Parsing as a lifting problem

and the

Chomsky-Schützenberger Representation Theorem

Paul-André Melliès Noam Zeilberger

based on a paper to be presented at MFPS 2022
preliminary version: hal-03702762 (your comments are welcome!)

Structure Meets Power 2022 ⚘ Paris ⚘ 3 July 2022

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

$$p \quad : \quad \mathbb{D} \longrightarrow \mathbb{T}$$

- ▷ from a category $\mathbb{D}$ whose maps are **typing derivations**
- ▷ to a category $\mathbb{T}$ whose maps are the **terms** of those derivations.

A functorial view of type systems

Consider the **free cartesian closed category**

$$\mathbb{D} \quad = \quad \mathtt{freeccc}(\mathbb{o}, \mathbb{p}, \mathbb{q})$$

generated by atoms $\mathbb{o}$, $\mathbb{p}$, $\mathbb{q}$ with the cartesian closed functor

$$p \quad : \quad \mathbb{D} \longrightarrow \mathbb{T}$$

which maps $\mathbb{D}$ to the **syntactic cartesian closed category** $\mathbb{T}$

- ▷ whose objects are the natural numbers $m, n \in \mathbb{N}$,
- ▷ whose maps $m \to n$ are the families $M_1, \cdots, M_m$ of pure λ-terms

$$x_1 \, , \, x_2 \, , \, \cdots \, , \, x_m \quad \vdash \quad M_k \qquad 1 \leq k \leq n$$

with free variables among the m variables $x_1, \ldots, x_m$.

Typing as a lifting problem

Suppose given a pure λ-term

$$x_1, x_2, \cdots, x_m \quad \vdash \quad M$$

defining a map $M : \underbrace{1 + \cdots + 1}_{m} \to 1$ in the syntactic category $\mathbb{T}$.

A **typing judgment** in the usual sense

$$x_1 : \sigma_1, x_2 : \sigma_2, \ldots, x_m : \sigma_m \quad \vdash \quad M : \tau$$

is the same thing as a triple (R, M, S) with two objects of the category $\mathbb{D}$

$$R = \sigma_1, \sigma_2, \ldots, \sigma_m \qquad\qquad S = \tau$$

indicating the **types** of the variables and of the term M, and thus:

$$p(R) = 1 + \cdots + 1 \quad \text{and} \quad p(S) = 1.$$

Typing as a lifting problem

More generally, suppose given a term of the form

$$A \xrightarrow{f} B$$

with **intrinsic types** A and B.

Definition. A **typing judgment** is a triple (R, f, S) of the form:

$$\begin{tikzcd} R \arrow[d, dash, dashed] & S \arrow[d, dash, dashed] \\ A \arrow[r, "f"] & B \end{tikzcd}$$

where R and S are **extrinsic types** in $\mathbb{D}$ of the term $f : A \to B$ in $\mathbb{T}$.

Typing as a lifting problem

Given a **typing judgment** of the form

$$\begin{tikzcd} R \arrow[d, dash, dashed] & S \arrow[d, dash, dashed] \\ A \arrow[r, "f"] & B \end{tikzcd}$$

a **typing derivation** is a map $\alpha : R \to S$ in the category $\mathbb{D}$

$$\begin{tikzcd} R \arrow[r, "\alpha"] \arrow[d, dash, dashed] & S \arrow[d, dash, dashed] \\ A \arrow[r, "f"] & B \end{tikzcd}$$

whose image $p(\alpha)$ is equal to the term $f : A \to B$ in the category $\mathbb{T}$.

Word recognition as a lifting problem

Interestingly, the very same pattern appears in automata theory.

Suppose given an alphabet Σ.

Every non-deterministic finite state automaton induces a category $\mathbb{Q}$

- whose objects are the **states of the automaton**
- whose maps are the **runs (= transition paths) of the automaton**

freely generated by the transition graph of the automaton.

Word recognition as a lifting problem

The automaton on the alphabet Σ induces a **labelling functor**

$$p \quad : \quad \mathbb{Q} \longrightarrow \mathbb{B}_\Sigma$$

where $\mathbb{B}_\Sigma$ is the category with a unique object $*$ and one map

$$w \quad : \quad * \longrightarrow *$$

for each finite word w on the alphabet Σ.

