Context-free grammars and finite-state automata over categories

Noam Zeilberger[1]

Ecole Polytechnique (LIX)

Theoretical Computer Science Seminar
Birmingham, 1 December 2023

[1]Based on joint work with Paul-André Melliès, in particular our MFPS 2022 paper "Parsing as a lifting problem and the Chomsky-Schützenberger Representation Theorem", and a revised and significantly expanded version that will hopefully be online soon...

Back in time

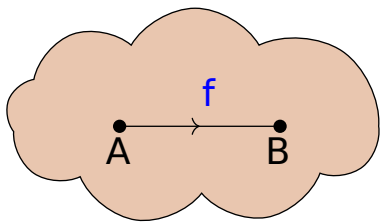
"Functors are Type Refinement Systems" (a manifesto, ca. 2010s)

Abstract

The standard reading of type theory through the lens of category theory is based on the idea of viewing a type system as a category of well-typed terms. We propose a basic revision of this reading: rather than interpreting type systems as categories, we describe them as *functors* from a category of typing derivations to a category of underlying terms. Then, turning this around, we explain how in fact *any* functor gives rise to a generalized type system, with an abstract notion of typing judgment, typing derivations and typing rules. This leads to a purely categorical reformulation of various natural classes of type systems as natural classes of functors.

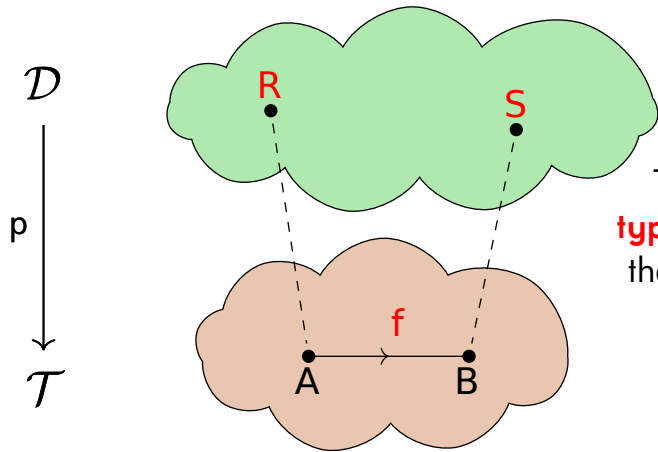
Typing as a lifting problem

$\mathcal{T}$



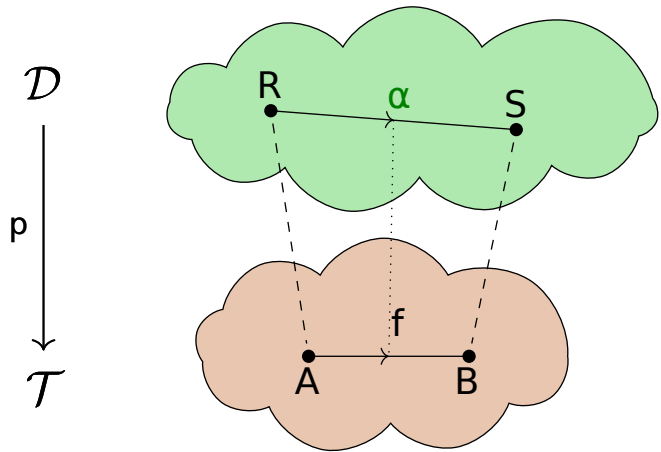
f is a **term** with
"intrinsic" type A → B

Typing as a lifting problem



The triple (R,f,S) form a **typing judgment**, asserting that f may be assigned an "extrinsic" type R → S

Typing as a lifting problem



α is a **typing derivation** providing evidence for the judgment

Beyond type theory

Our original motivations actually came from proof theory, and through chance circumstances realized a connection with type refinement systems.

In the initial paper series, we sketched how to apply this perspective to other deductive systems, including linear logic and separation logic.

From 2017, we tried to apply this perspective to parsing.[2]
Rough analogy: parsing = typing = lifting along a functor

Very slow progress until 2022, when we had the idea of analyzing the Chomsky-Schützenberger Representation Theorem (1963)

[2] Inspired in part by Hayo Thielecke's work on LL and LR!

The C-S Representation Theorem

A language is context-free iff it is a homomorphic image of the intersection of a Dyck language of balanced brackets with a regular language.

A couple reasons why it is interesting:

1. invokes a non-trivial closure property of CFLs (intersection with regular languages)
2. suggests that Dyck languages are in some sense "universal" CFLs

Can the theorem be analyzed categorically? (Spoiler alert: yes!)
Key ideas on two slides...

Idea #1: fibrational perspectives on grammars and automata

Two key definitions:

- a **generalized CFG** is a functor from a *free operad* (= multicategory) generated by a *pointed finite species* into an arbitrary *base operad*;
- a **generalized NDFA** is a functor satisfying the *unique lifting of factorizations* and *finite fiber* properties, from an arbitrary *bipointed category* or *pointed operad*.

This enables us to define generalized context-free and regular languages, and to establish suitable generalizations of many standard closure properties.

In particular, there is a nice proof that gCFLs are closed under intersection with gRLs.

Idea #2: a surprising adjunction between categories and operads

A functor $\mathcal{W} : \mathrm{Cat} \to \mathrm{Operad}$ we call the *operad of spliced arrows* construction enables us to define "context-free languages of arrows" (which include classical CFLs), and to turn any categorical NDFA into an operadic NDFA recognizing the same language.

