2.8 To illustrate some of the versatility afforded by the more general notion of context-free grammar of arrows, consider a CFG over the category $\mathcal{B}_\Sigma^\top$ from Example 2.4. Such a grammar may include production rules that can only be applied upon reaching the end of the input, which is useful in practice, albeit usually modelled in an ad hoc fashion. For example, the grammar of arithmetic expressions defined by Knuth in the original paper on LR parsing [17, example (27)] may be naturally described as a grammar over $\mathcal{B}_\Sigma^\top$, which in addition to having three "classical" non-terminals $E, T, P \sqsubset (*, *)$ contains a distinguished non-terminal $S \sqsubset (*, \top)$. Knuth's production $0 : S \to E\$$ is then just a unary node $0 : E \to S$ in $\mathcal{S}$, mapped by p to the operation $\epsilon - \$: (*, *) \to (*, \top)$ in $\mathcal{W}[\mathcal{B}_\Sigma^\top]$.

More significant examples of context-free languages of arrows over categories with more than one object will be given in §4, including context-free grammars over the runs of finite-state automata.

Finally, let us remark that context-free grammars of arrows may be organized into a comma category, observing that a grammar G may be equivalently considered as a triple of a *pointed* finite species $(\mathcal{S}, S)$, a *bipointed* finitely generated category (C, A, B), and a map of pointed operads $p : (\mathsf{Free}\,\mathcal{S}, S) \to (\mathcal{W}[C], (A, B))$. Since the operad of spliced arrows construction lifts to a functor $\mathcal{W}[-] : \mathrm{Cat}_{\bullet,\bullet} \to \mathrm{Operad}_\bullet$ sending a category C equipped with a pair of objects A and B to the operad $\mathcal{W}[C]$ equipped with the color (A, B), and likewise the free / forgetful adjunction (1) lifts to an adjunction between pointed species and pointed operads, a CFG can therefore be considered as an object of the comma category $\mathsf{Free} \downarrow \mathcal{W}$. Although we will not explore this perspective further here, let us mention that it permits another way of understanding the language of arrows generated by a grammar G: as constants of $\mathcal{W}[C]$ are in bijection with arrows of C, we have a natural isomorphism $\mathrm{el} \circ \mathcal{W}[-] \cong \hom$ for the evident functors $\hom : \mathrm{Cat}_{\bullet,\bullet} \to \mathrm{Set}$ and $\mathrm{el} : \mathrm{Operad}_\bullet \to \mathrm{Set}$, and L_G is precisely the image of the function $\mathrm{el}(p) : \mathrm{el}(\mathsf{Free}\,\mathcal{S}, S) \to \mathrm{el}(\mathcal{W}[C], (A, B)) \cong C(A, B)$.

2.3 *Properties of a context-free grammar and its associated language*

Standard properties of context-free grammars [30, Ch. 4], considered as CFGs of arrows $G = (\mathcal{B}_\Sigma, \mathcal{S}, S, p)$, may be reformulated as properties of either the species $\mathcal{S}$, the operad $\mathsf{Free}\,\mathcal{S}$, or the functor $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[\mathcal{B}_\Sigma]$, with varying degrees of naturality:

- G is *linear* just in case $\mathcal{S}$ only has nodes of arity ≤ 1. It is *left-linear* (respectively, *right-linear*) just in case it is linear and every unary node x of $\mathcal{S}$ is mapped by p to an operation of the form $\epsilon - w$ (resp. $p(x) = w - \epsilon$).
- G is in *Chomsky normal form* if $\mathcal{S}$ only has nodes of arity 2 or 0, the color S does not appear as the input of any node, every binary node is mapped by p to $\epsilon - \epsilon - \epsilon$ in $\mathcal{W}[\mathcal{B}_\Sigma]$, and every nullary node is mapped to a letter $a \in \Sigma$, unless $R = S$ in which case it is possible that $p(x) = \epsilon$. (This last condition can be made more natural by considering G as a context-free grammar over $\mathcal{B}_\Sigma^\top$ with $S \sqsubset (*, \top)$, see Example 2.8 above.)
- G is *bilinear* (a generalization of Chomsky normal form [19,22]) iff $\mathcal{S}$ only has nodes of arity ≤ 2.
- G is *unambiguous* iff for any pair of constants $\alpha, \beta : S$ in $\mathsf{Free}\,\mathcal{S}$, if $p(\alpha) = p(\beta)$ then $\alpha = \beta$. Note that if p is faithful then G is unambiguous, although faithfulness is a stronger condition in general.
- A non-terminal R of G is *nullable* if there exists a constant $\alpha : R$ of $\mathsf{Free}\,\mathcal{S}$ such that $p(\alpha) = \epsilon$.
- A non-terminal R of G is *useful* if there exists a pair of a constant $\alpha : R$ and a unary operation $\beta : R \to S$. Note that if G has no useless non-terminals then G is unambiguous iff p is faithful.

Observe that almost all of these properties can be immediately translated to express properties of context-free grammars of arrows over *any* category C. Basic closure properties of classical context-free languages also generalize easily to context-free languages of arrows.

Proposition 2.9 (i) *If $L_1, L_2 \subseteq C(A, B)$ are context-free languages of arrows, so is their union $L_1 \cup L_2 \subseteq C(A, B)$.*

(ii) *If $L_1 \subseteq C(A_1, B_1), \ldots, L_n \subseteq C(A_n, B_n)$ are context-free languages of arrows, and $w_0 - w_1 - \ldots - w_n : (A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ is an operation of $\mathcal{W}[C]$, then the "spliced concatenation" $w_0 L_1 w_1 \ldots L_n w_n = \{ w_0 u_1 w_1 \ldots u_n w_n \mid u_1 \in L_1, \ldots, u_n \in L_n \} \subseteq C(A, B)$ is also context-free.*

(iii) *If $L \subseteq C(A, B)$ is a context-free language of arrows in a category C and $F : C \to \mathcal{D}$ is a functor of categories, then the functorial image $F(L) \subseteq \mathcal{D}(F(A), F(B))$ is also context-free.*

Proof. The proofs of (i) and (ii) are just refinements of the standard proofs for context-free languages of words, keeping track of the underlying gap types. For (iii), suppose given a grammar $G = (C, \mathcal{S}, S, p)$ and a functor of categories $F : C \to \mathcal{D}$. Then the grammar $F(G)$ generating the language $F(L_G)$ is defined by postcomposing p with $\mathcal{W}[F] : \mathcal{W}[C] \to \mathcal{W}[\mathcal{D}]$ while keeping the species $\mathcal{S}$ and start symbol S the same, $F(G) = (\mathcal{D}, \mathcal{S}, S, p\mathcal{W}[F])$. □

We will see in §4.1 that other classical closure properties also generalize to context-free languages of arrows. Finally, we can state a **translation principle** that two grammars $G_1 = (C, \mathcal{S}_1, S_1, p_1)$ and $G_2 = (C, \mathcal{S}_2, S_2, p_2)$ over the same category have the same language whenever there is a fully faithful functor of operads $T : \mathsf{Free}\,\mathcal{S}_1 \to \mathsf{Free}\,\mathcal{S}_2$ such that $p_1 = T p_2$ and $T(S_1) = S_2$.

2.4 *A fibrational view of parsing as a lifting problem*

We have seen how any context-free grammar $G = (C, \mathcal{S}, S, p)$ gives rise to a language $L_G = \{\, p(\alpha) \mid \alpha : S \,\}$, corresponding to the arrows of C which, seen as constants of $\mathcal{W}[C]$, are in the image of some constant of color S of the free operad $\mathsf{Free}\,\mathcal{S}$. However, beyond characterizing the language defined by a grammar, in practice one is often confronted with a dual problem, namely that of parsing: given a word w, we want to compute the set of all its parse trees, or at least determine all of the non-terminals which derive it. In our functorial formulation of context-free derivations, this amounts to computing the inverse image of w along the functor p, i.e., the set of constants $p^{-1}(w) = \{\, \alpha \mid p(\alpha) = w \,\}$, or alternatively the set of colors in the image of $p^{-1}(w)$ along the output-color function.

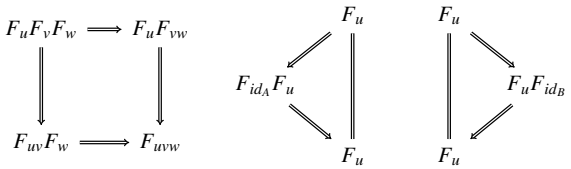
To better understand this view of parsing as a lifting problem along a functor of operads, we find it helpful to first recall the correspondence between functors of categories $p : \mathcal{D} \to \mathcal{T}$ and lax functors $F : \mathcal{T} \to \mathrm{Span(Set)}$, where Span(Set) is the bicategory whose objects are sets, whose 1-cells $S : X \nrightarrow Y$ are spans $X \leftarrow S \to Y$, and whose 2-cells are morphisms of spans. Suppose given such a functor $p : \mathcal{D} \to \mathcal{T}$. To every object A of $\mathcal{T}$ there is an associated "fiber" $F_A = p^{-1}(A)$ of objects in $\mathcal{D}$ living over A, while to every arrow $w : A \to B$ of $\mathcal{T}$ there is an associated fiber $F_w = p^{-1}(w)$ of arrows in $\mathcal{D}$ living over w, equipped with a pair of projection functions $F_A \leftarrow F_w \to F_B$ mapping any lifting $\alpha : R \to S$ of $w : A \to B$ to its source $R \in F_A$ and target $S \in F_B$. Moreover, given a pair of composable arrows $u : A \to B$ and $v : B \to C$ in $\mathcal{T}$, there is a morphism of spans

$$F_u F_v \Longrightarrow F_{uv} \quad : \quad F_A \nrightarrow F_C \tag{2}$$

from the composite of the spans $F_u : F_A \nrightarrow F_B$ and $F_v : F_B \nrightarrow F_C$ associated to $u : A \to B$ and $v : B \to C$ to the span $F_{uv} : F_A \nrightarrow F_C$ associated to the composite arrow $uv : A \to C$. This morphism of spans is realized using composition in the category $\mathcal{D}$, namely by the function taking any pair of a lifting $\alpha : R \to S$ of u and a lifting $\beta : S \to T$ of v to the composite $\alpha\beta : R \to T$, which is a lifting of uv by functoriality $p(\alpha\beta) = p(\alpha)p(\beta)$. Similarly, the identity arrows in the category $\mathcal{D}$ define, for every object A of the category $\mathcal{T}$, a morphism of spans

$$id_{F_A} \Longrightarrow F_{id_A} \quad : \quad F_A \nrightarrow F_A \tag{3}$$

from the identity span $F_A \leftarrow F_A \to F_A$ to the span associated to the identity arrow $id_A : A \to A$. Associativity and neutrality of composition in $\mathcal{D}$ ensure that the 2-cells (2) and (3) make the diagrams below commute:

