Parsing as a lifting problem and the Chomsky-Schützenberger Representation Theorem

Paul-André Melliès
Noam Zeilberger

Topos Institute Colloquium
30 June 2022

based on a paper to be presented at MFPS 2022
preliminary version: https://hal.archives-ouvertes.fr/hal-03702762 (comments welcome!)
1. Introduction
A functorial view of type systems
(cf. M&Z, "Functors are Type Refinement Systems", POPL 2015)

Manifesto.

The standard interpretation of type systems as categories collapses the distinction between terms, typing judgments, and typing derivations, and is therefore inadequate for understanding what type systems do mathematically. Instead, type systems are better modelled as functors $p : \mathcal{D} \to \mathcal{T}$ from a category $\mathcal{D}$ whose morphisms are typing derivations to a category $\mathcal{T}$ whose morphisms are the terms corresponding to the underlying subjects of those derivations.
Typing as a lifting problem

\[ \mathbb{T} \]

\[ f : A \to B \]

f is a term with "intrinsic" type \( A \to B \)
Typing as a lifting problem

The triple \((R,f,S)\) form a typing judgment, asserting that \(f\) may be assigned an "extrinsic" type \(R \rightarrow S\).
Typing as a lifting problem

α is a typing derivation providing evidence for the judgment
A functorial view of context-free grammars

We developed this perspective in a series of papers, and believe 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 CS.

Our starting point will be to represent CFGs as functors of operads $p : \mathcal{D} \to \mathcal{T}$, where $\mathcal{D}$ is a freely generated (colored) operad and $\mathcal{T} = W[\Sigma]$ is something we call the "operad of spliced words".
Reminder on operads

(Usage note: "operad" = colored operad = multicategory.)

**operations**

\[ f : R, B, G \to Y \]

\[ g : Y, P \to B \]

\[ c : R \]

**identity**

\[ \text{id}_G : G \to G \]

**partial / parallel composition**

\[ f \circ_0 c : B, G \to Y \]

\[ f \circ (c, g, \text{id}_G) : Y, P, G \to Y \]

+ associativity

&

unitality axioms
Reminder on CFGs

A context-free grammar is a tuple $G = (\Sigma, N, S, P)$ consisting of:

- a finite set $\Sigma$ of *terminal symbols*
- a finite set $N$ of *non-terminal symbols*
- a distinguished element $S \in N$ called the *start symbol*
- a finite set $P$ of *production rules* $R \rightarrow \sigma$ where $R \in N$ and $\sigma \in (N \cup \Sigma)^*$

We write $\sigma_1 \Rightarrow \sigma_2$ if there exist $\rho, \tau \in (N \cup \Sigma)^*$ and a production rule $R \rightarrow \sigma$ such that $\sigma_1 = \rho R \tau$, $\sigma_2 = \rho \sigma \tau$. The *language* of $G$ is the set of strings $\{ w \in \Sigma^* \mid S \Rightarrow^+ w \}$.
The operad of spliced words

Observation: any production rule can be factored as $R \rightarrow w_0 R_1 w_1 ... R_n w_n$, where $w_0, w_1, ..., w_n \in \Sigma^*$ and $R_1, ..., R_n \in N$.

If we forget the non-terminals, the remaining sequence $w_0 - w_1 - ... - w_n$ can be seen as an n-ary operation of the operad of spliced words $W[\Sigma]$. Composition in this (uncolored) operad is performed by "splicing into the gaps", for example:

$$(ha-ha-ha) \circ (foo,bar-baz) = hafoohabar-bar-bazha$$
Representing CFGs as functors of operads: example

\[ D \xrightarrow{W[\Sigma]} 1 : S \to NP \ VP \]
\[ 2 : NP \to mom \]
\[ 3 : NP \to tom \]
\[ 4 : VP \to loves \ NP \]

\[ \epsilon \cdot u \cdot \epsilon \]  \( \xrightarrow{\cdot} \)  \[ mom \]
\[ \xrightarrow{\cdot} \]  \[ tom \]
\[ \xrightarrow{\cdot} \]  \[ loves_{u \cdot \epsilon} \]

\[ \epsilon \cdot u \cdot \epsilon \circ (tom, loves_{u \cdot \epsilon} \circ mom) = tom \cdot loves_{u \cdot \epsilon} \circ mom \]
Plan for the talk

It turns out that taking "spliced words" extends to a functor $W[-] : \text{Cat} \rightarrow \text{Operad}$, allowing us to define CFGs of arrows over any category. We’ll see that representing CFGs as functors leads to a simplification of many standard concepts, and that closure properties of CF languages generalize to CF languages of arrows.