Surprisingly, this functor admits a left adjoint:

$$\mathrm{Operad} \underset{\mathcal{W}}{\overset{\mathcal{C}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathrm{Cat}$$

The unit of the adjunction implies that any pointed finite species may be equipped with a *universal CFG* generating a language of "tree contour words".

Plan of the talk

1. Context-free languages of arrows in a category (and gCFGs)
2. An interlude on displayed categories and operads
3. Finite state automata over categories and operads
4. The Representation Theorem
5. Related work and ongoing/future directions

Context-free languages of arrows in a category

Background: operads[4]

An **operad** is like a category where arrows can take multiple objects as input.

$$f : A_1, \ldots, A_n \to B$$

We usually refer to objects A, B as "colors" and (multi-)arrows f as "operations".

Composition of operations has the type of the cut rule in sequent calculus, plus there is an identity operation for every color:

$$\frac{g : \Omega \to A \quad f : \Gamma, A, \Delta \to B}{f \circ_i g : \Gamma, \Omega, \Delta \to B}\ (i{=}|\Gamma|) \qquad \frac{}{id_A : A \to A}$$

These satisfy appropriate associativity, unitality, and exchange axioms.[3]

[3] Equivalently, operads can be defined using *parallel composition* (= multi-cut) $f \circ (g_1, \ldots, g_n)$, which should satisfy appropriate associativity and unitality axioms.

[4] Usage note: "operad" = colored non-symmetric operad = multicategory.

Background: species[5]

A **species** is a diagram of sets of the form

$$C^* \xleftarrow{\ i\ } V \xrightarrow{\ o\ } C$$

We write $f : A_1, \ldots, A_n \to B$ for $f \in V$ when $i(f) = (A_1, \ldots, A_n)$ and $o(f) = B$.

(Formally, a species is like an operad without composition and identity.)

We say a species is **finite** when both C and V are finite.

[5] Usage note: "species" = colored non-symmetric species = multigraph / signature.

Categories of species and operads

There is a category $\mathrm{Species}$ whose morphisms are commutative diagrams of the form:

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

meaning that a map of species $\phi : \mathbb{R} \to \mathbb{S}$ sends colors to colors and nodes to nodes, with $f : A_1, \ldots, A_n \to B$ in $\mathbb{R}$ sent to $\phi(f) : \phi(A_1), \ldots, \phi(A_n) \to \phi(B)$ in $\mathbb{S}$.

A *functor of operads* $F : \mathcal{P} \to \mathcal{O}$ is a map between the underlying species that respects composition and identity. These define the morphisms of a (2-)category Operad.

Free / forgetful adjunction

The forgetful functor from operads to species has a left adjoint:

$$\mathrm{Species} \underset{\text{Forget}}{\overset{\text{Free}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathrm{Operad}$$

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

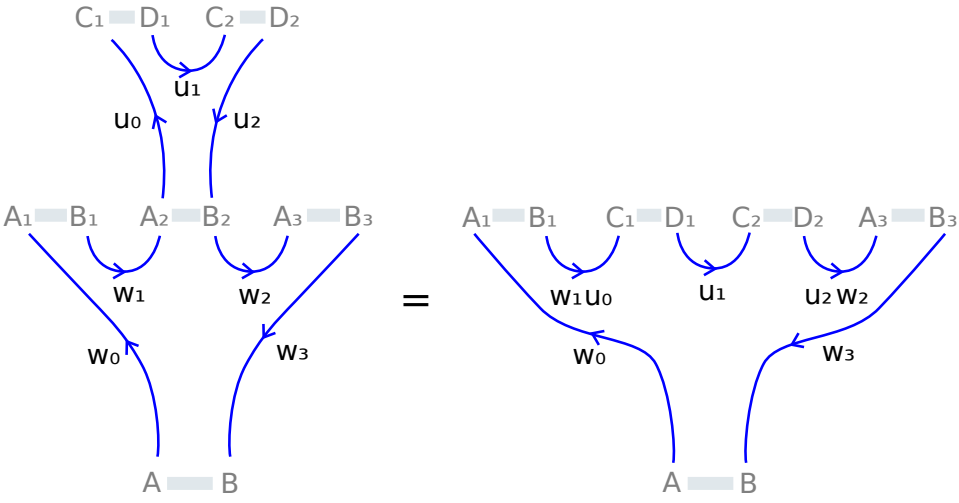
Operations of $\mathsf{Free}\,\mathbb{S}$ may be interpreted as rooted planar trees with colored edges whose nodes are drawn from the species $\mathbb{S}$, a.k.a. "$\mathbb{S}$-trees".

The splicing construction

For any category $\mathcal{C}$, the **operad of spliced arrows** $\mathcal{W}[\mathcal{C}]$ is defined as follows:

- colors are pairs (A, B) of objects of $\mathcal{C}$;
- operations $w_0 - w_1 - \ldots - w_n : (A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ consist of sequences of $n+1$ arrows in $\mathcal{C}$, where $w_i : B_i \to A_{i+1}$ for $0 \leqslant i \leqslant n$ under the convention that $B_0 = A$ and $A_{n+1} = B$;
- the identity operation on (A, B) is $id_A - id_B$;
- composition performed by "splicing into the gaps"...