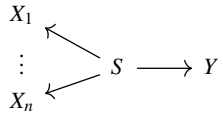


for all triples of composable arrows $u : A \to B$, $v : B \to C$ and $w : C \to D$, and therefore that this collection of data defines what is called a lax functor $F : \mathcal{T} \to \mathrm{Span(Set)}$. In general it is *only* lax, in the sense that the 2-cells $F_u F_v \Rightarrow F_{uv}$ and $id_{F_A} \Rightarrow F_{id_A}$ are not necessarily invertible.

Conversely, starting from the data provided by a lax functor $F : \mathcal{T} \to \mathrm{Span(Set)}$, we can define a category noted $\int F$ together with a functor $\pi : \int F \to \mathcal{T}$. The category $\int F$ has objects the pairs (A, R) of an object A in $\mathcal{T}$ and an element $R \in F_A$, and arrows $(w, \alpha) : (A, R) \to (B, S)$ the pairs of an arrow $w : A \to B$ in $\mathcal{T}$ and an element $\alpha \in F_w$ mapped to $R \in F_A$ and $S \in F_B$ by the respective legs of the span $F_A \leftarrow F_w \to F_B$. The composition and identity of the category $\int F$ are then given by the morphisms of spans $F_u F_v \Rightarrow F_{uv}$ and $id_{F_A} \Rightarrow F_{id_A}$ witnessing the

lax functoriality of $F : \mathcal{T} \to \mathrm{Span}(\mathrm{Set})$. The functor $\pi : \int F \to \mathcal{T}$ is given by the first projection. This construction of a category $\int F$ equipped with a functor $\pi : \int F \to \mathcal{T}$ starting from a lax functor $F : \mathcal{T} \to \mathrm{Span}(\mathrm{Set})$ is a mild variation of Bénabou's construction of the same starting from a lax normal functor $F : \mathcal{T}^{\mathrm{op}} \to \mathrm{Dist}$ [3, §7], which is itself a generalization of the well-known Grothendieck construction of a fibration starting from a pseudofunctor $F : \mathcal{T}^{\mathrm{op}} \to \mathrm{Cat}$. One can show that given a functor of categories $p : \mathcal{D} \to \mathcal{T}$, the construction applied to the associated lax functor $F : \mathcal{T} \to \mathrm{Span}(\mathrm{Set})$ induces a category $\int F$ isomorphic to $\mathcal{D}$, in such a way that p coincides with the isomorphism composed with π. Recently, Ahrens and Lumsdaine [1] have introduced the useful terminology "displayed category" to refer to this way of presenting a category $\mathcal{D}$ equipped with a functor $\mathcal{D} \to \mathcal{T}$ as a lax functor $\mathcal{T} \to \mathrm{Span}(\mathrm{Set})$, with their motivations coming from computer formalization of mathematics.

The constructions which turn a functor of categories $p : \mathcal{D} \to \mathcal{T}$ into a lax functor $F : \mathcal{T} \to \mathrm{Span}(\mathrm{Set})$ and back into a functor $\pi : \int F \to \mathcal{T}$ can be adapted smoothly to functors of operads, viewing $\mathrm{Span}(\mathrm{Set})$ as a 2-categorical operad whose n-ary operations $S : X_1, \ldots, X_n \nrightarrow Y$ are multi-legged-spans



or equivalently spans $X_1 \times \cdots \times X_n \leftarrow S \to Y$, and with the same notion of 2-cell. We will follow Ahrens and Lumsdaine's suggestion and refer to the data of such a lax functor $F : \mathcal{T} \to \mathrm{Span}(\mathrm{Set})$ representing an operad $\mathcal{D} \cong \int F$ equipped with a functor $p : \mathcal{D} \to \mathcal{T}$ as a **displayed operad**.

2.5 *An inductive formula for displayed free operads*

It is folklore that the free operad over a species $\mathcal{S} = (C, V, i, o)$ may be described concretely as a certain family of trees: operations of $\mathsf{Free}\,\mathcal{S}$ are interpreted as rooted planar trees whose edges are colored by the elements of C and whose nodes are labelled by the elements of V, subject to the constraints imposed by the functions $i : V \to C^*$ and $o : V \to C$. The formal construction of the free operad may be viewed as a free monoid construction, adapted to a situation where the ambient monoidal product (in this case, the composition product of species) is only distributive on the left, see [24, II.1.9] and [2, Appendix B].

From the perspective of programming semantics, it is natural to consider the underlying species of $\mathsf{Free}\,\mathcal{S}$ as an inductive data type, corresponding to the initial algebra for the endofunctor $W_{\mathcal{S}}$ on C-colored species defined by

$$W_{\mathcal{S}} = \mathcal{R} \mapsto \mathcal{I} + \mathcal{S} \circ \mathcal{R}$$

where $+$ denotes the coproduct of C-colored species which is constructed by taking the disjoint union of operations, while $\circ$ and $\mathcal{I}$ denote respectively the composition product of C-colored species and the identity species, defined as follows. Given two C-colored species $\mathcal{S}$ and $\mathcal{R}$, the n-ary nodes $R_1, \ldots, R_n \to R$ of $\mathcal{S} \circ \mathcal{R}$ are formal composites $g \bullet (f_1, \ldots, f_k)$ consisting of a node $g : S_1, \ldots, S_k \to S$ of $\mathcal{S}$ and of a tuple of nodes $f_1 : \Gamma_1 \to S_1, \ldots, f_k : \Gamma_k \to S_k$ of $\mathcal{R}$, such that the concatenation of the lists of colors $\Gamma_1, \ldots, \Gamma_k$ is equal to the list $R_1, \ldots, R_n$. The unit $\mathcal{I}$ is the C-colored species with a single unary node $*_R : R \to R$ for every color $R \in C$, and no other nodes.

As the initial $W_{\mathcal{S}}$-algebra, the free operad over $\mathcal{S}$ is equipped with a map of species $\mathcal{I} + \mathcal{S} \circ \mathsf{Free}\,\mathcal{S} \longrightarrow \mathsf{Free}\,\mathcal{S}$, which by the Lambek lemma is invertible, with the following interpretation: any operation of $\mathsf{Free}\,\mathcal{S}$ is either an identity operation, or the parallel composition of a node of $\mathcal{S}$ with a list of operations of $\mathsf{Free}\,\mathcal{S}$. Note that this interpretation also corresponds to a canonical way of decomposing trees labelled by the species $\mathcal{S}$, also known as *$\mathcal{S}$-rooted trees* [4, §3.2].

It is possible to derive an analogous inductive characterization of *functors* $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{O}$ from a free operad into an arbitrary operad $\mathcal{O}$ considered as displayed free operads, i.e., as lax functors $F : \mathcal{O} \to \mathrm{Span}(\mathrm{Set})$ generated by an underlying map of species $\phi : \mathcal{S} \to \mathcal{O}$. Two subtleties arise. First, that the species $\mathcal{S}$ and the operad $\mathcal{O}$ may in general have a different set of colors, related by the change-of-color function ϕ_C. To account for this, rather than restricting the operations $+, \circ, \mathcal{I}$ to the category of C-colored species, one should consider them as global functors

$$+, \circ : \mathrm{Species} \times_{\mathrm{Set}} \mathrm{Species} \to \mathrm{Species} \qquad \mathcal{I} : \mathrm{Set} \to \mathrm{Species}$$

on the "polychromatic" category of species, which respect the underlying sets of colors in a functorial way. Second, and more significantly, the above functor $W_{\mathcal{S}}$ transports a species $\mathcal{R}$ living over $\mathcal{O}$ to a species living over $\mathcal{I} + \mathcal{O} \circ \mathcal{O}$, so that in order to obtain again a species living over $\mathcal{O}$ (and thus define an endofunctor) one needs to "push forward"

along the canonical W_O-algebra $[e, m] : \mathcal{I} + O \circ O \longrightarrow O$ that encodes the operad structure of O, seen as a monoid in (Species, $\circ, \mathcal{I}$). A detailed proof is beyond the scope of this paper, but we nevertheless state the following:

Proposition 2.10 *Let $\phi : \mathcal{S} \to O$ be a map of species from a species $\mathcal{S}$ into an operad O, and let $p : \mathsf{Free}\,\mathcal{S} \to O$ be the corresponding functor from the free operad. Then the associated lax functor $F : O \to \mathrm{Span(Set)}$ computing the fibers of p is given by $F_A = \phi^{-1}(A)$ on colors of O, and by the least family of sets F_f indexed by operations $f : A_1, \ldots, A_n \to A$ of O such that*

$$F_f \cong \sum_{\substack{f = id_A \\ \phi(R) = A}} id_R + \sum_{f = g \circ (h_1, \ldots, h_k)} \phi^{-1}(g) \bullet (F_{h_1}, \ldots, F_{h_k}) \tag{4}$$

where we write $\circ$ for composition in the operad O and $\bullet$ for formal composition of nodes in $\mathcal{S}$ with operations in $\mathsf{Free}\,\mathcal{S}$. Specializing the formula to constant operations, the left summand disappears and (4) simplifies to:

$$F_c \cong \sum_{c = g \circ (c_1, \ldots, c_k)} \phi^{-1}(g) \bullet (F_{c_1}, \ldots, F_{c_k}) \tag{5}$$