Later, we will see that $W[-]$ has a left adjoint $C[-] : \text{Operad} \rightarrow \text{Cat}$. This construction, called the "contour category" of an operad, has a nice geometric interpretation, and we will use it to prove (a refinement and generalization of) the Chomsky-Schützenberger Representation Theorem*.

In between, we will also talk about automata over categories and operads.

*original version: « any CF language is the homomorphic image of the intersection of a Dyck language with a regular language »
Related work

The idea of defining CFGs as functors from free multicategories was discussed briefly by R.F.C. Walters in "A note on context-free languages", JPAA 62 (1989).

This idea is also closely related to Philippe de Groote’s encoding of CFGs as *abstract categorial grammars*, although the ACG work is expressed within a λ-calculus framework rather than a categorical / operadic one.

See introduction to our paper for a bit more discussion of related work. Additional pointers to related work are of course welcome. (Has the contour / splicing adjunction not been noticed before??)
2. Context-free languages of arrows
The operad of spliced arrows

Let $\mathbb{C}$ be a category. The operad $W[\mathbb{C}]$ is defined as follows:

- its colors are pairs $(A,B)$ of objects of $\mathbb{C}$;
- 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 $\mathbb{C}$ 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” (see next slide);
- the identity operation on $(A,B)$ is given by $\text{id}_A - \text{id}_B$.

($W[\mathbb{C}]$ generalizes $W[\Sigma]$, taking $\mathbb{C} = \mathbb{B}\Sigma$ the free monoid seen as one-object category.)
The operad of spliced arrows

$w_0 - w_1 - w_2 - w_3 : (A_1, B_1), (A_2, B_2), (A_3, B_3) \rightarrow (A, B)$
The operad of spliced arrows

identity

partial composition

=
The splicing functor

The operad of spliced arrows construction defines a functor

\[
\text{Cat} \xrightarrow{W[-]} \text{Operad}
\]

since any functor of categories \( F : \mathbb{C} \to \mathbb{D} \) induces a functor of operads \( W[F] : W[\mathbb{C}] \to W[\mathbb{D}] \).
Species (some terminology)

A (colored non-symmetric) species is a span of sets of the form

\[ \begin{array}{ccc}
  C^* & \xleftarrow{i} & V \\
  \downarrow & & \downarrow \\
  D^* & \xleftarrow{i'} & W \\
\end{array} \quad \begin{array}{ccc}
  V & \xrightarrow{o} & C \\
  \downarrow & & \downarrow \\
  W & \xrightarrow{o'} & D \\
\end{array} \]

with the following interpretation: \( C \) is a set of "colors", \( V \) a set of "nodes", and \( i : V \to C^* \) and \( o : V \to C \) return respectively the list of input colors and the unique output color of each node. We say a species is finite (aka "polynomial") iff both \( C \) and \( V \) are finite. A map of species is a pair of functions \((\phi_C, \phi_V)\) making the diagram commute:
The free / forgetful adjunction

Any operad has an **underlying species**, where $C$ is the set of colors and $V$ the set of operations, just forgetting about composition and identity.

Conversely, to any species $𝕊$ there is an associated **free operad** $\text{Free } 𝕊$.

$$\text{Species}(\text{Free } 𝕊, \emptyset) \cong \text{Operad}(𝕊, \text{Forget } \emptyset)$$
Definition

A context-free grammar of arrows is a tuple \(G = (\mathcal{C}, \mathcal{S}, S, \phi)\) consisting of a (finitely generated) category \(\mathcal{C}\), a finite species \(\mathcal{S}\) equipped with a distinguished color \(S \in \mathcal{S}\) and a functor of operads \(p : \text{Free } \mathcal{S} \to W[\mathcal{C}]\).

