

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

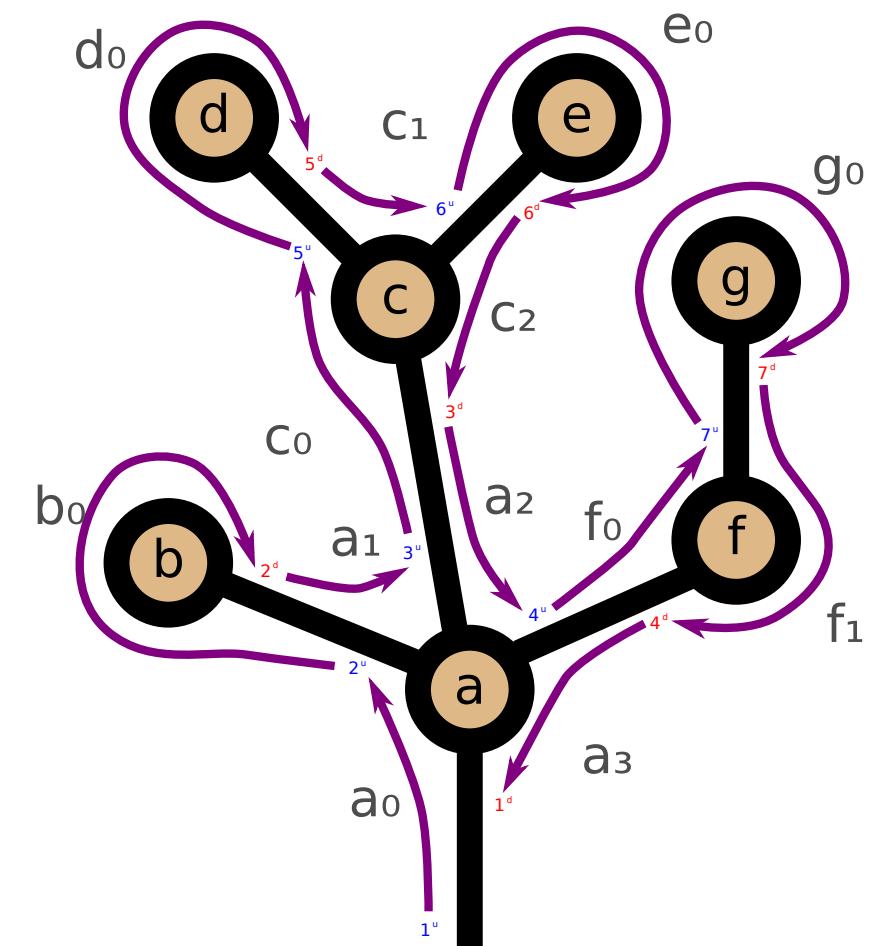
Noam Zeilberger\*

Journée GT DAAL  
EPITA Kremlin-Bicêtre  
21 April 2023

\* Joint work with Paul-André Melliès, in proceedings of MFPS 2022.

<https://doi.org/10.46298/entics.10508>

(long version in preparation)



# A personal timeline, for context

**2013-2016:** we wrote a series of papers on *type systems as functors*. Our original motivation was to better understand not just type systems but other deductive systems as well (e.g., separation logic).

**2017:** we started thinking about context-free grammars and parsing

**2018-2022:** kept thinking...

**24.04.2022:** "how about we try to analyze the C-S rep thm?"

**two weeks later:** wow, that was an interesting theorem!

# The C-S representation theorem

Classical statement: *every CF language is the homomorphic image of the intersection of a Dyck language with a regular language.*

Some reasons why the theorem is interesting:

1. implicitly uses closure of CFLs under intersection with regular langs
2. proof relies on ability to represent derivation trees as words
3. says that Dyck languages are in some sense "universal" CFLs

# A surprising adjunction

We found that at the heart of the C-S representation theorem is an *adjunction* between categories and multicategories:

$$\text{MultiCat} \begin{array}{c} \xrightarrow{C[-]} \\ \perp \\ \xleftarrow{W[-]} \end{array} \text{Cat}$$

This elementary\* adjunction transformed our perspective on CFGs. For reasons to be clear we call it the **contour / splicing** adjunction.