2.6 Application to parsing

Instantiating (5) with the underlying functor $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[C]$ of a context-free grammar of arrows generated by a map of species $\phi : \mathcal{S} \to \mathcal{W}[C]$, we immediately obtain the following characteristic formula for the family of sets of parse trees F_w of an arrow w in C, seen as liftings of the constant w in $\mathcal{W}[C]$ to a constant in $\mathsf{Free}\,\mathcal{S}$:

$$F_w \cong \sum_{w = w_0 u_1 w_k \ldots u_n w_k} \phi^{-1}(w_0 - w_1 - \ldots - w_k) \bullet (F_{u_1}, \ldots, F_{u_k}) \tag{6}$$

Taking the image along the function returning the root label of a parse tree (i.e., the underlying color of the constant in $\mathsf{Free}\,\mathcal{S}$), we get that the family of sets of non-terminals N_w deriving w is the least family of sets closed under the following inference rule:

$$\frac{\begin{array}{c}(x : R_1, \ldots, R_k \to R) \in \mathcal{S} \\ w = w_0 u_1 w_1 \ldots u_k w_k \quad \phi(x) = w_0 - w_1 - \ldots - w_n \quad R_1 \in N_{u_1} \quad \ldots \quad R_k \in N_{u_k}\end{array}}{R \in N_w} \tag{7}$$

This inference rule is essentially the characteristic formula expressed by Leermakers [22] for the defining relation of the "C-parser", which generalizes the well-known Cocke-Younger-Kasami (CYK) algorithm. Presentations of the CYK algorithm are usually restricted to grammars in Chomsky normal form (cf. [19]), but as observed by Leermakers, the relation N_w defined by (7) can be solved effectively for any context-free grammar G and given word $w = a_1 \ldots a_n$ by building up a parse matrix $N_{i,j}$ indexed by the subwords $w_{i,j} = a_{i+1} \ldots a_j$ for all $1 \le i \le j \le n$, yielding a cubic complexity algorithm in the case that G is bilinear (cf. §2.3). Moreover, by adding non-terminals, it is always possible to transform a CFG into a bilinear CFG that generates the same language, even preserving the original derivations up to isomorphism.

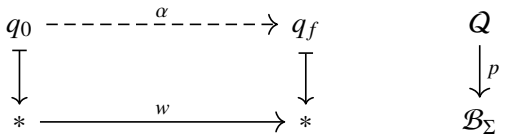
Proposition 2.11 *For any context-free grammar of arrows $G = (C, \mathcal{S}, S, p)$, there is a bilinear context-free grammar of arrows $G_{\mathsf{bin}} = (C, \mathcal{S}_{\mathsf{bin}}, S, p_{\mathsf{bin}})$ together with a fully faithful functor of operads $B : \mathsf{Free}\,\mathcal{S} \to \mathsf{Free}\,\mathcal{S}_{\mathsf{bin}}$ such that $p = Bp_{\mathsf{bin}}$. In particular, $L_G = L_{G_{\mathsf{bin}}}$ by the translation principle.*

3 Non-deterministic finite state automata as finitary ULF functors over categories and operads

3.1 Warmup: non-deterministic word automata as finitary ULF functors over categories

Classically, a non-deterministic finite state automaton $M = (\Sigma, Q, \delta, q_0, F)$ consists of a finite alphabet Σ, a finite set Q of states, a function $\delta : \mathit{Tran} \to Q \times \Sigma \times Q$ from a finite set *Tran* of transitions, an initial state $q_0 \in Q$ and a finite set of accepting states $F \subseteq Q$. We will focus first on the underlying "bare" automaton $M = (\Sigma, Q, \delta)$ where the initial and the accepting states have been removed. Every such bare automaton M induces a functor of categories

$p : \mathcal{Q} \to \mathcal{B}_\Sigma$ where $\mathcal{Q}$ is the category with the states of the automaton as objects, and with arrows freely generated by arrows of the form $t : q \to q'$ for any transition $t \in \mathit{Tran}$ such that $\delta(t) = (q, a, q')$; and where the functor $p : \mathcal{Q} \to \mathcal{B}_\Sigma$ transports every transition $t : q \to q'$ with $\delta(t) = (q, a, q')$ to the arrow $a : * \to *$ representing the letter $a \in \Sigma$ in the category $\mathcal{B}_\Sigma$. Under this formulation, every arrow $\alpha : q_0 \to q_f$ of the category $\mathcal{Q}$ describes a run of the automaton M over the word $w = p(\alpha) : * \to *$ which starts in state $q_0 \in Q$ and ends in state $q_f \in Q$, as depicted below:



One distinctive property of the functor $p : \mathcal{Q} \to \mathcal{B}_\Sigma$ is that it has the unique lifting of factorizations (ULF) property in the sense of Lawvere and Menni [21]. Recall that a functor of categories $p : \mathcal{D} \to \mathcal{T}$ has the ULF property when:

> For any arrow α of the category $\mathcal{D}$, if $p(\alpha) = uv$ for some pair of arrows u and v of the category $\mathcal{T}$, there exists a unique pair of arrows β and γ in $\mathcal{D}$ such that $\alpha = \beta\gamma$ and $p(\beta) = u$ and $p(\gamma) = v$.

Note that a functor $p : \mathcal{D} \to \mathcal{T}$ has the ULF property precisely when the structure maps of the corresponding lax functor $F : \mathcal{T} \to \mathrm{Span(Set)}$ discussed in §2.4 are invertible, i.e., F is a pseudofunctor. The ULF property implies an important structural property of non-deterministic finite state automata: that every arrow $\alpha : q_0 \to q_f$ lying above some arrow $p(\alpha) = w$ corresponding to a run of the automaton can be factored uniquely as a sequence of transitions along the letters of the word w. Conversely, we can easily establish that

Proposition 3.1 *A ULF functor $p : \mathcal{Q} \to \mathcal{B}_\Sigma$ corresponds to a bare non-deterministic finite state automaton precisely when the fiber $p^{-1}(*)$ as well as the fiber $p^{-1}(w)$ is finite for all words $w : * \to *$.*

This leads us to the following definitions.

Definition 3.2 We say that a functor $p : \mathcal{Q} \to C$ is **finitary** if either of the following equivalent conditions hold:

- the fiber $p^{-1}(A)$ as well as the fiber $p^{-1}(w)$ is finite for every object A and arrow w in the category C;
- the associated lax functor $F : C \to \mathrm{Span(Set)}$ factors via Span(FinSet).

Definition 3.3 A **non-deterministic finite state automaton over a category** is given by a tuple $M = (C, \mathcal{Q}, p : \mathcal{Q} \to C, q_0, q_f)$ consisting of two categories C and $\mathcal{Q}$, a finitary ULF functor $p : \mathcal{Q} \to C$, and a pair q_0, q_f of objects of $\mathcal{Q}$. An object of $\mathcal{Q}$ is then called a **state** and an arrow of $\mathcal{Q}$ is called a **run** of the automaton. The **regular language of arrows** L_M recognized by the automaton is the set of arrows w in C that can be lifted along p to an arrow $\alpha : q_0 \to q_f$ in $\mathcal{Q}$, that is $L_M = \{\, p(\alpha) \mid \alpha : q_0 \to q_f \,\}$.

Note that the regular language of arrows L_M recognized by an automaton M is a subset of the hom-set $C(A, B)$, where $A = p(q_0)$ and $B = p(q_f)$.

Remark 3.4 Any non-deterministic finite state automaton M in the standard sense may be converted into an automaton with a single accepting state (and without ϵ-transitions) that accepts the same language, *except* in the case that the language contains ϵ and is not closed under concatenation. The usual construction defines a new automaton M' with an additional state q_f and the same transitions as M, except that every transition $q \to q'$ to an accepting state $q' \in F$ of the old automaton is replaced by a transition $q \to q_f$ in the new automaton. The problem arises when the initial state q_0 is also accepting, in which case the language accepted by M' will be closed under concatenation.

Observe that this issue goes away if we instead consider the automaton obtained by transformation of M as an automaton $M' = (\mathcal{B}_\Sigma^\top, \mathcal{Q}', p', q_0, q_f)$ over the two-object category $\mathcal{B}_\Sigma^\top = \mathcal{B}_\Sigma +_\sigma \mathbf{1}$ defined in Example 2.4. Indeed, we can take $\mathcal{Q}'$ and p' to be defined from $\mathcal{Q}$ and p by adjoining a single object q_f lying over $\top$, together with a single arrow $q' \to q_f$ lying over $\$: * \to \top$ for every accepting state $q' \in F$ of M. Since arrows of type $* \to \top$ do not compose, the aforementioned problem does not arise.

Proposition 3.5 *A language $L \subseteq \Sigma^*$ is regular in the classical sense if and only if $L\$$ is the regular language of arrows of a non-deterministic finite state automaton over $\mathcal{B}_\Sigma^\top$.*

3.2 *Non-deterministic tree automata as finitary ULF functors over operads*

One nice aspect of the fibrational approach to non-deterministic finite state automata based on finitary ULF functors is that it adapts smoothly when one shifts from word automata to tree automata. As a first step in that direction, we first describe how the ULF and finite fiber properties may be extended to functors of operads.