The splicing construction



$$(w_0 - w_1 - w_2 - w_3) \circ_1 (u_0 - u_1 - u_2) = w_0 - w_1 u_0 - u_1 - u_2 w_2 - w_3$$

Context-free grammar over a category

A tuple $G = (\mathcal{C}, \mathbb{S}, S, p)$ consisting of:

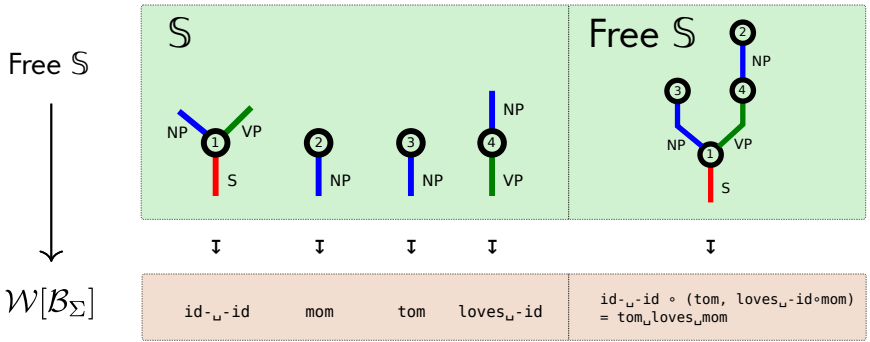
- a category $\mathcal{C}$
- a finite species $\mathbb{S}$
- a color $S \in \mathbb{S}$
- a functor $p : \mathrm{Free}\,\mathbb{S} \to \mathcal{W}[\mathcal{C}]$

The **CFL of arrows** generated by G is the sub-homset $L_G = \{\, p(\alpha) \mid \alpha : S \,\} \subseteq \mathcal{C}(A, B)$ where $p(S) = (A, B)$.

Observation: a classical CFG is just a CFG over $\mathcal{B}_\Sigma = *^{\circlearrowleft a \in \Sigma}$.

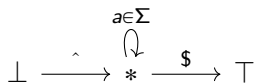
Representing CFGs by functors: an example

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



Categories as typed alphabets

Let $\mathcal{B}_{\Sigma}^{\hat{\$}}$ be obtained from $\mathcal{B}_{\Sigma}$ by freely adjoining a pair of objects and arrows:



A CFG over $\mathcal{B}_{\Sigma}^{\hat{\$}}$ may include non-terminals that can only be derived at the beginning/end of a string. This already arises in practice (e.g., in LR parsing) but is usually treated in an ad hoc way.

Properties of CFGs over categories

Let $G = (\mathcal{C}, \mathbb{S}, S, p)$ be a CFG. We can neatly express many standard properties:

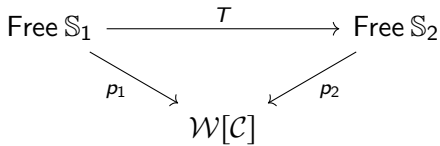
- G is *linear* just in case $\mathbb{S}$ only has nodes of arity $\leqslant 1$.
- G is *bilinear* just in case $\mathbb{S}$ only has nodes of arity $\leqslant 2$.
- G is *unambiguous* iff for any pair of constants $\alpha, \beta : S$ in Free $\mathbb{S}$, if $p(\alpha) = p(\beta)$ then $\alpha = \beta$. (p faithful $\Rightarrow$ G unambiguous.)
- A non-terminal R is *nullable* if there exists $\alpha : R$ in Free $\mathbb{S}$ such that $p(\alpha) = id$.
- A non-terminal R is *useful* if there exist $\alpha : R$ and $\beta : R \to S$ in Free $\mathbb{S}$. (G has no useless non-terminals $\Rightarrow$ [p faithful $\iff$ G unambiguous].)

Closure properties of CFLs of arrows

1. If $L_1, \ldots, L_k \subseteq \mathcal{C}(A, B)$ are CFLs of arrows, so is their union $\bigcup_{i=1}^{k} L_i \subseteq \mathcal{C}(A, B)$.
2. If $L_1 \subseteq \mathcal{C}(A_1, B_1), \ldots, L_n \subseteq \mathcal{C}(A_n, B_n)$ are CFLs of arrows, and if $w_0 - \ldots - w_n : (A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ is an operation of $\mathcal{W}[\mathcal{C}]$, then $w_0 L_1 w_1 \ldots L_n w_n = \{\, w_0 u_1 w_1 \ldots u_n w_n \mid u_1 \in L_1, \ldots, u_n \in L_n \,\} \subseteq \mathcal{C}(A, B)$ is a CFL.
3. If $L \subseteq \mathcal{C}(A, B)$ is a CFL of arrows in a category $\mathcal{C}$ and $F : \mathcal{C} \to \mathcal{D}$ is a functor of categories, then the functorial image $F(L) \subseteq \mathcal{D}(F(A), F(B))$ is also context-free.

Translation principle

Let $G_1 = (\mathcal{C}, \mathbb{S}_1, S_1, p_1)$ and $G_2 = (\mathcal{C}, \mathbb{S}_2, S_2, p_2)$ be two CFGs over the same category, and suppose there is a fully faithful functor T commuting with the projection functors



and preserving the start symbol $T(S_1) = S_2$. Then $L_{G_1} = L_{G_2}$, moreover with the grammars generating isomorphic sets of parse trees for the arrows in the language.

Example application: any CFL is generated isomorphically by a bilinear CFG.

Generalized context-free grammars

For many purposes, a functor $p : \mathrm{Free}\,\mathbb{S} \to \mathcal{O}$ from a free operad generated by a pointed finite species may be considered as a "generalized" CFG, inducing a language of constants in $\mathcal{O}$.

Choosing different base operads $\mathcal{O}$ yields different interesting examples.