\*but seemingly previously unnoticed!

# (What is a multicategory?)

Like a category, but morphisms can have multiple inputs  $f : A_1, \dots, A_n \rightarrow B$

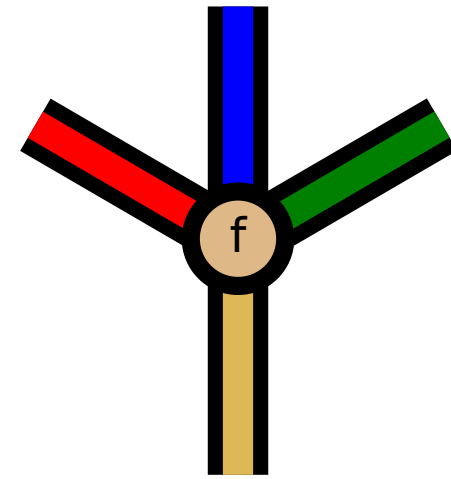
Typical examples:

- Vect, the multicategory of vector spaces and multilinear maps
- a *free multicategory* whose n-ary operations  $f(x_1, \dots, x_n)$  are symbolic expressions in n variables

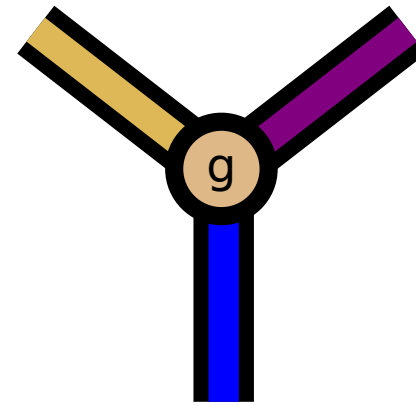
Multicategories are also known as (colored) **operads**, and I will adopt that terminology (which we use in the MFPS paper) from now on...

# What is an operad, a bit more formally

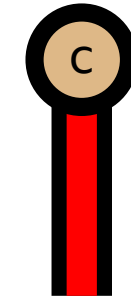
*set of colors*  
+  
*set of operations*