Definition 3.6 A functor of operads $p : \mathcal{D} \to \mathcal{T}$ has the **unique lifting of factorizations property** (or **is ULF**) if any of the following equivalent conditions hold:

(i) for any operation α of $\mathcal{D}$, if $p(\alpha) = g \circ (h_1, \ldots, h_n)$ for some operation g and list of operations $h_1, \ldots, h_n$ of $\mathcal{T}$, there exists a unique operation β and list of operations $\gamma_1, \ldots, \gamma_n$ of $\mathcal{D}$ such that $\alpha = \beta \circ (\gamma_1, \ldots, \gamma_n)$ and $p(\beta) = g$ and $p(\gamma_1) = h_1, \ldots, p(\gamma_n) = h_n$;

(ii) for any operation α of $\mathcal{D}$, if $p(\alpha) = g \circ_i h$ for some operations g and h of $\mathcal{T}$ and index i, there exists a unique pair of operations β and γ of $\mathcal{D}$ such that $\alpha = \beta \circ_i \gamma$ and $p(\beta) = g$ and $p(\gamma) = h$;

(iii) the structure maps of the associated lax functor of operads $F : \mathcal{T} \to \mathrm{Span}(\mathrm{Set})$ discussed in §2.4 are invertible.

Definition 3.7 We say that a functor of operads $p : \mathcal{Q} \to \mathcal{O}$ is **finitary** if either of the following equivalent conditions hold:

- the fiber $p^{-1}(A)$ as well as the fiber $p^{-1}(f)$ is finite for every color A and operation f of the operad $\mathcal{O}$;
- the associated lax functor of operads $F : \mathcal{O} \to \mathrm{Span}(\mathrm{Set})$ factors via $\mathrm{Span}(\mathrm{FinSet})$.

One can check that the underlying bare automaton $M = (\Sigma, Q, \delta)$ of any non-deterministic finite state tree automaton [7] gives rise to a finitary ULF functor of operads $p : \mathcal{Q} \to \mathsf{Free}\,\Sigma$, where $\mathsf{Free}\,\Sigma$ is the free operad generated by the ranked alphabet Σ (which may be seen as an uncolored non-symmetric species), where the operad $\mathcal{Q}$ has states of the automaton as colors, and operations freely generated by n-ary nodes of the form $t : q_1, \ldots, q_n \to q$ for every transition $t \in \mathit{Tran}$ of the form $\delta(t) = (q_1, \ldots, q_n, a, q)$ where a is an n-ary letter in Σ, and where p transports every such n-ary transition $t : q_1, \ldots, q_n \to q$ to the underlying n-ary letter $a : *, \ldots, * \to *$. This motivates us to proceed as for word automata and propose a more general notion of finite state automaton over an arbitrary operad:

Definition 3.8 A **non-deterministic finite state automaton over an operad** is given by a tuple $M = (\mathcal{O}, \mathcal{Q}, p : \mathcal{Q} \to \mathcal{O}, q)$ consisting of two operads $\mathcal{O}$ and $\mathcal{Q}$, a finitary ULF functor of operads $p : \mathcal{Q} \to \mathcal{O}$, and a color q of $\mathcal{Q}$. A color of $\mathcal{Q}$ is called a **state**, and an operation of $\mathcal{Q}$ is called a **run tree** of the automaton $p : \mathcal{Q} \to \mathcal{O}$. The **regular language of constants** L_M recognized by the automaton is the set of constants c in $\mathcal{O}$ that can be lifted along p to a constant $\alpha : q$ in $\mathcal{Q}$, that is $L_M = \{\, p(\alpha) \mid \alpha : q \,\}$.

3.3 *From a word automaton to a tree automaton on spliced words*

We now state a simple property of ULF functors establishing a useful connection between word and tree automata.

Proposition 3.9 *Suppose that $p : \mathcal{Q} \to \mathcal{C}$ is a functor of categories. Then, if p is ULF functor, so is the functor of operads $\mathcal{W}[p] : \mathcal{W}[\mathcal{Q}] \to \mathcal{W}[\mathcal{C}]$. Moreover, if p is finitary then so is $\mathcal{W}[p]$.*

From this it follows that every non-deterministic finite state automaton $M = (\mathcal{C}, \mathcal{Q}, p, q_0, q_f)$ over a given category $\mathcal{C}$ induces a non-deterministic finite state automaton $\mathcal{W}[M] = (\mathcal{W}[\mathcal{C}], \mathcal{W}[\mathcal{Q}], \mathcal{W}[p], (q_0, q_f))$ over the spliced arrow operad $\mathcal{W}[\mathcal{C}]$. Moreover, it is immediate that $L_M = L_{\mathcal{W}[M]}$ since the constants of $\mathcal{W}[\mathcal{C}]$ are exactly the arrows of $\mathcal{C}$. As we will see, these observations play a central role in our understanding of the Chomsky and Schützenberger representation theorem [5]. Finally, let us emphasize that the notion of finite state automaton over an operad is really a proper generalisation of the classical notion of tree automaton since it allows taking a non-free operad as target, and in particular the non-free operad of sliced arrows (see Remark 2.3). This is what enables us to transform an automaton on the arrows of $\mathcal{C}$ into an automaton on the operations of the spliced arrow operad $\mathcal{W}[\mathcal{C}]$, which could be seen as a kind of tree automaton over "trees that bend" (cf. Fig. 2).

4 The Chomsky-Schützenberger Representation Theorem

4.1 Pulling back context-free grammars along finite state automata

Proposition 4.1 *Suppose given a species $\mathcal{S}$, a functor of operads $p : \mathsf{Free}\,\mathcal{S} \to \mathcal{O}$ and a ULF functor of operads $p_Q : \mathcal{Q} \to \mathcal{O}$. In that case, the pullback of p along p_Q in the category of operads is obtained from a corresponding pullback of $\phi : \mathcal{S} \to \mathcal{O}$ along $p_Q : \mathcal{Q} \to \mathcal{O}$ in the category of species:*

$$\begin{array}{ccc} \mathsf{Free}\,\mathcal{S}' & \xrightarrow{\mathsf{Free}\,\psi} & \mathsf{Free}\,\mathcal{S} \\ {\scriptstyle p'}\downarrow & \textit{pullback} & \downarrow{\scriptstyle p} \\ \mathcal{Q} & \xrightarrow[p_Q]{} & \mathcal{O} \end{array} \qquad \begin{array}{ccc} \mathcal{S}' & \xrightarrow{\psi} & \mathcal{S} \\ {\scriptstyle \phi'}\downarrow & \textit{pullback} & \downarrow{\scriptstyle \phi} \\ \mathcal{Q} & \xrightarrow[p_Q]{} & \mathcal{O} \end{array} \tag{8}$$

This observation may be applied to pull back a context-free grammar $G = (C, \mathcal{S}, S, p)$ of arrows in a category C, along a non-deterministic finite state automaton $M = (C, Q, p_M : Q_M \to C, q_0, q_f)$ over the same category. The construction is performed by first considering the ULF functor of operads of spliced arrows $\mathcal{W}[p_M] : \mathcal{W}[Q] \to \mathcal{W}[C]$ deduced from the ULF functor of categories p_M using Prop. 3.9. We therefore have a pullback diagram of the form (8) in the category of operads for $p_Q = \mathcal{W}[p_M]$ where the species $\mathcal{S}'$ and the map of species $\phi' : \mathcal{S}' \to \mathcal{W}[Q]$ determining p' are computed by a pullback in the category of species. This pullback admits a concrete description: the colors of $\mathcal{S}'$ are triples (q, R, q') where $p(R) = (p_M(q), p_M(q'))$ and its n-ary nodes $(q_1, R_1, q_1'), \ldots, (q_n, R_n, q_n') \to (q, R, q')$ are pairs (x, α) of a n-ary node $x : R_1, \ldots, R_n \to R$ of the species $\mathcal{S}_G$ together with a n-ary spliced arrow $\alpha = \alpha_0 - \ldots - \alpha_n : (q_1, q_1'), ..., (q_n, q_n') \to (q, q')$ in $\mathcal{W}[Q]$ such that $p_M(\alpha) = p(x)$, while the map of species ϕ' transports a color (q, R, q') to the color (q, q') of $\mathcal{W}[Q]$ and a n-ary node (x, α) to the n-ary operation α. Since $\mathcal{S}$ is finite and the functor p_M has finite fibers, the species $\mathcal{S}'$ is also finite. To complete the construction, the pullback grammar $G' = (Q, \mathcal{S}', S', p')$ is defined by taking the color $S' = (q_0, S, q_f)$ of the species $\mathcal{S}'$ as start symbol. Note that G' is a context-free grammar over the arrows of Q_M, which correspond to runs of the automaton M. In traditional syntax of context-free grammars, we could describe it as having a production rule $(q, R, q') \to \alpha_0(q_1, R_1, q_1')\alpha_1 \ldots (q_n, R_n, q_n')\alpha_n$ for every production rule $R \to w_0 R_1 w_1 \ldots R_n w_n$ of the original grammar G and sequence of $n + 1$ runs of the automaton $\alpha_0 : q \to q_1$, $\alpha_1 : q_1' \to q_2$, $\ldots$, $\alpha_n : q_n' \to q'$ over the respective words $w_0, \ldots, w_n$.

We can then also derive a grammar $G'' = p_M(G')$ of arrows in C by taking the functorial image (Prop. 2.9(iii)) of G' along the functor $p_M : Q \to C$, which by construction will generate the intersection of the context-free language of G and the regular language of M.