The context-free language of arrows \(L_G\) generated by the grammar \(G\) is the subset of arrows in \(\mathcal{C}\) which, seen as constants of \(W[\mathcal{C}]\), are in the image of constants of color \(S\) in \(\text{Free } \mathcal{S}\), that is, \(L_G = \{ p(\alpha) \mid \alpha : S \}\).

Proposition: A language \(L \subseteq \Sigma^*\) is context-free in the classical sense iff it is the language of arrows of a context-free grammar over \(\mathbb{B}_\Sigma\).
(Another look at the example)

1 : S → NP VP
2 : NP → mom
3 : NP → tom
4 : VP → loves NP

Free $S$

\[
\begin{align*}
\text{id}_\mathcal{U} \cdot \text{id} & \quad \text{mom} \\
\text{tom} & \quad \text{loves}_\mathcal{U} \cdot \text{id}
\end{align*}
\]

\[
\text{id}_\mathcal{U} \cdot \text{id} \circ (\text{tom}, \text{loves}_\mathcal{U} \cdot \text{id} \circ \text{mom}) = \text{tom}_\mathcal{U} \cdot \text{loves}_\mathcal{U} \cdot \text{mom}
\]
Refining classical CFGs with "gap types"

A feature of the general notion of CFG of arrows is that non-terminals are sorted. Adopting our conventions for type refinement, we sometimes write \( R \sqsubseteq (A,B) \) to indicate \( p(R) = (A,B) \) and say that \( R \) refines the gap type \( (A,B) \). The language generated by a grammar with start symbol \( S \sqsubseteq (A,B) \) is a subset of \( \mathbb{C}(A,B) \).

As a simple example, consider the category \( \mathbb{B}^T_\Sigma = \mathbb{B}_\Sigma + \sigma \) 1 constructed from \( \mathbb{B}_\Sigma \) by freely adjoining an object \( T \) and an arrow \( * : * \to T \). A CFG over \( \mathbb{B}^T_\Sigma \) may include production rules that can only be applied upon reaching end of input, like Knuth’s "0th production" rule \( S' \to SS$ \) from the original paper on LR parsing. (Here \( S \sqsubseteq (\ast,\ast) \) is "classical" and \( S' \sqsubseteq (\ast,T) \) is "end-of-input-aware".)

More significant examples coming up, including CFGs over runs of automata!
Reformulating standard properties of CFGs

Let $G = (\mathbb{C}, \mathbb{S}, S, p)$ be a CFG of arrows.

• $G$ is **linear** iff $\mathbb{S}$ only has nodes of arity $\leq 1$. It is **left-linear** iff it is linear and every unary node $x$ of $\mathbb{S}$ is mapped by $p$ to an operation of the form $\text{id} - w$.

• $G$ is **bilinear** (a generalization of Chomsky NF) iff $\mathbb{S}$ only has nodes of arity $\leq 2$.

• $G$ is **unambiguous** iff for any constants $\alpha, \beta : S$ in Free $\mathbb{S}$, if $p(\alpha) = p(\beta)$ then $\alpha = \beta$.

• A non-terminal $R$ is **nullable** if there exists a constant $\alpha : R$ of Free $\mathbb{S}$ s.t. $p(\alpha) = \text{id}$.

• A non-terminal $R$ is **useful** if there exists a constant $\alpha : R$ and a unary op $\beta : R \to S$. Note that if $G$ has no useless non-terminals then $G$ is unambiguous iff $p$ is faithful.
Basic closure properties of CF languages

[Union] If $L_1, L_2 \subseteq \mathbb{C}(A,B)$ are CF, so is $L_1 \cup L_2 \subseteq \mathbb{C}(A,B)$.

[Spliced concatenation] If $L_1 \subseteq \mathbb{C}(A_1,B_1),...,L_n \subseteq \mathbb{C}(A_n,B_n)$ are CF, and $w_0 - w_1 - \cdots - w_n : (A_1,B_1),...,(A_n,B_n) \rightarrow (A,B)$ is an operation of $W[\mathbb{C}]$, then $w_0L_1w_1\cdots L_nw_n \subseteq \mathbb{C}(A, B)$ is also CF.

[Functorial image] If $L \subseteq \mathbb{C}(A, B)$ is CF, and $F : \mathbb{C} \rightarrow \mathbb{D}$ is a functor of categories, then $F(L) \subseteq \mathbb{D}(F(A), F(B))$ is also CF.