$$f : R, B, G \rightarrow Y$$



$$g : Y, P \rightarrow B$$



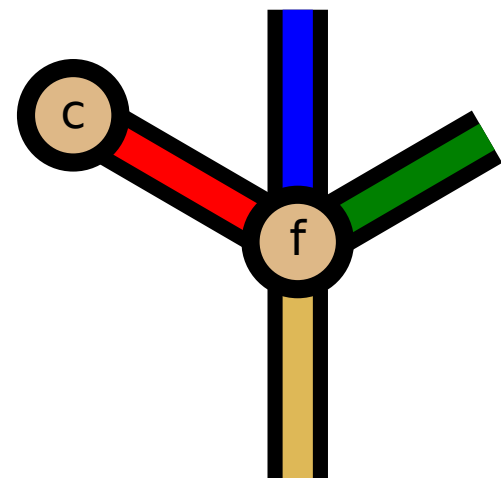
$$c : R$$

+ *identity operation*

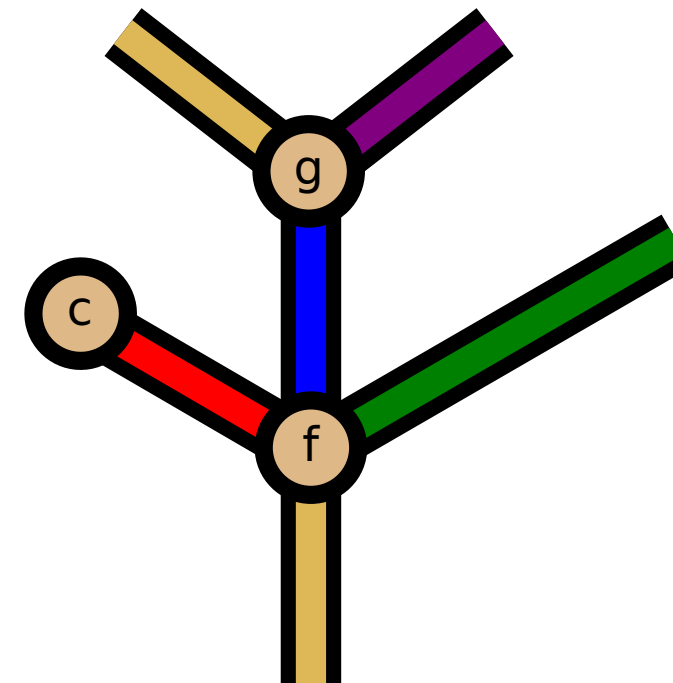


$$\text{id}_G : G \rightarrow G$$

+ *partial / parallel composition*



$$f \circ c : B, G \rightarrow Y$$



$$f \circ (c, g, \text{id}_G) : Y, P, G \rightarrow Y$$

+ *associativity*  
&  
*unitality axioms*