Proposition 4.2 *For G' and G'' defined as above, we have $L_{G'} = p_M^{-1}(L_G) \cap Q(q_0, q_f)$ and $L_{G''} = L_G \cap L_M$.*

Corollary 4.3 *Context-free languages of arrows are closed under pullback along non-deterministic finite state automata, and under intersection with regular languages.*

Example 4.4 For any word $w = a_1 \ldots a_n$ of length n, there is an $(n + 1)$-state automaton M_w that recognizes the singleton language $\{w\}$, with initial state 0, accepting state n, and transitions of the form $(i, a_{i+1}, i + 1)$ for each $0 \leq i < n$. By pulling back any context-free grammar G along M_w, we obtain a new grammar that may be seen as a specialization of G to the word w, with non-terminals (i, R, j) representing the fact that the subword $w_{i,j} = a_{i+1} \ldots a_j$ parses as R (cf. §2.6). This example generalizes to context-free grammars of arrows over any category C with the property that every arrow w has only finitely many factorizations $w = uv$ of length 2, by observing that the underlying bare automaton of M_w is isomorphic to the *interval category* [21] of w. In general, for any arrow $w : A \to B$ of a category C, the interval category $\mathrm{I}w$ is defined by taking objects to be triples (X, u, v) of an object $X \in C$ and a pair of arrows $u : A \to X$, $v : X \to B$ such that $w = uv$, and arrows $(X, u, v) \to (X', u', v')$ to be arrows $x : X \to X'$ such that $u' = ux$ and $v = xv'$. The interval category $\mathrm{I}w$ has an initial object (id_A, w) and a terminal object (w, id_B), and it comes equipped with an evident forgetful functor $\mathrm{I}w \to C$, which is always ULF, and moreover finitary by the stated condition on C. The tuple $M_w = (C, \mathrm{I}w, \mathrm{I}w \to C, (id_A, w), (w, id_B))$ therefore defines a finite-state automaton, and any CFG of arrows over C can be pulled back along M_w to obtain a CFG specialized to the arrow w.

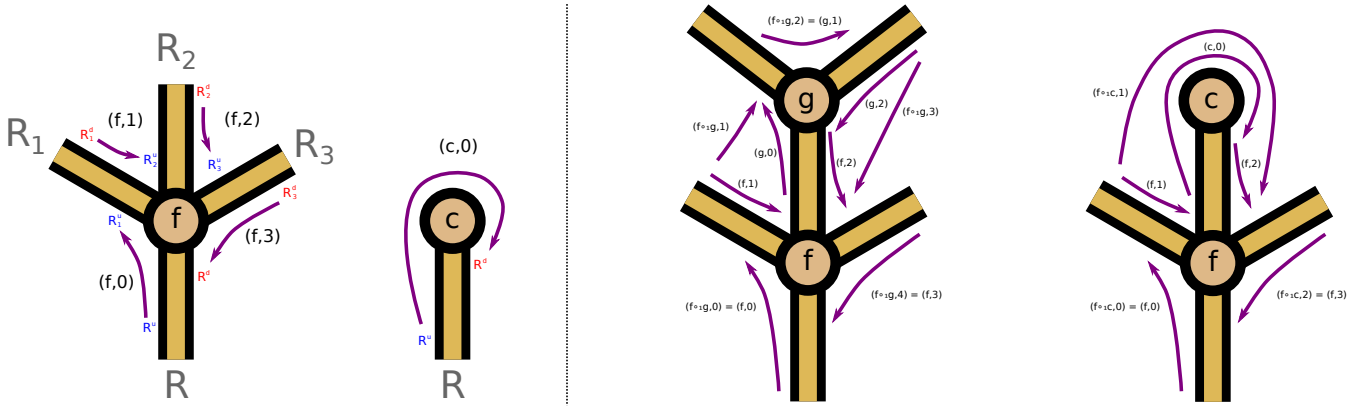


Fig. 3. Left: interpretation of the generating arrows of the contour category $C[\mathcal{O}]$. Right: interpretation of equations (9) and (10).

4.2 *The contour category of an operad and the contour / splicing adjunction*

In §2.1, we saw how to construct a functor

$$\mathcal{W}[-] : \mathrm{Cat} \to \mathrm{Operad}$$

transforming any category C into an operad $\mathcal{W}[C]$ of spliced arrows of arbitrary arity, which played a central role in our definition of context-free language of arrows in a category. We construct now a left adjoint functor

$$C[-] : \mathrm{Operad} \to \mathrm{Cat}$$

which extracts from any given operad $\mathcal{O}$ a category $C[\mathcal{O}]$ whose arrows correspond to "oriented contours" along the boundary of the operations of the operad.

Definition 4.5 The **contour category** $C[\mathcal{O}]$ of an operad $\mathcal{O}$ is defined as a quotient of the following free category:

- objects are given by *oriented colors* R^ϵ consisting of a color R of $\mathcal{O}$ and an orientation $\epsilon \in \{u, d\}$ ("up" or "down");
- arrows are generated by pairs (f, i) of an operation $f : R_1, \ldots, R_n \to R$ of $\mathcal{O}$ and an index $0 \leq i \leq n$, defining an arrow $R_i^d \to R_{i+1}^u$ under the conventions that $R_0^d = R^u$ and $R_{n+1}^u = R^d$;

subject to the conditions that $id_{R^u} = (id_R, 0)$ and $id_{R^d} = (id_R, 1)$ as well as the following equations:

$$(f \circ_i g, j) = \begin{cases} (f, j) & j < i \\ (f, i)(g, 0) & j = i \\ (g, j - i) & i < j < i + m \\ (g, m)(f, i + 1) & j = i + m \\ (f, j - m + 1) & j > i + m \end{cases} \tag{9}$$

$$(f \circ_i c, j) = \begin{cases} (f, j) & j < i \\ (f, i)(c, 0)(f, i + 1) & j = i \\ (f, j + 1) & j > i \end{cases} \tag{10}$$

whenever the left-hand side is well-formed, for every operation f, operation g of positive arity $m > 0$, constant c, and indices i and j in the appropriate range.

We refer to each generating arrow (f, i) of the contour category $C[\mathcal{O}]$ as a **sector** of the operation f. See Fig. 3 for a graphical interpretation of sectors and of the equations on contours seen as compositions of sectors.

Remark 4.6 In the case of a free operad over a species $\mathcal{S}$, we also write $C[\mathcal{S}]$ for the contour category $C[\mathsf{Free}\,\mathcal{S}]$ because it admits an even simpler description as a free category generated by the arrows $(x, i) : R_i^d \to R_{i+1}^u$ for every node $x : R_1, \ldots, R_n \to R$ of the species $\mathcal{S}$. We refer to these generating arrows (x, i) consisting of an n-ary node x

and an index $0 \leq i \leq n$ as **corners** since they correspond to the corners of $\mathcal{S}$-rooted trees seen as rooted planar maps [28]. Note that every sector of an operation of $\mathsf{Free}\,\mathcal{S}$ factors uniquely in the contour category $C[\mathcal{S}]$ as a sequence of corners. Thinking of the nodes of $\mathcal{S}$ as the production rules of a context-free grammar, the corners (x, i) correspond exactly to the *items* used in LR parsing and Earley parsing [17,10].

The contour construction provides a left adjoint to the spliced arrow construction because a functor of operads $\mathcal{O} \to \mathcal{W}[C]$ is entirely described by the data of a pair of objects $(A, B) = (R^u, R^d)$ in C for every color R in $\mathcal{O}$ together with a sequence $f_0, f_1, \ldots, f_n$ of $n+1$ arrows in C, where $f_i : R_i^d \to R_{i+1}^u$ for $0 \leq i \leq n$ for each operation $f : R_1, \ldots, R_n \to R$ of $\mathcal{O}$, under the same conventions as above. The equations (9) and (10) on the generators of $C[\mathcal{O}]$ reflect the equations imposed by the functor of operads $\mathcal{O} \to \mathcal{W}[C]$ on the spliced arrows of C appearing as the image of operations in $\mathcal{O}$. In that way we transform any functor of operads $\mathcal{O} \to \mathcal{W}[C]$ into a functor $C[\mathcal{O}] \to C$ which may be seen as an interpretion of the contours of the operations of $\mathcal{O}$ in C.

The unit and counit of the contour / splicing adjunction also have nice descriptions. The unit of the adjunction defines, for any operad $\mathcal{O}$, a functor of operads $\mathcal{O} \to \mathcal{W}[C[\mathcal{O}]]$ that acts on colors by $R \mapsto (R^u, R^d)$, and on operations by sending an operation $f : R_1, \ldots, R_n \to R$ of $\mathcal{O}$ to the spliced word of sectors $(f, 0) - \ldots - (f, n) : (R_1^u, R_1^d), \ldots, (R_n^u, R_n^d) \to (R^u, R^d)$. The counit of the adjunction defines, for any category C, a functor of categories $C[\mathcal{W}[C]] \to C$ that acts on objects by $(A, B)^u \mapsto A$ and $(A, B)^d \mapsto B$, and on arrows by sending the ith sector of a spliced word to its ith word, $(w_0 - \ldots - w_n, i) \mapsto w_i$.

In contrast to the situation of Prop. 3.9, it is *not* the case that $C[-]$ always preserves the ULF property.