Basic idea: The functor p transports each transition of the automaton

$$q \xrightarrow{\ a\ } q'$$

to the underlying letter $a \in \Sigma$ of the alphabet.

Word recognition as a lifting problem

Given a **word recognition problem** of the form

$$\begin{tikzcd} q_0 \arrow[d, dashed, no head] & q_f \arrow[d, dashed, no head] \\ * \arrow[r, "w"] & * \end{tikzcd}$$

a **run of the automaton** is a map $\alpha : q_0 \to q_f$ in the category $\mathbb{Q}$

$$\begin{tikzcd} q_0 \arrow[r, "\alpha"] \arrow[d, dashed, no head] & q_f \arrow[d, dashed, no head] \\ * \arrow[r, "w"] & * \end{tikzcd}$$

whose image $p(\alpha)$ is equal to the word $w : * \to *$ in the category $\mathbb{B}_\Sigma$.

Two key properties of NDFAs

Consider a functor $p : \mathbb{D} \to \mathbb{T}$ of categories.

▷ The functor p is **finitary** if the fibers $p^{-1}(A)$ and $p^{-1}(w)$ are finite for every object A and every arrow w in the category $\mathbb{T}$.

▷ The functor p is **ULF** (unique lifting of factorization, Lawvere & Meni) if

for any arrow α of $\mathbb{D}$, if $p(\alpha) = uv$ for some pairs of arrows u and v of $\mathbb{T}$, there exists a unique pair of arrows β and γ in $\mathbb{D}$ such that

$$\alpha = \beta\gamma, \; p(\beta) = u, \; p(\gamma) = v.$$

Basic observation.

A functor $p : \mathbb{Q} \to \mathbb{B}_{\Sigma}$ is the underlying bare automaton of a NDFA with alphabet Σ iff it is both finitary and ULF.

Definition

A NDFA over a category is a tuple

$$M \quad = \quad (\mathbb{C}, \mathbb{Q}, p : \mathbb{Q} \to \mathbb{C}, q_0, q_f)$$

consisting of

- ▷ a category $\mathbb{C}$ whose maps are called **arrows**
- ▷ a category $\mathbb{Q}$ whose maps are called **runs**
- ▷ a finitary ULF functor $p : \mathbb{Q} \to \mathbb{C}$ transporting runs into arrows,
- ▷ a pair q_0, q_f of objects of $\mathbb{Q}$.

Definition

The **regular language of arrows** L_M recognized by the automaton

$$M \quad = \quad (\mathbb{C}, \mathbb{Q}, p : \mathbb{Q} \to \mathbb{C}, q_0, q_f)$$

is the subset of arrows in the category $\mathbb{C}$

$$\begin{tikzcd}
q_0 \arrow[d, dash, dashed] & q_f \arrow[d, dash, dashed] \\
A \arrow[r, "w"] & B
\end{tikzcd}$$

which can be lifted along p to a run $\alpha : q_0 \to q_f$ in the category $\mathbb{Q}$.

Word recognition as a lifting problem

In other words, an arrow of the category $\mathbb{C}$

$$w \quad : \quad A \longrightarrow B$$

is an element of L_M precisely when the lifting problem

$$\begin{tikzcd}
q_0 \arrow[d, dash, dashed] & & q_f \arrow[d, dash, dashed] \\
A \arrow[rr, "w"] & & B
\end{tikzcd}$$

can be resolved by a run $\alpha : q_0 \to q_f$ of the automaton in $\mathbb{Q}$:

$$\begin{tikzcd}
q_0 \arrow[rr, "\alpha"] \arrow[d, dash, dashed] & & q_f \arrow[d, dash, dashed] \\
A \arrow[rr, "w"] & & B
\end{tikzcd}$$

Context-free grammars

Reminder: A **context-free grammar** is a tuple

$$G \quad = \quad (\Sigma, N, S, P)$$

consisting of

- ▷ a finite set Σ of terminal symbols
- ▷ a finite set N of non-terminal symbols
- ▷ a distinguished element $S \in N$ called the **start symbol**
- ▷ a finite set of **production rules**

$$R \longrightarrow \sigma$$

where $R \in N$ and $\sigma \in (N \cup \Sigma)^*$.