(Proofs left as an exercise!)
The translation principle

Let $G_1 = (\mathcal{C}, S_1, S_1, p_1)$ and $G_2 = (\mathcal{C}, S_2, S_2, p_2)$ be two CFGs over the same category $\mathcal{C}$.

If there is a fully faithful functor of operads $T : \text{Free } S_1 \to \text{Free } S_2$ such that $p_1 = T p_2$ and $T(S_1) = S_2$, then $L_{G_1} = L_{G_2}$.

Example use of translation principle: for any CFG of arrows, there is a bilinear CFG of arrows generating the same language.
Parsing as a lifting problem

Besides characterizing the language generated by a grammar, we’re often interested in the dual problem of parsing. In our functorial formulation of context-free grammars, parsing an arrow \( w \) may be considered as the problem of computing its inverse image along \( p : \text{Free } S \to W[C] \).

One high-level tool for analyzing this problem is the correspondence between functors of categories \( p : \mathcal{D} \to \mathcal{T} \) and lax functors \( F : \mathcal{T} \to \text{Span}(\text{Set}) \) into the bicategory of spans of sets, which can be extended smoothly to functors of operads. Adapting terminology introduced by Ahrens and Lumsdaine, we refer to a lax functor of operads \( F : \mathcal{T} \to \text{Span}(\text{Set}) \) as a **displayed operad**.
Displayed free operads, and generalized CYK parsing

One can derive an inductive formula for displayed free operads, which refines the inductive formula for free operads $\text{Free } S \equiv I + S \circ \text{Free } S$ that characterizes the free operad over $S$ as a species of $S$-labelled trees.

Specializing the formula to the underlying functor of a CFG seen as a displayed operad $F : W[C] \to \text{Span}(\text{Set})$, we obtain a formula for the fiber $F_w$ of parse trees of any given arrow $w$. We can also derive an inductive rule for computing the set $N_w$ of non-terminals deriving $w$, which is essentially the rule given by Leermakers (1989) in his generalization of CYK parsing to arbitrary CFGs. As he explained, the relation $N_w$ can be solved in cubic time for bilinear grammars.

\[
\begin{align*}
\text{if } & (x : R_1, \ldots, R_k \to R) \in S \\
w &= w_0u_1w_1 \ldots u_kw_k \\
\phi(x) &= w_0 \cdot w_1 \cdot \ldots \cdot w_n \\
R_1 &\in N_{u_1} \quad \ldots \quad R_k &\in N_{u_k} \\
R &\in N_w
\end{align*}
\]
3. Finite-state automata over categories and operads
Reminder on finite state automata

An NDFA: [recognizing the language \((a+b)\)(abb+ba)]

\[
\text{alphabet } \Sigma = \{a,b\} \quad \text{state set } Q = \{0,1,2,3,4\} \\
\text{(no } \varepsilon \text{-transitions)}
\]
Representing automata as functors

\[ p \downarrow B_\Sigma \]

\[ \varnothing \]

\[ a \rightarrow b \rightarrow \]

\[ 0 \rightarrow 1 \rightarrow 2 \]

\[ 3 \rightarrow 4 \]

\[ a \circ \ast \rightarrow b \circ \]

\[ \mathfrak{B} \Sigma \]
Two key properties of NDFAs

Let \( p : \mathcal{D} \to \mathcal{T} \) be a functor of categories.

\( p \) is **finitary** if either of the following equivalent conditions hold:
\begin{itemize}
  \item the fibers \( p^{-1}(A) \) and \( p^{-1}(w) \) are finite for every object \( A \) and arrow \( w \) in \( \mathcal{T} \);
  \item the associated lax functor \( F : \mathcal{T} \to \text{Span(Set)} \) factors via \( \text{Span}(\text{FinSet}) \).
\end{itemize}

\( p \) is **ULF** if either of the following equivalent conditions hold:
\begin{itemize}
  \item for any arrow \( \alpha \) of \( \mathcal{D} \), if \( p(\alpha) = uv \) for some pair of arrows \( u \) and \( v \) of \( \mathcal{T} \), there exists a unique pair of arrows \( \beta \) and \( \gamma \) in \( \mathcal{D} \) such that \( \alpha = \beta \gamma \), \( p(\beta) = u \), \( p(\gamma) = v \).
  \item the associated lax functor \( F : \mathcal{T} \to \text{Span(Set)} \) is a pseudofunctor.
\end{itemize}

Proposition: a functor \( p : \mathcal{Q} \to \mathcal{B}_\Sigma \) is the underlying bare automaton of a NDFA with alphabet \( \Sigma \) iff \( p \) is both finitary and ULF.