Remark 4.7 Consider the category $\mathbf{2}$ with two objects A and B and only identity arrows, and the unique functor p to the terminal category $\mathbf{1}$. We claim that the associated ULF functor of operads $\mathcal{W}[p]$ induces a functor of categories $C[\mathcal{W}[p]]$ which is not ULF. Consider the two binary operations $f = id_A - id_A - id_A$ and $g = id_A - id_A - id_B$ and the constant $c = id_A$ in $\mathcal{W}[\mathbf{2}]$, as well as the binary operations $h = id_* - id_* - id_*$ and the constant $d = id_*$ in $\mathcal{W}[\mathbf{1}]$. The category $C[\mathcal{W}[\mathbf{2}]]$ has the sequence of sectors $\alpha = (f, 0)(c, 0)(g, 1)$ as an arrow, which is different from the identity. On the other hand, it is mapped by $C[\mathcal{W}[p]]$ to the sequence $w = (h, 0)(d, 0)(h, 1)$, which is equal thanks to Equation (10) to the sector $(h \circ_0 d, 0)$ of the unary operation $h \circ_0 d = id_* - id_*$ of $\mathcal{W}[\mathbf{1}]$, and hence $w = id_{(*,*)^u}$. Since the factorization $id = idid$ in $\mathcal{W}[\mathbf{1}]$ lifts to two distinct factorizations $\alpha = id\alpha = \alpha id$ in $\mathcal{W}[\mathbf{2}]$, p is not ULF.

Still, we can verify that maps of species induce ULF functors between their contour categories.

Proposition 4.8 *If $\psi : \mathcal{S} \to \mathcal{R}$ is a map of species, then $C[p] : C[\mathcal{S}] \to C[\mathcal{R}]$ is a ULF functor of categories.*

4.3 The universal context-free grammar of a pointed species, and its associated tree contour language

Every finite species $\mathcal{S}$ equipped with a color S comes with a *universal* context-free grammar $\mathsf{Univ}_{\mathcal{S},S} = (C[\mathcal{S}], \mathcal{S}, S, p_{\mathcal{S}})$, characterized by the fact that $p_{\mathcal{S}} : \mathsf{Free}\,\mathcal{S} \to \mathcal{W}[C[\mathcal{S}]]$ is the unit of the contour / splicing adjunction. By "universal" context-free grammar, we mean that any context-free grammar of arrows $G = (C, \mathcal{S}, S, p)$ with the same underlying species and start symbol factors uniquely through $\mathsf{Univ}_{\mathcal{S},S}$ in the sense that there exists a unique functor $q_G : C[\mathcal{S}] \to C$ satisfying the equation

$$\mathsf{Free}\,\mathcal{S} \xrightarrow{\;p\;} \mathcal{W}[C] \quad = \quad \mathsf{Free}\,\mathcal{S} \xrightarrow{\;p_{\mathcal{S}}\;} \mathcal{W}[C[\mathcal{S}]] \xrightarrow{\;\mathcal{W}[q_G]\;} \mathcal{W}[C]$$

We refer to the language of arrows $L_{\mathsf{Univ}_{\mathcal{S},S}}$, also noted $L_{\mathcal{S},S}$, as a **tree contour language**, and to its arrows as **tree contour words**, since they describe the contours of $\mathcal{S}$-rooted trees with root color S, see left side of Fig. 4 for an illustration. The factorization above shows that the context-free grammar G is the functorial image of the universal grammar $\mathsf{Univ}_{\mathcal{S},S}$ along the functor of categories q_G, whose purpose is to transport each corner of a node in $\mathcal{S}$ to the corresponding arrow in C as determined by the grammar G. At the level of languages, we have $L_G = q_G\, L_{\mathcal{S},S}$.

Remark 4.9 The notion of tree contour language makes sense even for non-finitary pointed species $(\mathcal{S}, S)$, although in that case the resulting universal grammar $\mathsf{Univ}_{\mathcal{S},S}$ is no longer context-free, having infinitely many non-terminals. Still, it may be an interesting object of study. In particular, the tree contour language $\mathsf{Univ}_{\mathbb{N},*}$ generated by the *terminal species* $\mathbb{N}$ with one color and a single operation of every arity appears to be of great combinatorial interest, with words in the language describing the shapes of rooted planar trees with arbitrary node degrees.

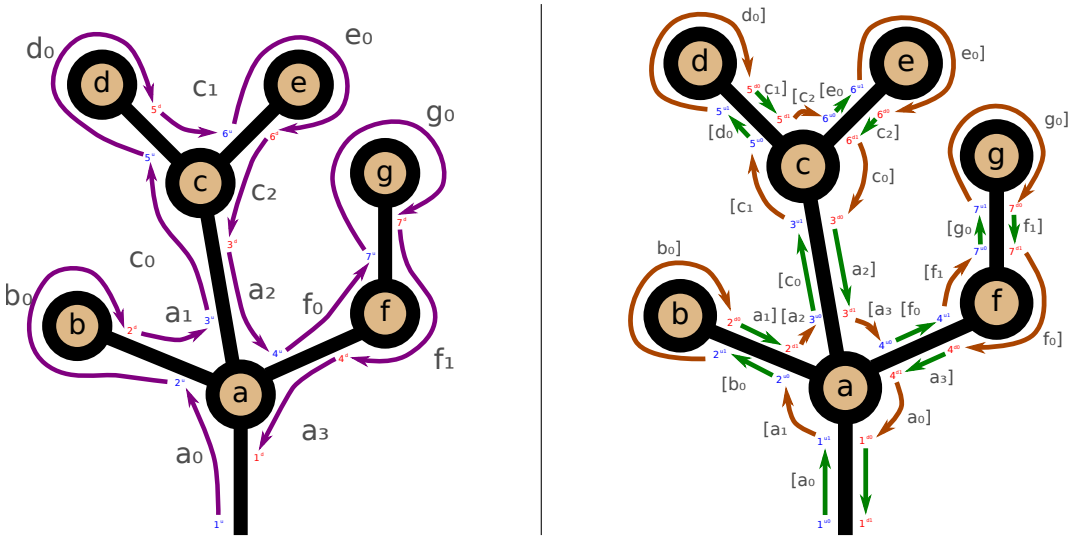


Fig. 4. Left: an $\mathcal{S}$-rooted tree of root color 1 and its corresponding contour word $\mathsf{a}_0\mathsf{b}_0\mathsf{a}_1\mathsf{c}_0\mathsf{d}_0\mathsf{c}_1\mathsf{e}_0\mathsf{c}_2\mathsf{a}_2\mathsf{f}_0\mathsf{g}_0\mathsf{f}_1\mathsf{a}_3 : 1^u \to 1^d$. Right: the corresponding Dyck word obtained by first decomposing each corner of the contour into alternating actions of walking along an edge and turning around a node, and then annotating each arrow both by the orientation (with $u = [, d =]$) and the node-edge pair of its target.

4.4 Representation theorem

The achievement of the Chomsky-Schützenberger representation theorem [5, §5] is to separate any context-free grammar $G = (\Sigma, N, S, P)$ into two independent components: a context-free grammar Dyck_n with only one non-terminal over an alphabet $\Sigma_{2n} = \{ [_1,]_1, \ldots, [_n,]_n \}$ of size $2n$ (for some n), which generates Dyck words of balanced brackets describing the shapes of parse trees with nodes labelled by production rules of G ; and a finite state automaton M to check that the edges of these trees may be appropriately colored by the non-terminals of G according to the labels of the nodes specifying the productions. The original context-free language L_G is then obtained as the image of the intersection of the languages generated by Dyck_n and by M, under a homomorphism $\Sigma_{2n}^* \to \Sigma^*$ that interprets each bracket of the Dyck word by a word in the original alphabet, with a choice to either interpret the open or the close brackets as empty words.

In this section, we give a new proof of the Chomsky-Schützenberger representation theorem, generalized to context-free grammars of arrows G over any category C. Since the category C may have more than one object, the appropriate statement of the representation theorem cannot require the grammar describing the shapes of parse trees to have only one non-terminal, but we can nonetheless construct one that is *C-chromatic* in the following sense.

Definition 4.10 A context-free grammar of arrows in C is C-chromatic when its non-terminals are the colors of $\mathcal{W}[C]$, in other words the pairs (A, B) of objects of the category C.

Moreover, rather than using Dyck words to represent parse trees, we find it more natural to use tree contour words, based on the observation given above (§4.3) that every context-free language may be *canonically* represented as the image of a tree contour language generated by a context-free grammar with the same set of non-terminals.

As preparation to our proof of the representation theorem, we establish:

Proposition 4.11 *Let $\phi : \mathcal{S} \to \mathcal{R}$ be a map of species with underlying change-of-color function ϕ_C. Let $\phi_C\,\mathcal{S}$ be the species with the same underlying set of nodes as $\mathcal{S}$, but where every node $x : R_1, \ldots, R_n \to R$ in $\mathcal{S}$ becomes a node $x : \phi_C(R_1), \ldots, \phi_C(R_n) \to \phi_C(R)$ in $\phi_C\,\mathcal{S}$. Then ϕ factors as*

$$\mathcal{S} \xrightarrow{\ \phi\ } \mathcal{R} \quad = \quad \mathcal{S} \xrightarrow{\ \phi_{\mathsf{colors}}\ } \phi_C\,\mathcal{S} \xrightarrow{\ \phi_{\mathsf{nodes}}\ } \mathcal{R}$$

where ϕ_{colors} is the identity on nodes and ϕ_{nodes} is the identity on colors.