The operad of spliced words

Key observation: any production rule can be factored as

$$R \longrightarrow w_0 \, R_1 \, w_1 \, R_2 \, \ldots \, R_n \, w_n$$

where $w_0, w_1, \ldots, w_n \in \Sigma^*$ and $R_1, \ldots, R_n \in N$.

If we forget the non-terminals, the remaining sequence of words

$$w_0 - w_1 - \ldots - w_n$$

can be seen as an n-ary operation of the **operad of spliced words** $W[\Sigma]$.

The operad of spliced words

Composition in this operad is performed by "splicing into the gaps".

Typically, the binary operation

$$\texttt{super-fragili-cious}$$

may be composed with a word and a unary operation:

$$\texttt{super-fragili-cious} \quad \circ \quad (\texttt{cali}, \texttt{stic-lido})$$

in order to obtain the unary operation

$$\texttt{supercalifragilistic-lidocious}$$

The operad of spliced arrows

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

- its colors are pairs (A, B) of objects of $\mathbb{C}$
- its n-ary operations

  $$(A_1, B_1), \ldots, (A_n, B_n) \longrightarrow (A, B)$$

  are sequences of $n+1$ arrows in $\mathbb{C}$ separated by n gaps

  $$w_0 - w_1 - \ldots - w_n$$

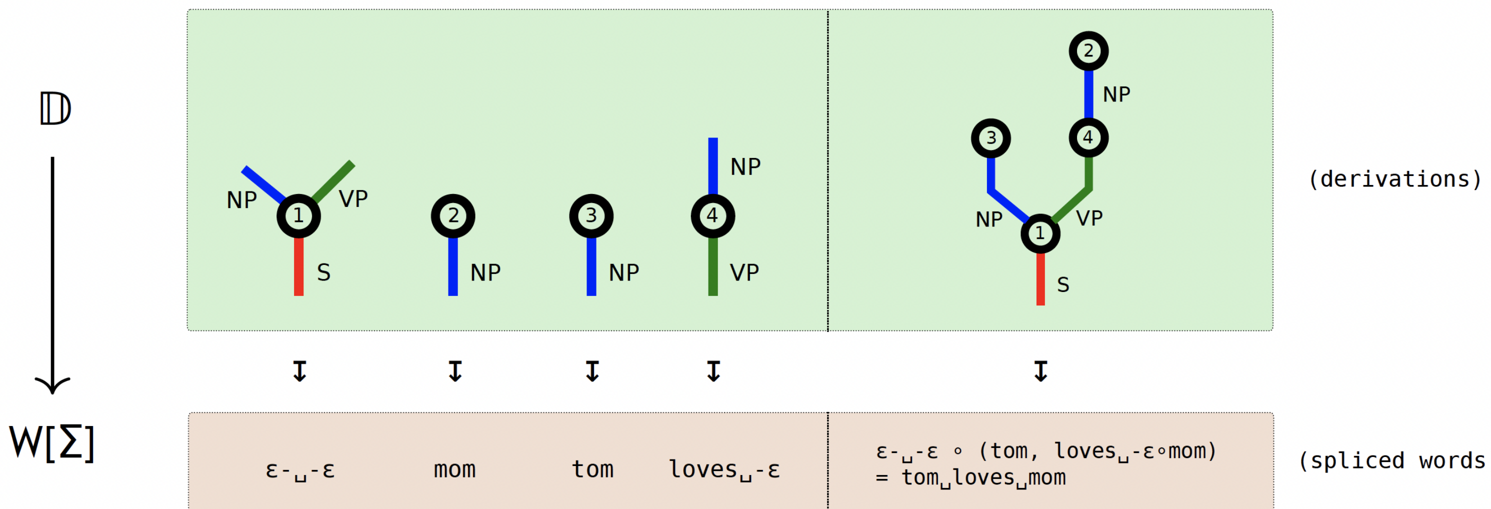
  where each arrow must have type

  $$w_i \quad : \quad B_i \longrightarrow A_{i+1}$$

  under the convention that $B_0 = A$ and that $A_{n+1} = B$.