- $\mathcal{O} = \mathcal{W}[\mathcal{B}_\Sigma]$: classical CFGs
- $\mathcal{O} = \mathcal{W}[\mathcal{C}]$: CFGs over categories
- $\mathcal{O} = !_{\mathsf{aff}}\mathcal{W}[\mathcal{C}]$: *multiple CFGs* (Seki et al. 1991, for $\mathcal{C} = \mathcal{B}_\Sigma$)
- $\mathcal{O} = !_{\mathsf{cart}}\mathcal{W}[\mathcal{C}]$: *parallel multiple CFGs* (Seki et al.)
- $\mathcal{O} = \mathrm{Set}$: more "semantic" examples, e.g., the gCFL of series-parallel graphs

Closure properties extend naturally to gCFLs.

An interlude on displayed categories and operads

Parsing as a lifting problem

Given a word w, we often want to compute the set of all its parse trees.

This amounts to computing the inverse image $p^{-1}(w) = \{\, \alpha \mid p(\alpha) = w \,\}$.

To better understand this view of parsing as a lifting problem, it is helpful to first recall the correspondence between functors $p : \mathcal{D} \to \mathcal{C}$ and lax functors $F : \mathcal{C} \to \mathrm{Span}(\mathrm{Set})$.

From a functor $p : \mathcal{D} \to \mathcal{C}$ to a lax functor $F : \mathcal{C} \to \mathrm{Span}(\mathrm{Set})$

For every arrow $w : A \to B$ of $\mathcal{C}$, taking fibers $F = p^{-1}$ defines a span

$$F(A) \longleftarrow F(w) \longrightarrow F(B)$$

that sends an arrow $\alpha : R \to S$ over w to the objects R and S lying over A and B.

Moreover, given $u : A \to B$ and $v : B \to C$ in $\mathcal{C}$, there is a morphism of spans $F(u)F(v) \Rightarrow F(uv)$ realized by the mapping

$$\begin{array}{ccc} \mathcal{D} & R \xrightarrow{\alpha} S \xrightarrow{\beta} T & \\ \downarrow p & & \mapsto \\ \mathcal{C} & A \xrightarrow{u} B \xrightarrow{v} C & \end{array} \qquad \begin{array}{c} R \xrightarrow{\alpha\beta} T \\ \\ A \xrightarrow{uv} C \end{array}$$

Similarly, $id_{F(A)} \Rightarrow F(id_A)$. However, in general these morphisms are *not* invertible.

From a lax functor $F : \mathcal{C} \to \mathrm{Span}(\mathrm{Set})$ to a functor $p : \mathcal{D} \to \mathcal{C}$

A variant of the Grothendieck construction (à la Bénabou).

The category $\mathcal{D} = \int F$ has:

- objects given by pairs (A, R) of an object A in $\mathcal{C}$ and an element $R \in F(A)$;
- arrows $(w, \alpha) : (A, R) \to (B, S)$ given by pairs of an arrow $w : A \to B$ in $\mathcal{C}$ and an element $\alpha \in F(w)$ mapped to (R, S) by the span $F(A) \leftarrow F(w) \to F(B)$;
- composition and identity are defined using the lax structure;
- $\pi : \int F \to \mathcal{C}$ is the projection functor.

Displayed categories and operads

This correspondence adapts smoothly to one between functors of operads $p : \mathcal{P} \to \mathcal{O}$ and lax functors $F : \mathcal{O} \to \mathrm{Span}(\mathrm{Set})$, viewing $\mathrm{Span}(\mathrm{Set})$ as a 2-categorical operad whose n-ary operations $X_1, \ldots, X_n \nrightarrow Y$ are spans $X_1 \times \cdots \times X_n \leftarrow S \to Y$.

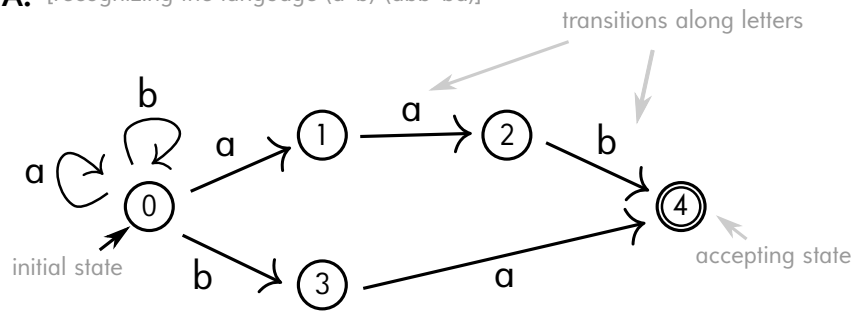
Ahrens and Lumsdaine introduced the terminology **displayed category** for the data of a lax functor $\mathcal{C} \to \mathrm{Span}(\mathrm{Set})$ presenting a category $\mathcal{D}$ living over $\mathcal{C}$, and similarly we can speak of **displayed operads**.

Any gCFG over $\mathcal{O}$ induces a corresponding displayed free operad $\mathcal{O} \to \mathrm{Span}(\mathrm{Set})$.