ULF = "unique lifting of factorizations" (Lawvere & Meni)
A NDFA over a category is a tuple \( M = (\mathbb{C}, \mathbb{Q}, p : \mathbb{Q} \rightarrow \mathbb{C}, q_0, q_f) \) consisting of two categories \( \mathbb{C} \) and \( \mathbb{Q} \), a finitary ULF functor \( p : \mathbb{Q} \rightarrow \mathbb{C} \), and a pair \( q_0, q_f \) of objects of \( \mathbb{Q} \).

The regular language of arrows \( L_M \) recognized by the automaton \( M \) is the subset of arrows in \( \mathbb{C} \) that can be lifted along \( p \) to an arrow \( \alpha : q_0 \rightarrow q_f \) in \( \mathbb{Q} \), that is, \( L_M = \{ p(\alpha) \mid \alpha : q_0 \rightarrow q_f \} \).

Proposition: A language \( L \subseteq \Sigma^* \) is regular in the classical sense iff \( L$ \) is the regular language of arrows of a NDFA over \( \mathbb{B}_\Sigma^T \).
Automata over operads

The notions of finitary and ULF extend smoothly to functors of operads.

By analogy, an **NDFA over an operad** is a tuple $M = (\mathcal{O}, \mathcal{Q}, p : \mathcal{Q} \to \mathcal{O}, q)$ where $p : \mathcal{Q} \to \mathcal{O}$ is a finitary ULF functor of operads, and $q$ a color of $\mathcal{Q}$.

When $\mathcal{O}$ is a free operad, this recovers the standard notion of ND finite state tree automaton. But the notion of NDFA over an operad is more general!

Proposition: if a functor of categories $p : \mathcal{Q} \to \mathcal{C}$ is finitary or ULF, so is the functor of operads $W[p] : W[\mathcal{Q}] \to W[\mathcal{C}]$.

∴ any NDFA over a category induces an NDFA over its spliced arrow operad, by the mapping $(p : \mathcal{Q} \to \mathcal{C}, q_0, q_f) \mapsto (W[p] : W[\mathcal{Q}] \to W[\mathcal{C}], (q_0, q_f))$
4. The Representation Theorem
Overview

Chomsky & Schützenberger (1963): Any CF language is the homomorphic image of the intersection of a Dyck language with a regular language.

Our version: Any CF language of arrows in $\mathbb{C}$ is the functorial image of the intersection of a $\mathbb{C}$-chromatic tree contour language and a regular language.

The proof relies on two constructions that are of more general interest:

1. The pullback of a CFG of arrows along an NDFA, which we use to show that CF languages are closed under intersection with regular languages.

2. The contour category of an operad, providing a left adjoint to the splicing functor, which we use to define a "universal CFG" for any pointed finite species.
An important property of ULF functors

Let $p_Q : \mathbb{Q} \to \mathcal{O}$ be a ULF functor of operads. Then the pullback of $p : \text{Free } \mathcal{S} \to \mathcal{O}$ along $p_Q$ in the category of operads is obtained from a corresponding pullback of $\varphi : \mathcal{S} \to \mathcal{O}$ along $p_Q : \mathbb{Q} \to \mathcal{O}$ in Species.

\[
\begin{array}{ccc}
\text{Free } \mathcal{S}' & \xrightarrow{\text{Free } \psi} & \text{Free } \mathcal{S} \\
p' \downarrow & \mathclap{\text{ pullback }} & \downarrow p \\
\mathbb{Q} & \xrightarrow{p_Q} & \mathcal{O}
\end{array}
\quad
\begin{array}{ccc}
\mathcal{S}' & \xrightarrow{\psi} & \mathcal{S} \\
\varphi' \downarrow & \mathclap{\text{ pullback }} & \downarrow \varphi \\
\mathbb{Q} & \xrightarrow{p_Q} & \mathcal{O}
\end{array}
\]
By the previous result, we can compute the pullback on the right:

\[ \mathbb{S} \left[ \mathbb{C} \right] \rightarrow \mathbb{S} \left[ \mathbb{Q} \right] \]

The pullback of \( G \) along \( M \) is the grammar \( G' = (Q, S', (S, (q_0, q_f)), p') \). Note that \( G' \) generates a language of runs of \( M \)!