Context free grammars as functors of operads

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

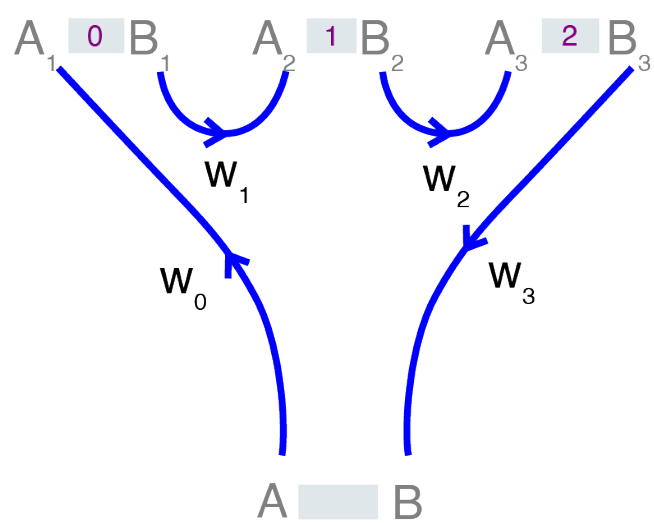


The operad of spliced arrows

The ternary operation

$$w_0 - w_1 - w_2 - w_3 : (A_1, B_1)\,,\,(A_2, B_2)\,,\,(A_3, B_3) \longrightarrow (A, B)$$

is depicted as follows:

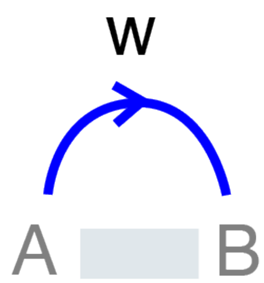


The operad of spliced arrows

The constant operation

$$w : (A, B)$$

is an arrow $w : A \longrightarrow B$ of the category $\mathbb{C}$ depicted as follows:

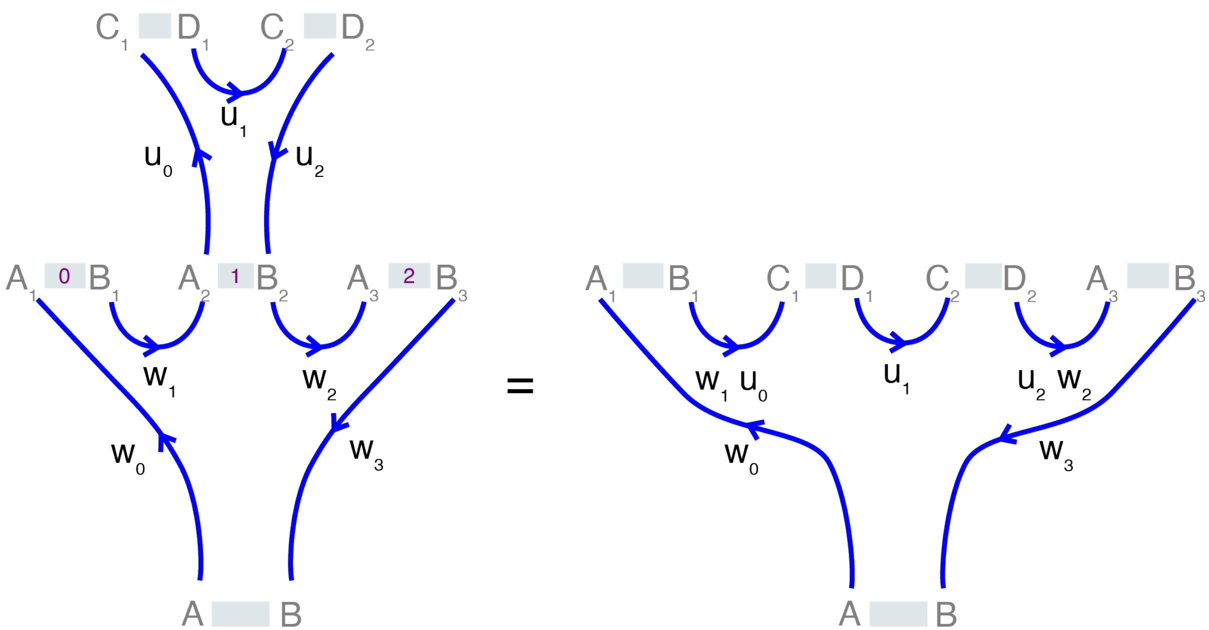


The operad of spliced arrows

The partial composition

$$w_0 - w_1 - w_2 - w_3 \;\circ_1\; u_0 - u_1 - u_2$$

is defined using the composition in the category $\mathbb{C}$, as follows:



Context-free grammar of arrows

Definition.