Finite state automata over categories and operads

Reminder on finite state automata

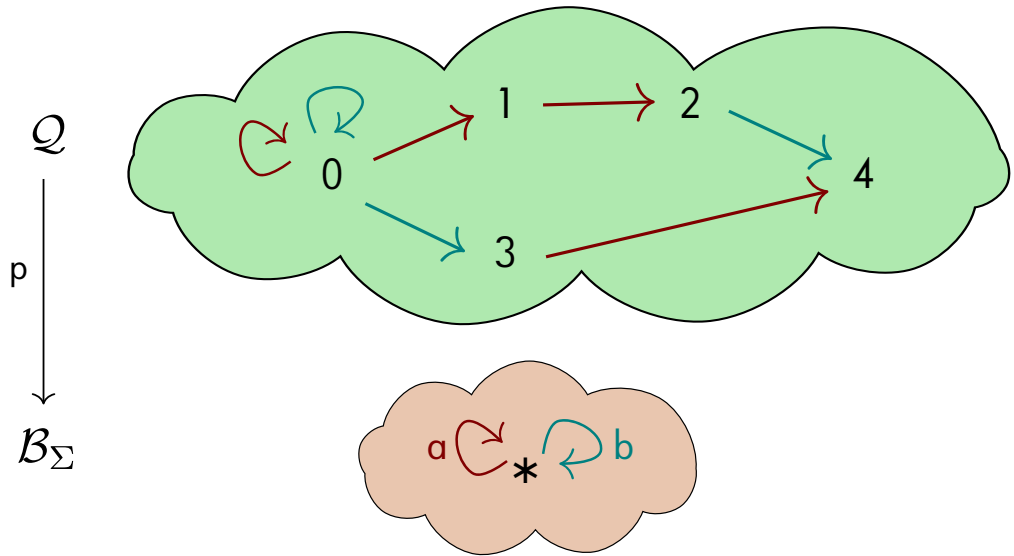
An **NDFA:** [recognizing the language (a+b)*(abb+ba)]



alphabet $\Sigma = \{a,b\}$ ***state set*** $Q = \{0,1,2,3,4\}$

(no ε-transitions)

Representing the "bare" NDFA by a functor



Unique lifting of factorizations property

A functor $p : \mathcal{D} \to \mathcal{C}$ is **ULF** if either of the following equivalent conditions hold:

- for every arrow α of $\mathcal{D}$, if $p(\alpha) = uv$ for some arrows u and v of $\mathcal{C}$, there exists a unique pair of arrows β and γ in $\mathcal{D}$ such that $\alpha = \beta\gamma$ and $p(\beta) = u$ and $p(\gamma) = v$;
- the structure maps of the associated lax functor $F : \mathcal{C} \to \mathrm{Span}(\mathrm{Set})$ are invertible, i.e., it is a pseudofunctor.

Proposition. Let $p : \mathcal{D} \to \mathcal{C}$ be a functor into a category $\mathcal{C} = \mathrm{Free}\,\mathbb{G}$ freely generated by some graph $\mathbb{G}$. Then p is ULF iff $\mathcal{D} = \mathrm{Free}\,\mathbb{H}$ is free over some graph $\mathbb{H}$ and $p = \mathrm{Free}\,\phi$ is generated by a graph homomorphism $\phi : \mathbb{H} \to \mathbb{G}$.

Finite fiber property

A functor $p : \mathcal{D} \to \mathcal{C}$ 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 $\mathcal{C}$;
- the associated lax functor $F : \mathcal{C} \to \mathrm{Span}(\mathrm{Set})$ factors via $\mathrm{Span}(\mathrm{FinSet})$.

Proposition. Let $\phi : \mathbb{H} \to \mathbb{G}$ be a homomorphism into a finite graph $\mathbb{G}$. Then $\mathrm{Free}\,\phi : \mathrm{Free}\,\mathbb{H} \to \mathrm{Free}\,\mathbb{G}$ is finitary iff $\mathbb{H}$ is finite.

Corollary. A functor $p : \mathcal{Q} \to \mathcal{B}_\Sigma$ is a bare NDFA iff p is ULF and finitary.

Non-deterministic finite-state automaton over a category

A tuple $M = (\mathcal{C}, \mathcal{Q}, p : \mathcal{Q} \to \mathcal{C}, q_0, q_f)$ consisting of:

- two categories $\mathcal{C}$ and $\mathcal{Q}$;
- a finitary ULF functor $p : \mathcal{Q} \to \mathcal{C}$;
- a pair q_0, q_f of objects of $\mathcal{Q}$

The **regular language of arrows** L_M recognized by M is the sub-homset

$$L_M = \{\, p(\alpha) \mid \alpha : q_0 \to q_f \,\} \subseteq \mathcal{C}(A, B)$$

where $p(q_0) = A, p(q_f) = B$.

Relationship to classical automata

Proposition. A language $L \subseteq \Sigma^*$ is regular in the classical sense if and only if $\hat{}L\$$ is the language of arrows recognized by a NDFA over $\mathcal{B}_\Sigma^{\hat{}\$}$.

An NDFA $M = (\mathcal{C}, \mathcal{Q}, p, q_0, q_f)$ is:

- **deterministic** if p is a discrete opfibration;
- **codeterministic** if p is a discrete fibration;
- **bideterministic** if p is a discrete bifibration;
- a **partial** (co/bi)deterministic automaton if p is a partial discrete op/bi/fibration.

ϵ-transitions are naturally modelled as arrows $\alpha : q \to q'$ such that $p(\alpha) = id$, but the ULF property implies they don't exist! (Need a more relaxed notion of ULF...)

Examples of NDFA over non-free categories

Product automaton: The product of two finitary ULF functors $p : \mathcal{Q} \to \mathcal{C}$ and $p' : \mathcal{Q}' \to \mathcal{C}'$ is again a finitary ULF functor $p \times p' : \mathcal{Q} \times \mathcal{Q}' \to \mathcal{C} \times \mathcal{C}'$. Hence, given an NDFA M over $\mathcal{C}$ and an NDFA M' over $\mathcal{C}'$ there is a NDFA $M \times M'$ over $\mathcal{C} \times \mathcal{C}'$, recognizing $L_{M \times M'} = L_M \times L_{M'}$.