Taking the image of \( G' \) along \( p_M \) yields a grammar generating \( L_G \cap L_M \).
The contour category of an operad

Let $\mathcal{O}$ be an operad. The category $C[\mathcal{O}]$ is a quotient of the free category with:

- objects given by *oriented colors* $R^\varepsilon$ consisting of a color $R$ of $\mathcal{O}$ and an orientation $\varepsilon \in \{ \text{u,d} \}$ ("up" or "down");
- arrows 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 $R_i^d \to R_{i+1}^u$ where $R_0^d = R^u$ and $R_{n+1}^u = R^d$;

subject to the equations $\text{id}_{R^u} = (\text{id}_R,0)$ and $\text{id}_{R^d} = (\text{id}_R,1)$ plus the equations

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

and

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

for every operation $f$, operation $g$ of positive arity $m > 0$, and constant $c$. 
The contour category of an operad
The contour category of an operad

\[(f \circ_1 g, 2) = (g, 1)\]

\[(f \circ_1 g, 1) = (f, 1)\]

\[(g, 0) \quad \text{(g,0)}\]

\[(f, 2) \quad \text{(f,2)}\]

\[(f \circ_1 c, 1) = (f, 3)\]

\[(f \circ_1 c, 0) = (f, 0)\]

\[(f \circ_1 c, 2) = (f, 3)\]
The contour / splicing adjunction

This construction provides a left adjoint to the splicing construction:

\[
\begin{array}{ccc}
\text{Operad} & \xrightarrow{\text{C[-]}} & \text{Cat} \\
\downarrow & & \downarrow \\
W[-] & \xleftarrow{\text{W[-]}} & \\
\end{array}
\]

\[
\text{Operad}(\mathbb{O}, W[\mathbb{C}]) \cong \text{Cat}(C[\mathbb{O}], \mathbb{C})
\]

The unit and counit have nice descriptions:

\[
\eta : \mathbb{O} \to W[C[\mathbb{O}]] \quad \varepsilon : C[W[\mathbb{C}]] \to \mathbb{C}
\]

\[
R \mapsto (R^u, R^d) \quad (A, B)^u \mapsto A \quad (A, B)^d \mapsto B
\]

\[
f \mapsto (f, 0) \dashrightarrow \cdots \dashrightarrow (f, n) \quad (w_0 \cdots w_n, i) \mapsto w_i
\]
Free contour categories

The contour category of a free operad is itself a free category, with $C[\text{Free } S]$ generated by the **corners** $(x,i)$ consisting of an $n$-ary node $x$ and index $0 \leq i \leq n$.

We sometimes write $C[S]$ as another name for this category.

Although $C[-]$ does not preserve ULF in general, we have that for any species map $\psi : S \to \mathbb{R}$, the functor of categories $C[\psi] : C[S] \to C[\mathbb{R}]$ is ULF.

*Note that the word "corner" comes from the theory of planar maps, but in parsing theory, corners are called "dotted rules"!
The universal CFG of a pointed finite species

By the contour / splicing adjunction, any \( p : \text{Free } \mathcal{S} \rightarrow W[\mathbb{C}] \) factors as

\[
\text{Free } \mathcal{S} \xrightarrow{\eta_{\mathcal{S}}} W[C[\text{Free } \mathcal{S}]] \xrightarrow{W[q]} W[\mathbb{C}]
\]

for a unique functor of categories \( q : C[\text{Free } \mathcal{S}] \rightarrow \mathbb{C} \).

The CFG \( \text{Univ}_{\mathcal{S},\mathcal{S}} = (C[\text{Free } \mathcal{S}], \mathcal{S}, \mathcal{S}, \eta_{\mathcal{S}}) \) is therefore "universal", in the sense that any other CFG \( G = (\mathbb{C}, \mathcal{S}, \mathcal{S}, p) \) with the same species and start symbol is obtained uniquely as the functorial image \( G = q \text{ Univ}_{\mathcal{S},\mathcal{S}} \).