A **context-free grammar** of arrows is a tuple

$$G = (\mathbb{C}, \mathbb{S}, S, \varphi)$$

consisting of

- ▷ a category $\mathbb{C}$
- ▷ a finite species $\mathbb{S}$ equipped with a distinguished color S
- ▷ a functor of operads

$$p \quad : \quad Free\,\mathbb{S} \longrightarrow \mathbb{W}[\mathbb{C}]$$

Parsing as a lifting problem

The context-free language of arrows

$$L_G$$

generated by the grammar

$$G = (\mathbb{C}, \mathbb{S}, S, \varphi)$$

is the subset of arrows in $\mathbb{C}$ which, seen as constants of $\mathbb{W}[\mathbb{C}]$, are in the image of constants of color S in the operad $Free\,\mathbb{S}$.

Key idea:

The constants of color S in the operad $Free\,\mathbb{S}$ are the **parsing trees** of the context-free grammar G generated by the start symbol S.

The splicing functor

The **operad of spliced arrows construction** defines a functor

$$\mathbb{W}[-] \quad : \quad \mathbf{Cat} \longrightarrow \mathbf{Operad}$$

which happens (not described here) to have a left adjoint

$$\mathbb{C}[-] \quad : \quad \mathbf{Operad} \longrightarrow \mathbf{Cat}$$

which turns an operad into its **contour category**.

Context-free languages ∩ regular languages

Given a functor of operads describing a context-free grammar

$$p \quad : \quad \mathit{Free}\,\mathbb{S} \longrightarrow \mathbb{W}[\mathbb{C}]$$

and a ULF functor of categories

$$F \quad : \quad \mathbb{D} \longrightarrow \mathbb{C}$$

Proposition.

The pullback of p along $\mathbb{W}[F]$ can be computed as a pullback of species

$$\begin{tikzcd}
\mathit{Free}\,\mathbb{S}' \arrow[r, "\mathit{Free}\,\psi"] \arrow[d, "p'"'] \arrow[dr, phantom, "\mathit{pullback}"] & \mathit{Free}\,\mathbb{S} \arrow[d, "p"] \\
\mathbb{W}[\mathbb{D}] \arrow[r, "\mathbb{W}[F]"'] & \mathbb{W}[\mathbb{C}]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{S}' \arrow[r, "\psi"] \arrow[d, "p'"'] \arrow[dr, phantom, "\mathit{pullback}"] & \mathbb{S} \arrow[d, "p"] \\
\mathbb{W}[\mathbb{D}] \arrow[r, "\mathbb{W}[F]"'] & \mathbb{W}[\mathbb{C}]
\end{tikzcd}$$

Context-free languages ∩ regular languages

Given a functor of operads describing a context-free grammar

$$p \quad : \quad Free\,\mathbb{S} \longrightarrow \mathbb{W}[\mathbb{C}]$$

and a NFDA described by a finitary ULF functor

$$F \quad : \quad \mathbb{Q} \longrightarrow \mathbb{C}$$

Fact. The language of arrows in $\mathbb{C}$ of the context-free grammar

$$\begin{tikzcd}
Free\,\mathbb{S}' \arrow[r, "Free\,\psi"] \arrow[d, "p'"'] \arrow[dr, phantom, "pullback"] & Free\,\mathbb{S} \arrow[d, "p"] \\
\mathbb{W}[\mathbb{Q}] \arrow[r, "\mathbb{W}[F]"'] & \mathbb{W}[\mathbb{C}]
\end{tikzcd}$$

is the intersection of the context-free and of the regular language in $\mathbb{C}$.

The representation theorem

This talk only provides a brief 15 minutes introduction to the topic.

If you want to know more, you are welcome to look at our preliminary paper

hal-03702762

You may also listen to **Noam's recent talk** at the Topos Institute!

or take part to the **MFPS 38 conference** which will take place

Monday, Tuesday, Wednesday 11, 12, 13 July

simultaneously in Ithaca and Paris.