Singleton automaton: For any arrow $w : A \to B$ of a category $\mathcal{C}$, there is a *category of factorizations* Fact_w whose objects are pairs of arrows (u, v) such that $w = uv$ and whose morphisms $(u, v) \to (u', v')$ are arrows x such that $u' = ux$ and $v = xv'$. The forgetful functor $\mu_w : \mathrm{Fact}_w \to \mathcal{C}$ is always ULF, but not necessarily finitary. When it is, $M_w = (\mathcal{C}, \mathrm{Fact}_w, \mu_w, (id_A, w), (w, id_B))$ defines an NDFA such that $L_{M_w} = \{ w \}$.

Total automaton: The identity functor $id_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ is trivially ULF and finitary, hence $M_{\mathcal{C}(A,B)} = (\mathcal{C}, \mathcal{C}, id_{\mathcal{C}}, A, B)$ is an NDFA recognizing the total homset $\mathcal{C}(A, B)$.

NDFA over an operad

A tuple $M = (\mathcal{Q}, \mathcal{O}, p : \mathcal{Q} \to \mathcal{O}, q_r)$ consisting of:

- two operads $\mathcal{O}$ and $\mathcal{Q}$;
- a finitary ULF functor $p : \mathcal{Q} \to \mathcal{O}$ (in the expected operadic sense);
- a color q_r of $\mathcal{Q}$

The **regular language of constants** L_M recognized by M is the sub-conset

$$L_M = \{\, p(\alpha) \mid \alpha : q_r \,\} \subseteq \mathcal{C}(A, B)$$

where $p(q_0) = A, p(q_f) = B$.

Operadic NDFA as a unifying concept

Two key examples:

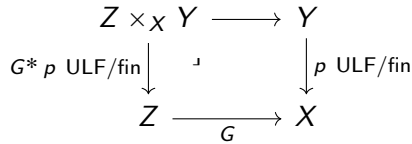
1. Let Σ be a graded alphabet, which we can see as a (uncolored) species.
   A non-deterministic tree automaton on Σ is just an NDFA over Free Σ!
2. The splicing functor $\mathcal{W}[-] : \mathrm{Cat} \to \mathrm{Operad}$ preserves finitary-ness and ULF-ness.
   Hence an NDFA on a category induces an NDFA over its spliced arrows operad

$$M = (\mathcal{C}, \mathcal{Q}, p, q_0, q_f) \qquad \mapsto \qquad \mathcal{W}[M] = (\mathcal{W}[\mathcal{C}], \mathcal{W}[\mathcal{Q}], \mathcal{W}[p], (q_0, q_f))$$

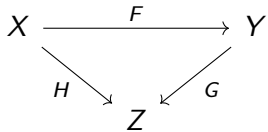
   generating the same language $L_{\mathcal{W}[M]} = L_M$!

Additional properties of ULF and finitary functors (of categories or operads)

Both ULF and finitary are closed under pullback along arbitrary functors.



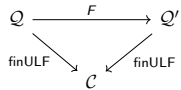
Consider a commutative triangle of functors:

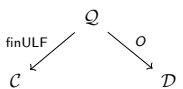


- if F and G are ULF/finitary, then so is H
- if G and H are ULF then so is F; if H is finitary then so is F

A quick aside on (bi)categories of automata

$\mathrm{finULF}/\mathcal{C}$ = category of bare NDFA and **functional simulations**



A span  defines a **ND finite state transducer** $\mathcal{C} \nrightarrow \mathcal{D}$,

associating to every run $\alpha : q_0 \to q_f$ in $\mathcal{Q}$ over an arrow $w : A \to B$ of $\mathcal{C}$ a corresponding output arrow $O(\alpha) : O(q_0) \to O(q_f)$ in $\mathcal{D}$.

By the aforementioned properties, transducers compose via ordinary span composition.

Closure properties of regular languages

Regular languages of arrows enjoy generalizations of standard closure properties:[6]

1. If $L \subseteq \mathcal{C}(A, B)$ is regular then $F^{-1}(L) \cap \mathcal{D}(R, S)$ is regular, for any functor $F : \mathcal{D} \to \mathcal{C}$ and objects R, S such that $F(R) = A, F(S) = B$.
2. If $L \subseteq \mathcal{C}(A, B)$ is regular and $F : \mathcal{C} \to \mathcal{D}$ is a finitary ULF functor then $F(L) \subseteq \mathcal{D}(F(A), F(B))$ is regular.

Similarly for regular languages of constants.

The restriction to finitary ULF functors in (2) really is necessary.

[6] We expect more standard closure properties to hold (e.g., union, spliced concatenation), but ϵ-transitions would help for the proofs...

The Representation Theorem

Classic version

A language is context-free iff it is a homomorphic image of the intersection of a language of balanced brackets with a regular language.

Proof separates the generation of a context-free language L_G in three pieces.

1. A CFG with only one non-terminal generating Dyck words over an alphabet $\Sigma_{2n} = \{ [_1,]_1, \ldots, [_n,]_n \}$, describing the *shapes* of parse trees in G;
2. An NDFA testing that the words encoding trees may be appropriately colored by the non-terminals of G according to its productions rules;
3. A homomorphism $\Sigma_{2n}^* \to \Sigma^*$ that interprets each bracket by a word in the original alphabet, with a choice to either interpret open or close brackets as empty words.

Our version for CFLs of arrows

We say $G = (\mathcal{C}, \mathbb{S}, p, S)$ is $\mathcal{C}$-**chromatic** when $p : \mathrm{Free}\,\mathbb{S} \to \mathcal{W}[\mathcal{C}]$ is injective on colors. (So non-terminals may be considered as pairs (A, B) of objects of $\mathcal{C}$.)