The language generated by \( \text{Univ}_{\mathcal{S},\mathcal{S}} \) is a language of tree contour words.
A tree contour word over a species $\mathcal{S}$

$$a_0 b_0 a_1 c_0 d_0 c_1 e_0 c_2 a_2 f_0 g_0 f_1 a_3 : 1^u \rightarrow 1^d$$
Idea of the representation theorem

Separate the generation of a CF language into three pieces:

1. generate "uncolored" contour words describing shapes of $\mathcal{S}$-trees;

2. use an automaton to check that the contour words denote well-colored $\mathcal{S}$-trees with root color $S$;

3. interpret each corner of the contour as an appropriate arrow.
Another basic fact about species

Any map of species $\phi : S \to \mathbb{R}$ factors as:

$$S \xrightarrow{\phi_{\text{colors}}} \phi_C S \xrightarrow{id \text{ on colors}} \mathbb{R}$$

In particular, we can apply this factorization to the underlying map of species $\phi : S \to W[C]$ of a given CFG of arrows.

The functor $C[\phi_{\text{colors}}] : C[S] \to C[\phi_C S]$ paired with the states $S^u$ and $S^d$ defines an automaton on contour words!
The proof in a diagram

\[
\begin{array}{ccc}
\text{Free } \mathcal{S} & \xrightarrow{\text{Free } \varphi_{\text{colors}}} & \text{Free } \varphi_{\mathcal{C}} \mathcal{S} \\
\eta_{\mathcal{S}} & \downarrow & \ \\
W[C[\text{Free } \mathcal{S}]] & \overset{W[C[\text{Free } \varphi_{\text{colors}}]]}{\longrightarrow} & W[C[\text{Free } \varphi_{\mathcal{C}} \mathcal{S}]] \\
\downarrow & & \downarrow \\
W[q] & \overset{W[q_{\text{nodes}}]}{\longrightarrow} & W[\mathcal{C}] \\
\end{array}
\]

\[
L_G = q L_{\mathcal{S},\mathcal{S}} = q_{\text{nodes}} C[\varphi_{\text{colors}}] L_{\mathcal{S},\mathcal{S}} = q_{\text{nodes}} (L_{\varphi_{\mathcal{C}}} \mathcal{S}, \mathcal{S} \cap L_{M_{\text{colors}}})
\]

*The naturality square is not a pullback, but the canonical functor Free \( \mathcal{S} \to \text{Free } \mathbb{R} \) to the pullback is fully faithful, hence we can apply the translation principle!
From contour words to Dyck words

\[ \text{Diagram showing relationships between contour and Dyck words.} \]
5. Example
Colors / nodes factorization

$\mathcal{S}$

$\varphi_{\text{colors}}$

$\varphi_{\text{nodes}}$

$W[\mathcal{B}_\Sigma]$

id\_u\_id  mom  tom  loves\_u\_id
Translation of corners

\[ \begin{align*}
1_0 & \mapsto \text{id} \\
1_1 & \mapsto \text{ } \\
1_2 & \mapsto \text{id} \\
2_0 & \mapsto \text{mom} \\
3_0 & \mapsto \text{tom} \\
4_0 & \mapsto \text{loves} \\
4_1 & \mapsto \text{id}
\end{align*} \]
Uncolored tree contour words

$1_03_01_14_02_04_11_2$

$1_03_01_12_01_2$

$4_01_02_11_3_01_24_1$

tom$\_\_\_loves_\_\_mom$

tom$\_\_\_mom$

loves$\_\_\_mom_\_\_tom$

...
Coloring automaton
6. Conclusion
Both CFGs and NDFAs may be naturally represented as functors, and generalized to define context-free / regular languages of arrows in a category.

Parsing may be naturally formulated as a lifting problem.

The Chomsky-Schützenberger Representation Theorem is deeply related to an elementary "contour / splicing" adjunction between operads and categories.

Are there other applications of spliced arrow operads and contour categories?

Next on our agenda: pushdown automata and LR parsing!