Proposition 4.12 *Every map of species $\psi : \mathcal{S} \to \mathcal{S}'$ injective on nodes induces a commutative diagram*

$$\begin{array}{ccc} \mathsf{Free}\,\mathcal{S} & \xrightarrow{\mathsf{Free}\,\psi} & \mathsf{Free}\,\mathcal{S}' \\ {\scriptstyle p_{\mathcal{S}}}\downarrow & & \downarrow{\scriptstyle p_{\mathcal{S}'}} \\ \mathcal{W}[C[\mathcal{S}]] & \xrightarrow[\mathcal{W}[C[\psi]]]{} & \mathcal{W}[C[\mathcal{S}']] \end{array} \tag{11}$$

where the canonical functor of operads from $\mathsf{Free}\,\mathcal{S}$ *to the pullback of $p_{\mathcal{S}'}$ along $\mathcal{W}[C[\psi]]$ is fully faithful.*

Now, let $G = (C, \mathcal{S}, S, p)$ be any context-free grammar of arrows, and by Prop. 4.11 consider the corresponding C-chromatic grammar $G_{\mathsf{nodes}} = (C, \phi_C\,\mathcal{S}, (A, B), p_{\mathsf{nodes}})$, where $(A, B) = p(S)$. We have a commutative diagram

$$\begin{array}{ccc} \mathsf{Free}\,\mathcal{S} & \xrightarrow{\mathsf{Free}\,\phi_{\mathsf{colors}}} & \mathsf{Free}\,\phi_C\,\mathcal{S} \\ {\scriptstyle p_{\mathcal{S}}}\downarrow & & \downarrow{\scriptstyle p_{\phi_C\,\mathcal{S}}} \\ \mathcal{W}[C[\mathcal{S}]] & \xrightarrow[\mathcal{W}[C[\phi_{\mathsf{colors}}]]]{} & \mathcal{W}[C[\phi_C\,\mathcal{S}]] \\ & {\scriptstyle \mathcal{W}[q_G]}\searrow \quad \swarrow{\scriptstyle \mathcal{W}[q_{G_{\mathsf{nodes}}}]} & \\ & \mathcal{W}[C] & \end{array} \tag{12}$$

where the commutativity of the lower triangle follows from the equation $\phi = \phi_{\mathsf{colors}}\phi_{\mathsf{nodes}}$ and the contour / splicing adjunction. Note also that $C[\phi_{\mathsf{colors}}]$ is a ULF functor of categories by Prop. 4.8 and also finitary because ϕ_{colors} is finitary in the expected sense (and even finite). From this follows that $M_{\mathsf{colors}} = (C[\phi_C\,\mathcal{S}], C[\mathcal{S}], C[\phi_{\mathsf{colors}}], S^u, S^d)$ defines a finite-state automaton. By Props 4.2 and 4.12 and the translation principle, we deduce that

$$C[\phi_{\mathsf{colors}}]\,L_{\mathcal{S},S} \;=\; L_{\phi_C\,\mathcal{S},(A,B)} \,\cap\, L_{M_{\mathsf{colors}}}.$$

Finally, using that G is the image of the universal grammar $\mathsf{Univ}_{\mathcal{S},S}$ and considering the commutative diagram (12), we conclude:

$$L_G = q_G\,L_{\mathcal{S},S} = q_{G_{\mathsf{nodes}}}\,C[\phi_{\mathsf{colors}}]\,L_{\mathcal{S},S} = q_{G_{\mathsf{nodes}}}\,(L_{\mathcal{S}_{\phi_C\,\mathcal{S},(A,B)}} \,\cap\, L_{M_{\mathsf{colors}}}).$$

Theorem 4.13 *Every context-free language of arrows of a category C is the functorial image of the intersection of a C-chromatic context-free tree contour language with a regular language.*

The original statement of the Chomsky-Schützenberger theorem can be recovered by relying on the fact that any tree contour word can be faithfully translated to a Dyck word, via an easy translation that doubles the number of letters, and which also has a geometric interpretation that involves decomposing each corner of the contour into alternating actions of walking along an edge and turning around a node, see right side of Fig. 4. Intriguingly, this decomposition suggests the existence of an embedding of the contour category $C[\mathcal{S}]$ into a *bipartite* contour category, where each object R^ϵ has been replaced by a pair of objects $R^{\epsilon 0}$ and $R^{\epsilon 1}$, in a way that is analogous to the embedding of the "oriented cartographic group" used to represent maps on oriented surfaces into the cartographic group for maps on not necessarily orientable surfaces (cf. [29,14]).

Acknowledgement

We thank Bryce Clarke for helpful discussions about this work, and to the anonymous reviewers for comments improving the presentation.

References

[1] Ahrens, B. and P. L. Lumsdaine, *Displayed categories*, Logical Methods in Computer Science **15** (2019).

[2] Baues, H.-J., M. Jibladze and A. Tonks, *Cohomology of monoids in monoidal categories*, Contemporary Mathematics **202** (1997).

[3] Bénabou, J., *Distributors at work* (2000), notes from a course at TU Darmstadt in June 2000, taken by Thomas Streicher.
URL https://www2.mathematik.tu-darmstadt.de/~streicher/FIBR/DiWo.pdf

[4] Bergeron, F., G. Labelle and P. Leroux, "Combinatorial Species and Tree-Like Structures," Cambridge University Press, 1998, translated by Margaret Readdy.

[5] Chomsky, N. and M. Schützenberger, *The algebraic theory of context-free languages*, in: P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, Studies in Logic and the Foundations of Mathematics **35**, North-Holland, 1963 pp. 118–161.

[6] Colcombet, T. and D. Petrişan, *Automata minimization: a functorial approach*, Logical Methods in Computer Science **16** (2020), pp. 32:1–32:28.

[7] Comon, H., M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison and M. Tommasi, *Tree Automata Techniques and Applications* (2008).
URL https://hal.inria.fr/hal-03367725

[8] de Groote, P., *Towards abstract categorial grammars*, in: *Association for Computational Linguistic, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference, July 9-11, 2001, Toulouse, France* (2001), pp. 148–155.

[9] de Groote, P. and S. Pogodalla, *On the expressive power of abstract categorial grammars: Representing context-free formalisms*, J. Log. Lang. Inf. **13** (2004), pp. 421–438.

[10] Earley, J., *An efficient context-free parsing algorithm*, Commun. ACM **13** (1970), pp. 94–102.

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

[12] Girard, J.-Y., *Linear logic*, Theoretical Computer Science **50** (1987), pp. 1–102.

[13] Girard, J.-Y., *Geometry of interaction 1: Interpretation of System F*, in: R. Ferro, C. Bonotto, S. Valentini and A. Zanardo, editors, *Logic Colloquium '88*, Studies in Logic and the Foundations of Mathematics **127**, Elsevier, 1989 pp. 221–260.

[14] Jones, G. A. and D. Singerman, *Maps, hypermaps, and triangle groups*, in: L. Schneps, editor, *The Grothendieck Theory of Dessins d'Enfants*, number 200 in London Mathematical Society Lecture Note Series, Cambridge University Press, 1994 .

[15] Joyal, A., *Une théorie combinatoire des séries formelles*, Advances in Mathematics **42** (1981), pp. 1–82.

[16] Joyal, A., *Foncteurs analytiques et espèces de structures*, in: G. Labelle and P. Leroux, editors, *Combinatoire énumérative*, Lecture Notes in Mathematics (1986), pp. 126–159.

[17] Knuth, D. E., *On the translation of languages from left to right*, Information and Control **8** (1965), pp. 607–639.

[18] Lambek, J., *Multicategories revisited*, Contemporary Mathematics **92** (1989), pp. 217–239.

[19] Lange, M. and H. Leiß, *To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm*, Informatica Didact. **8** (2009).

[20] Lawvere, F. W., *Ordinal sums and equational doctrines*, in: B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, Lecture Notes in Mathematics, 1969, pp. 141–155.

[21] Lawvere, F. W. and M. Menni, *The Hopf algebra of Möbius intervals*, Theory and Applications of Categories **24** (2010), pp. 221–265.

[22] Leermakers, R., *How to cover a grammar*, in: J. Hirschberg, editor, *27th Annual Meeting of the Association for Computational Linguistics, 26-29 June 1989, University of British Columbia, Vancouver, BC, Canada, Proceedings* (1989), pp. 135–142.

[23] Leinster, T., "Higher Operads, Higher Categories," London Mathematical Society Lecture Note Series **298**, Cambridge University Press, 2004.

[24] Markl, M., S. Schnider and J. Stasheff, "Operads in Algebra, Topology and Physics," Mathematical Surveys and Monographs **96**, American Mathematical Society, 2002.

[25] Melliès, P. and N. Zeilberger, *Functors are type refinement systems*, in: *POPL* (2015), pp. 3–16.

[26] Melliès, P. and N. Zeilberger, *A bifibrational reconstruction of Lawvere's presheaf hyperdoctrine*, in: *LICS* (2016), pp. 555–564.

[27] Melliès, P. and N. Zeilberger, *An Isbell duality theorem for type refinement systems*, Mathematical Structures in Computer Science **28** (2018), pp. 736–774.

[28] Schaeffer, G., *Planar maps*, in: M. Bóna, editor, *Handbook of Enumerative Combinatorics*, CRC, 2015 pp. 335–396.

[29] Shabat, G. and V. Voevodsky, *Drawing curves over number fields*, in: P. Cartier, L. Illusie, N. M. Katz, G. Laumon, Y. I. Manin and K. A. Ribet, editors, *The Grothendieck festschrift III*, number 88 in Progress in Mathematics, Birkhäuser, 1990 pp. 199–227.

[30] Sippu, S. and E. Soisalon-Soininen, "Parsing Theory - Volume I: Languages and Parsing," EATCS Monographs on Theoretical Computer Science **15**, Springer, 1988.

[31] Slavnov, S., *Classical linear logic, cobordisms and categorial grammar* (2020), arXiv:1911.03962.

[32] Walters, R. F. C., *A note on context-free languages*, Journal of Pure and Applied Algebra **62** (1989), pp. 199–203.

A Supplementary proofs

Proofs of Props. 2.9(i) and (ii).

(i) Given two grammars $G_1 = (C, \mathcal{S}_1, S_1, p_1)$ and $G_2 = (C, \mathcal{S}_2, S_2, p_2)$, where S_1 and S_2 both refine the same gap type (A, B), we define a new grammar $G = (C, \mathcal{S}, S, p)$ that generates the union of the two languages $L_G = L_{G_1} \cup L_{G_2}$ by taking $\mathcal{S}$ to be the disjoint union of the colors and operations of $\mathcal{S}_1$ and $\mathcal{S}_2$ combined with a distinguished color S and pair of unary nodes $i_1 : S_1 \to S$ and $i_2 : S_2 \to S$, and defining $\phi : \mathcal{S} \to \mathcal{W}[C]$ to be the copairing of ϕ_1 and ϕ_2 extended with the mappings $\phi(S) = (A, B)$ and $\phi(i_1) = \phi(i_2) = id_A - id_B$.