Theorem: $L \subseteq \mathcal{C}(A, B)$ is context-free iff it is a functorial image of the intersection of a $\mathcal{C}$-chromatic tree contour language with a regular language.

Tree contour languages arise from the contour / splicing adjunction, and as we will see their use removes any non-canonical choices from the proof.

But first we need to explain how to take the intersection of a CFL with a RL.

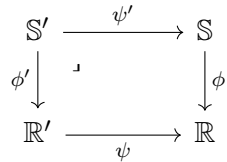
Pulling back a CFG along a NDFA

Lemma. Let $\mathbb{S}$ be a species, $p : \mathsf{Free}\,\mathbb{S} \to \mathcal{O}$ a functor of operads, and $p_{\mathcal{Q}} : \mathcal{Q} \to \mathcal{O}$ a ULF functor. Then the Operad-pullback of p along $p_{\mathcal{Q}}$ is obtained from a corresponding $\mathrm{Species}$-pullback of $\phi = p \circ \eta : \mathbb{S} \to \mathcal{O}$ along $p_{\mathcal{Q}} : \mathcal{Q} \to \mathcal{O}$:

$$\begin{array}{ccc} \mathbb{S}' & \xrightarrow{\psi'} & \mathbb{S} \\ {\scriptstyle \phi' = p_{\mathcal{Q}}^{*}\,\phi}\downarrow & \lrcorner & \downarrow{\scriptstyle \phi} \\ \mathcal{Q} & \xrightarrow[p_{\mathcal{Q}}]{} & \mathcal{O} \end{array} \in \mathrm{Species} \qquad \begin{array}{ccc} \mathsf{Free}\,\mathbb{S}' & \xrightarrow{\mathsf{Free}\,\psi'} & \mathsf{Free}\,\mathbb{S} \\ {\scriptstyle p' = p_{\mathcal{Q}}^{*}\,p}\downarrow & \lrcorner & \downarrow{\scriptstyle p} \\ \mathcal{Q} & \xrightarrow[p_{\mathcal{Q}}]{} & \mathcal{O} \end{array} \in \mathrm{Operad}$$

Pulling back a CFG along a NDFA

Lemma. Consider a pullback in the category of species:



If $\mathbb{S}$ is finite and ψ is finitary (i.e., has finite fibers) then $\mathbb{S}'$ is finite.

Pulling back a CFG along a NDFA

Theorem. For any CFG $G = (\mathcal{C}, \mathbb{S}, S, p_G)$ and NDFA $M = (\mathcal{C}, \mathcal{Q}, p_M, q_0, q_f)$ over the same category, with $p_G(S) = (p_M(q_0), p_M(q_f))$, there is a CFG $G' = M^* \, G$ generating the language $L_{G'} = p_M^{-1}(L_G) \cap \mathcal{Q}(q_0, q_f)$.

Proof: build the diagram on the right.

$$\begin{array}{ccc} \mathbb{S}' & \xrightarrow{\psi'} & \mathbb{S} \\ {\scriptstyle \phi'_G}\downarrow & \lrcorner & \downarrow{\scriptstyle \phi_G} \\ \mathcal{W}[\mathcal{Q}] & \xrightarrow[\mathcal{W}[p_M]]{} & \mathcal{W}[\mathcal{C}] \end{array} \in \mathrm{Species} \qquad \begin{array}{ccc} \mathsf{Free}\,\mathbb{S}' & \xrightarrow{\mathsf{Free}\,\psi'} & \mathsf{Free}\,\mathbb{S} \\ {\scriptstyle p'_G}\downarrow & \lrcorner & \downarrow{\scriptstyle p_G} \\ \mathcal{W}[\mathcal{Q}] & \xrightarrow[\mathcal{W}[p_M]]{} & \mathcal{W}[\mathcal{C}] \end{array} \in \mathrm{Operad}$$

Interpretation: G' generates a context-free language of runs of M!

Corollary. $L_G \cap L_M = \mathcal{W}[p_M](L_{G'})$

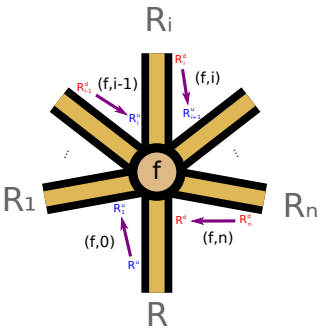
The contour construction

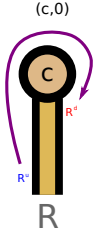
The spliced arrow functor $\mathcal{W} : \mathrm{Cat} \to \mathrm{Operad}$ admits a left adjoint:

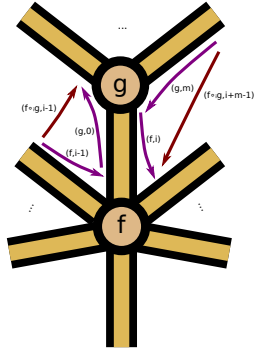
$$\mathrm{Operad} \underset{\mathcal{W}}{\overset{\mathcal{C}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathrm{Cat}$$

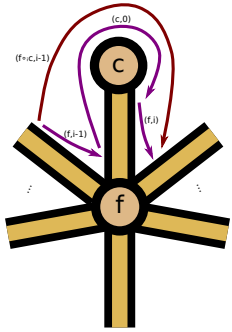
Given an operad $\mathcal{O}$, the **contour category** $\mathcal{C}[\mathcal{O}]$ has *oriented colors* as objects, and arrows generated by the *sectors of operations*, modulo some equations...