# The operad of spliced words

Consider the operad  $W[\Sigma]$  with a single color, with  $n$ -ary operations given by sequences of  $n+1$  words  $w_0-w_1-\dots-w_n$  over  $\Sigma$ , with composition given by "splicing into the gaps":

$$(ab-ca-ra) \circ (ra,dab) = abracadabra$$

and with identity operation  $\varepsilon-\varepsilon$ .

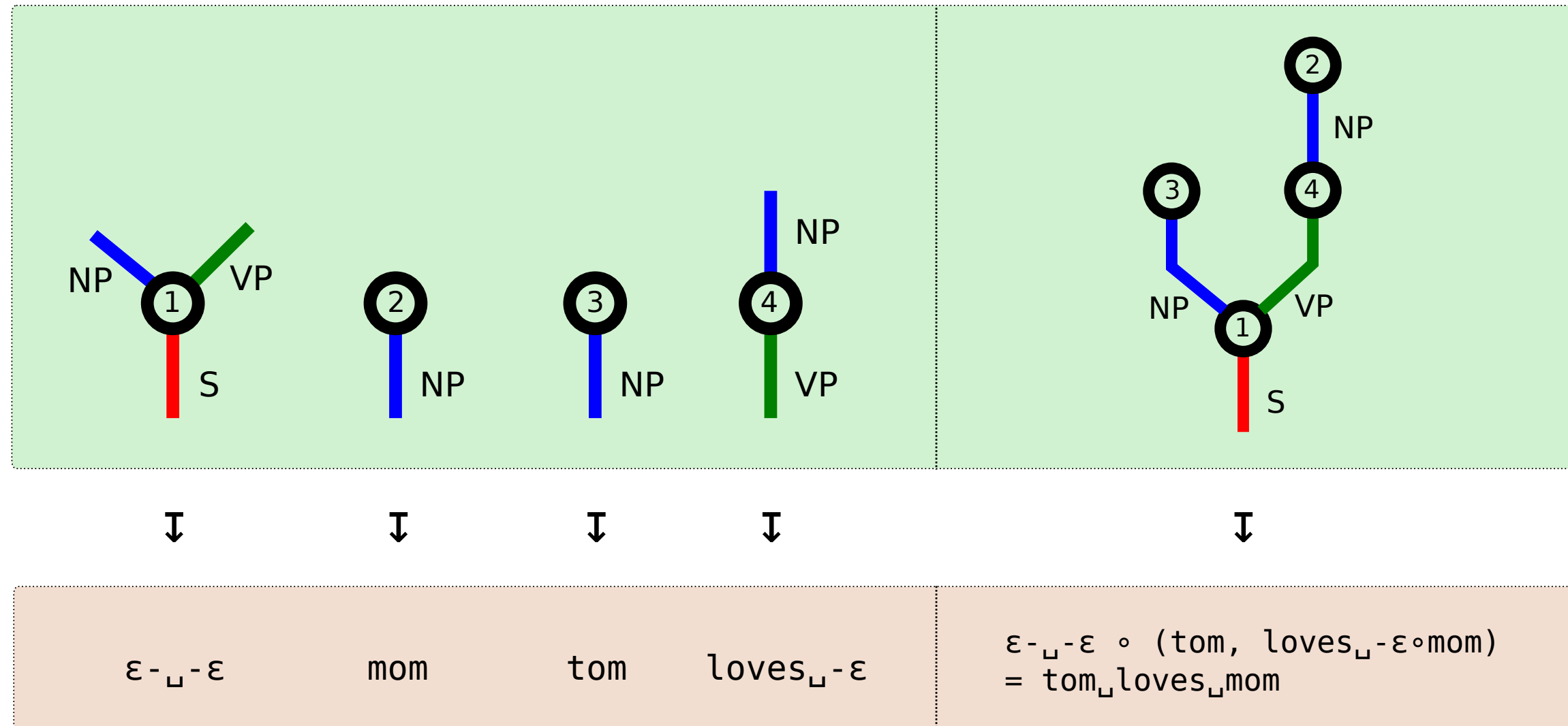
# A functorial view of context-free grammars