(ii) Given grammars $G_1 = (C, \mathcal{S}_1, S_1, p_1), \ldots, G_n = (C, \mathcal{S}_n, S_n, p_n)$ where $S_i \sqsubset (A_i, B_i)$ for each $1 \le i \le n$, together with an operation $w_0 - w_1 - \ldots - w_n : (A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ of $\mathcal{W}[C]$, we construct a new grammar $G = (C, \mathcal{S}, S, p)$ that generates the spliced concatenation $w_0 L_{G_1} w_1 \ldots L_{G_n} w_n$ by taking $\mathcal{S}$ to be the disjoint union of the colors and operations of $\mathcal{S}_1, \ldots, \mathcal{S}_n$ combined with a distinguished color S and a single n-ary node $x : S_1, \ldots, S_n \to S$, and defining $\phi : \mathcal{S} \to \mathcal{W}[C]$ to be the cotupling of $\phi_1, \ldots, \phi_n$ extended with the mappings $\phi(S) = (A, B)$ and $\phi(x) = w_0 - w_1 - \ldots - w_n$.

Proof of Prop. 2.11

Let $G = (C, \mathcal{S}, S, p)$ be a context-free grammar of arrows. A bilinear grammar $G_{\mathsf{bin}} = (C, \mathcal{S}_{\mathsf{bin}}, S, p_{\mathsf{bin}})$ over the same category and with the same start symbol is constructed as follows. $\mathcal{S}_{\mathsf{bin}}$ includes all of the colors and all of the nullary nodes of $\mathcal{S}$, with $\phi_{\mathsf{bin}}(R) = \phi(R)$ and $\phi_{\mathsf{bin}}(c) = \phi(c)$. Additionally, for every node $x : R_1, \ldots, R_n \to R$ of $\mathcal{S}$ of positive arity $n > 0$, where $\phi(x) = w_0 - \ldots - w_n : (A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ in $\mathcal{W}[C]$, we include in $\mathcal{S}_{\mathsf{bin}}$:

- n new colors $I_{x,0}, \ldots, I_{x,n-1}$, with $\phi_{\mathsf{bin}}(I_{x,i-1}) = (A, A_i)$ for $1 \le i \le n$;
- one nullary node $x_0 : I_{x,0}$, with $\phi_{\mathsf{bin}}(x_0) = w_0$;
- n binary nodes $x_1, \ldots, x_n$, where $x_i : I_{x,i-1}, R_i \to I_{x,i}$ and $\phi_{\mathsf{bin}}(x_i) = id_A - id_{A_i} - w_i$ for all $1 \le i \le n$, under the convention that $I_{x,n} = R$.

We define the functor $B : \mathsf{Free}\,\mathcal{S} \to \mathsf{Free}\,\mathcal{S}_{\mathsf{bin}}$ on colors by $B(R) = R$, on nullary nodes by $B(c) = c$, and on nodes $x : R_1, \ldots, R_n \to R$ of positive arity by $B(x) = x_n \circ_0 \cdots \circ_0 x_1 \circ_0 x_0$. By induction on n, there is a one-to-one correspondence between nodes $x : R_1, \ldots, R_n \to R$ of $\mathcal{S}$ and operations $B(x) : R_1, \ldots, R_n \to R$ of $\mathsf{Free}\,\mathcal{S}_{\mathsf{bin}}$, so the functor B is fully faithful.

Proof of Prop. 4.1

Suppose $p_Q : Q \to O$ is a ULF functor of operads, and consider the pullback of $\phi : \mathcal{S} \to O$ along $p_Q : Q \to O$ in the category of species:

$$\begin{array}{ccc} \mathcal{S}' & \xrightarrow{\psi} & \mathcal{S} \\ {\scriptstyle \phi'}\downarrow & \textit{pullback} & \downarrow{\scriptstyle \phi} \\ Q & \xrightarrow[p_Q]{} & O \end{array}$$

We wish to show that there is a corresponding pullback in the category of operads:

$$\begin{array}{ccc} \mathsf{Free}\,\mathcal{S}' & \xrightarrow{\mathsf{Free}\,\psi} & \mathsf{Free}\,\mathcal{S} \\ {\scriptstyle p'}\downarrow & \textit{pullback} & \downarrow{\scriptstyle p} \\ Q & \xrightarrow[p_Q]{} & O \end{array}$$

Note that Free $\mathcal{S}'$ and $\mathcal{S}'$ have the same colors, namely pairs (R, R') of a color R in $\mathcal{S}$ and a color R' in $\mathcal{Q}$ such that $\phi(R) = p_{\mathcal{Q}}(R')$. It suffices to show that any pair (α, α') of an operation α of Free $\mathcal{S}$ and an operation α' of $\mathcal{Q}$ such that $p(\alpha) = p_{\mathcal{Q}}(\alpha')$ corresponds to a unique operation β of Free $\mathcal{S}'$ such that (Free ψ)(β) $= \alpha$ and $p'(\beta) = \alpha'$. Now, by the inductive characterization of Free $\mathcal{S}$ (cf. §2.5), there are two cases to consider:

- $\alpha = id_R$ is an identity operation. Since $p_{\mathcal{Q}}(\alpha') = p(\alpha) = id_{p(R)}$ and ULF functors have unique liftings of identities, α' must also be an identity $\alpha' = id_{R'}$ for some R' such that $p_{\mathcal{Q}}(R') = p(R)$. We take $\beta = id_{(R,R')}$.
- $\alpha = x \bullet (\gamma_1, \ldots, \gamma_n)$ is a formal composition of some n-ary node x of $\mathcal{S}$ with operations $\gamma_1, \ldots, \gamma_n$ of Free $\mathcal{S}$. Since $p_{\mathcal{Q}}(\alpha') = p(\alpha) = \phi(x) \circ (p(\gamma_1), \ldots, p(\gamma_n))$, by ULF there exist unique $\beta', \gamma'_1, \ldots, \gamma'_n$ such that $\alpha' = \beta' \circ (\gamma'_1, \ldots, \gamma'_n)$ and $p_{\mathcal{Q}}(\beta') = \phi(x)$ and $p_{\mathcal{Q}}(\gamma'_1) = p(\gamma_1), \ldots, p_{\mathcal{Q}}(\gamma'_n) = p(\gamma_n)$. We take $\beta = (x, \beta') \bullet ((\gamma_1, \gamma'_1), \ldots, (\gamma_n, \gamma'_n))$.

Proof of Prop. 4.12.

Every map of species $\psi : \mathcal{S} \to \mathcal{S}'$ induces a naturality square

$$\begin{array}{ccc} \text{Free } \mathcal{S} & \xrightarrow{\text{Free } \psi} & \text{Free } \mathcal{S}' \\ {\scriptstyle p_{\mathcal{S}}}\downarrow & & \downarrow{\scriptstyle p_{\mathcal{S}'}} \\ \mathcal{W}[C[\mathcal{S}]] & \xrightarrow[\mathcal{W}[C[\psi]]]{} & \mathcal{W}[C[\mathcal{S}']] \end{array}$$

in the category of operads where the functors of operads $p_{\mathcal{S}}$ and $p_{\mathcal{S}'}$ associated to the universal grammars are the units of the contour / splicing adjunction, see §4.2 and §4.3. By Prop. 4.1, we know that the pullback of $p_{\mathcal{S}'}$ along $\mathcal{W}[C[\psi]]$ is obtained from a corresponding pullback in the category of species

$$\begin{array}{ccc} \mathcal{R} & \xrightarrow{\psi} & \mathcal{S}' \\ {\scriptstyle \rho}\downarrow & \textit{pullback} & \downarrow{\scriptstyle \phi_{\mathcal{S}'}} \\ \mathcal{W}[C[\mathcal{S}]] & \xrightarrow[\mathcal{W}[C[\psi]]]{} & \mathcal{W}[C[\mathcal{S}']] \end{array}$$

The pullback $\mathcal{R}$ of the map of species $\phi_{\mathcal{S}'}$ along the ULF functor of operads $\mathcal{W}[C[\psi]]$ is the species with colors defined as triples $(R, (R_1^u, R_2^d))$ where R is a color of $\mathcal{S}'$ and R_1, R_2 are colors of $\mathcal{S}$ such that $\psi(R_1) = \psi(R_2) = R$; and with n-ary nodes defined as pairs (x, f) where x is a n-ary node in $\mathcal{S}'$ and f is an n-ary operation in $\mathcal{W}[C[\mathcal{S}]]$ necessarily of the form $f = (y, 0) - \ldots - (y, n)$, for y the unique n-ary node of $\mathcal{S}$ such that $\psi(y) = x$, since the map of species $\psi : \mathcal{S} \to \mathcal{S}'$ is injective of nodes. The canonical map of species $\mathcal{S} \to \mathcal{R}$ transports every color R of $\mathcal{S}$ to the color $(R, \psi(R)^u, \psi(R)^d)$ and every n-ary node y of $\mathcal{S}$ to the n-ary node $(\psi(y), (y, 0) - \ldots - (y, n))$ of $\mathcal{R}$. From this follows that the canonical map of species $\mathcal{S} \to \mathcal{R}$ is injective on colors and bijective on nodes. Moreover, there are no nodes in $\mathcal{R}$ whose colors are outside of the image of $\mathcal{S}$. We conclude that the canonical functor of operads Free $\mathcal{S} \to$ Free $\mathcal{R}$ is fully faithful.

Contents