The contour construction









The contour / splicing adjunction

Natural correspondence:

$$\mathcal{O} \xrightarrow{F} \mathcal{W}[\mathcal{C}] \qquad \Longleftrightarrow \qquad \mathcal{C}[\mathcal{O}] \xrightarrow{G} \mathcal{C}$$

Unit and counit:

$$\begin{array}{ccc} \mathcal{O} & \xrightarrow{\eta_{\mathcal{O}}} & \mathcal{W}[\mathcal{C}[\mathcal{O}]] \\ R & \mapsto & (R^u, R^d) \\ f & \mapsto & (f,0) - \ldots - (f,n) \end{array} \qquad \begin{array}{rcl} \mathcal{C}[\mathcal{W}[\mathcal{C}]] & \xrightarrow{\varepsilon_{\mathcal{C}}} & \mathcal{C} \\ (A,B)^u & \mapsto & A \\ (A,B)^b & \mapsto & B \\ (w_0 - \ldots - w_n, i) & \mapsto & w_i \end{array}$$

Tree contour languages

To any pointed finite species $(\mathbb{S}, S)$ is associated a *universal CFG*

$$\mathrm{Univ}_{\mathbb{S},S} = (\mathcal{C}[\mathrm{Free}\,\mathbb{S}], \mathbb{S}, S, p_{\mathbb{S}} = \eta_{\mathrm{Free}\,\mathbb{S}})$$

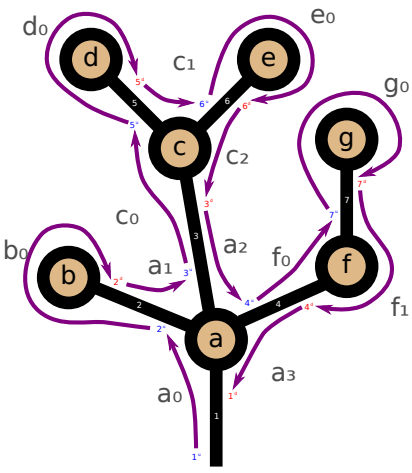
in the sense that for any other CFG of the form $G = (\mathcal{C}, \mathbb{S}, S, p_G)$ we have

$$\mathrm{Free}\,\mathbb{S} \xrightarrow{p_G} \mathcal{W}[\mathcal{C}] \quad = \quad \mathrm{Free}\,\mathbb{S} \xrightarrow{p_{\mathbb{S}}} \mathcal{W}[\mathcal{C}[\mathrm{Free}\,\mathbb{S}]] \xrightarrow{\mathcal{W}[q_G]} \mathcal{W}[\mathcal{C}]$$

for q_G derived uniquely from the adjunction, and hence $L_G = q_G\, L_{\mathrm{Univ}_{\mathbb{S},S}}$.

We write $TC_{\mathbb{S},S} = L_{\mathrm{Univ}_{\mathbb{S},S}}$ and say that it is a language of **tree contour words**.

Tree contour languages



$$a_0 b_0 a_1 c_0 d_0 c_1 e_0 c_2 a_2 f_0 g_0 f_1 a_3 : 1^u \to 1^d$$

Last steps of the proof

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

$$\mathbb{S} \xrightarrow{\ \phi\ } \mathbb{R} \quad = \quad \mathbb{S} \xrightarrow{\ \phi_{\text{colors}}\ } \phi_C\,\mathbb{S} \xrightarrow{\ \phi_{\text{nodes}}\ } \mathbb{R}$$

where ϕ_{colors} is id-on-nodes and surjective on colors and ϕ_{nodes} is injective on colors.

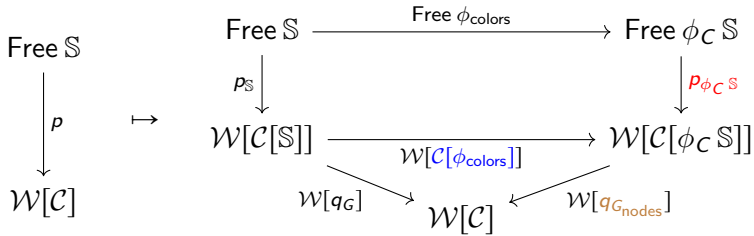
Lemma. Every map of species $\psi : \mathbb{S} \to \mathbb{S}'$ injective on nodes induces a diagram

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

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

Last steps of the proof

Build the diagram



and conclude (with an application of the Translation Principle) that

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

Related work and future directions

Related work (selected)

On modelling CFGs:

- R.F.C. Walters (1989), "A note on context-free languages"
- P. de Groote (2001), "Towards abstract categorial grammars"

On modelling NDFAs:

- B. Steinberg (2001), "Finite state automata: a geometric approach"
- P. Sobociński (2015), "Relational presheaves as labelled transition systems"
- T. Colcombet, D. Petrișan (2020), "Aut. minimization: a functorial approach"

On the contour / splicing adjunction:

- M. Earnshaw, J. Hefford, M. Román (2023), "The Produoidal Algebra of Process Decomposition" + "Contouring Prostar Autonomous Categories", who also cite B. Day and R. Street (2003), "Quantum categories, star autonomy, ..."

Ongoing and future directions

On the CFG side:

- CYK parsing (briefly discussed in MFPS2022 paper)
- Pushdown automata and LR/Earley parsing
- CFLs as initial models of CFGs

On the NDFA side:

- ϵ-transitions (and Kleene's theorem for a restricted class of categories?)
- automata over ω-words, asynchronous automata

On the contour / splicing side:

- deeper study of the adjunction with Peter Faul, who noticed ε is an equivalence!
- planar maps as automata over contour categories?