Building on the philosophy of type refinement systems, our starting point was the idea that any classical CFG can be represented by a ***functor of operads***  $p : \mathbb{D} \rightarrow W[\Sigma]$  from a freely generated operad  $\mathbb{D}$  into the operad of spliced words  $W[\Sigma]$ ...



# Representing CFGs as functors of operads: example

- 1 :  $S \rightarrow NP \ VP$
- 2 :  $NP \rightarrow mom$
- 3 :  $NP \rightarrow tom$
- 4 :  $VP \rightarrow loves \ NP$



# Plan for the talk

1. show how this functorial viewpoint may be naturally generalized to define CFGs *over any category*.
2. motivate why this viewpoint and this generalization are useful.
3. explain how both word + tree automata may also be viewed functorially, placing them on common ground with CFGs.
4. sketch how to derive (a generalization of) the C-S theorem.

(for details and some discussion of related work, see the MFPS paper)

# ***Context-free languages of arrows***

# The operad of spliced arrows

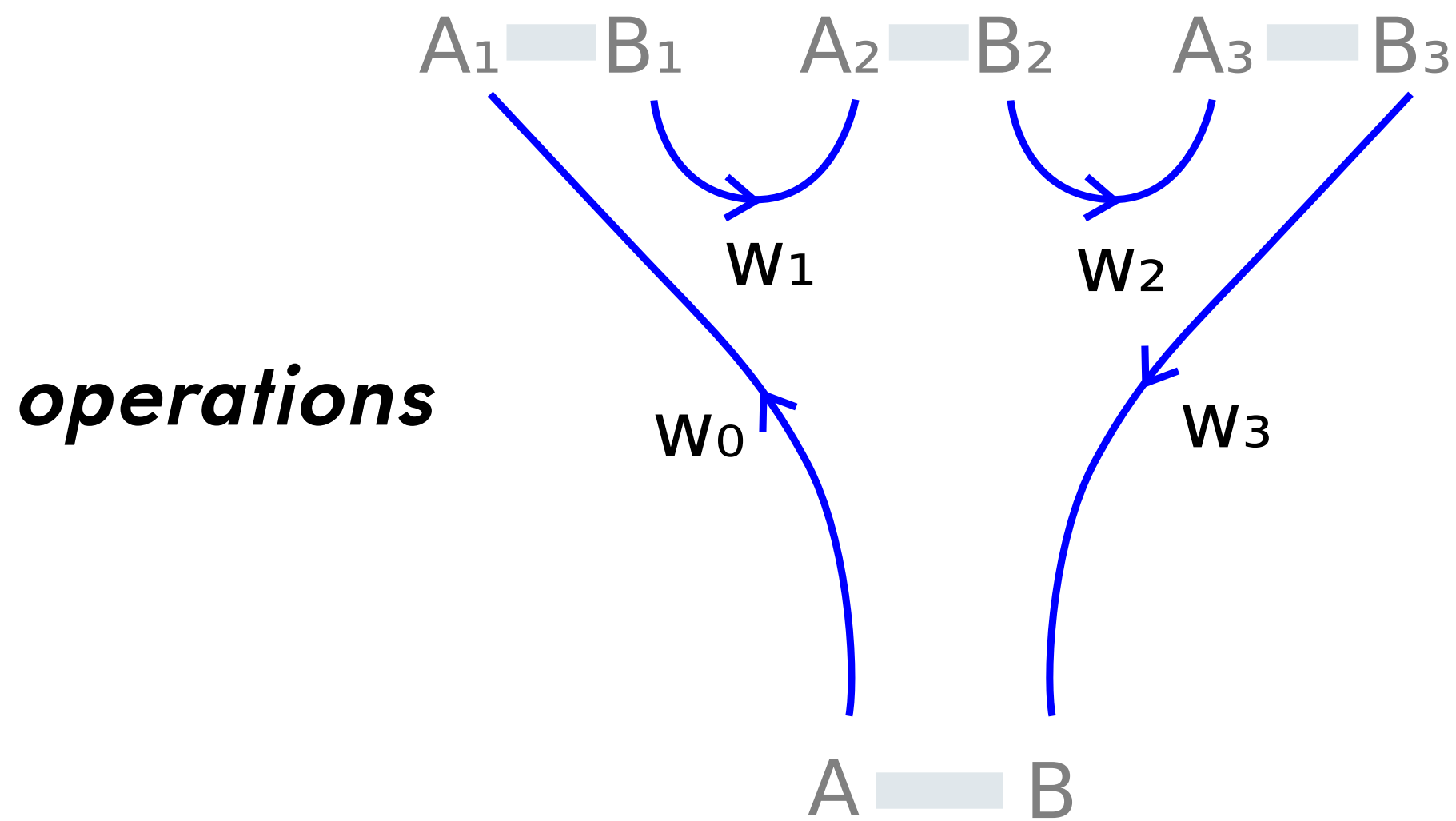
Let  $\mathbb{C}$  be a category. The operad  $W[\mathbb{C}]$  is defined as follows:

- its colors are pairs  $(A, B)$  of objects of  $\mathbb{C}$ ;
- its  $n$ -ary operations  $(A_1, B_1), \dots, (A_n, B_n) \rightarrow (A, B)$  consist of sequences  $w_0 - w_1 - \dots - w_n$  of  $n+1$  arrows in  $\mathbb{C}$  separated by  $n$  gaps notated  $-$ , where  $w_i : B_i \rightarrow A_{i+1}$  for  $0 \leq i \leq n$ , taking  $B_0 = A$  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  $\text{id}_A - \text{id}_B$ .

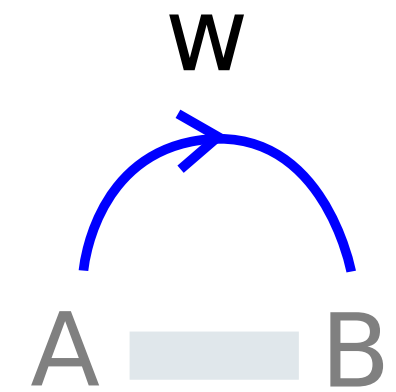
( $W[\mathbb{C}]$  generalizes  $W[\Sigma]$ , taking  $\mathbb{C} = \mathbb{B}_\Sigma$  the free monoid seen as one-object category.)

$$\mathbb{B}_\Sigma = \overset{a}{\curvearrowright} \ast \curvearrowleft \overset{b}{}$$

# The operad of spliced arrows



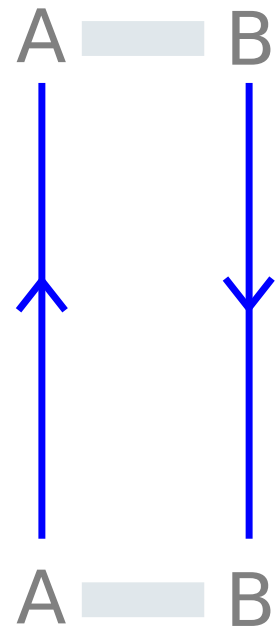
$$W_0 - W_1 - W_2 - W_3 : (A_1, B_1), (A_2, B_2), (A_3, B_3) \rightarrow (A, B)$$



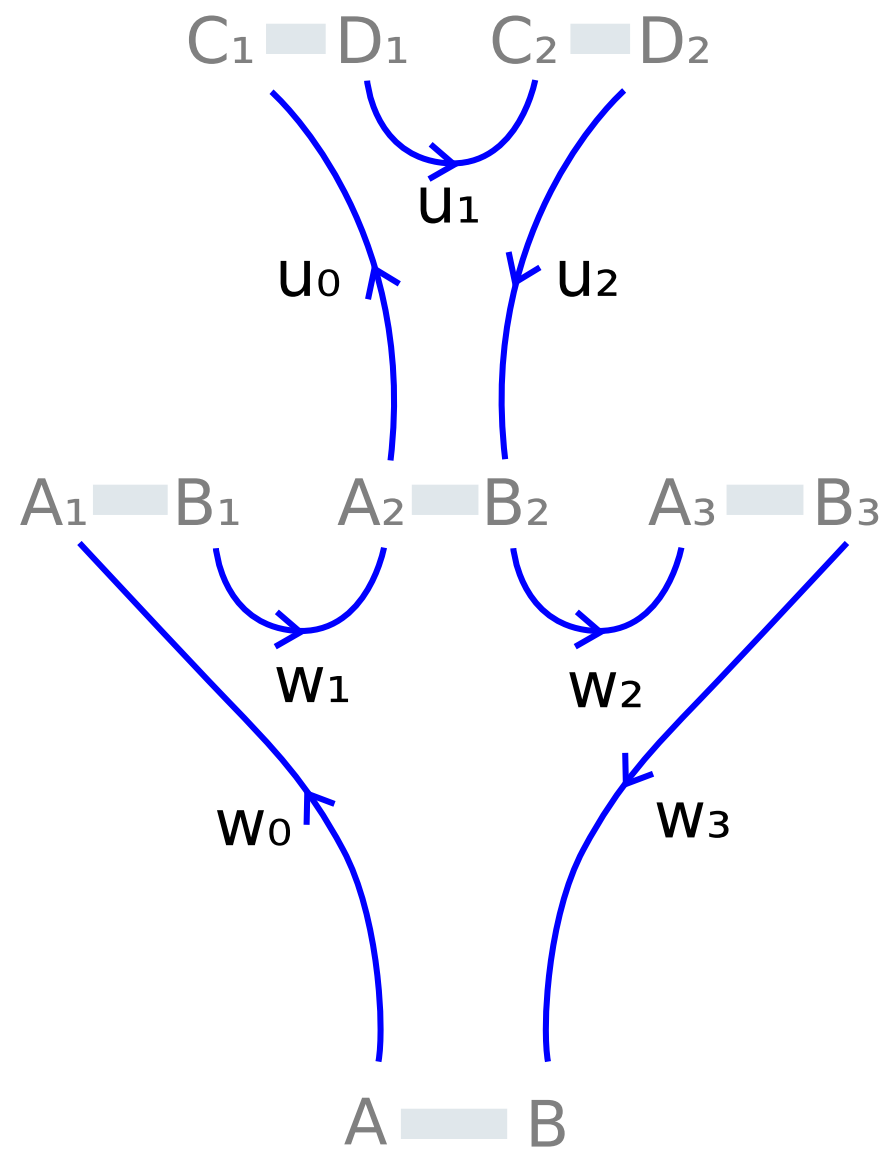
$$w : (A, B)$$

# The operad of spliced arrows

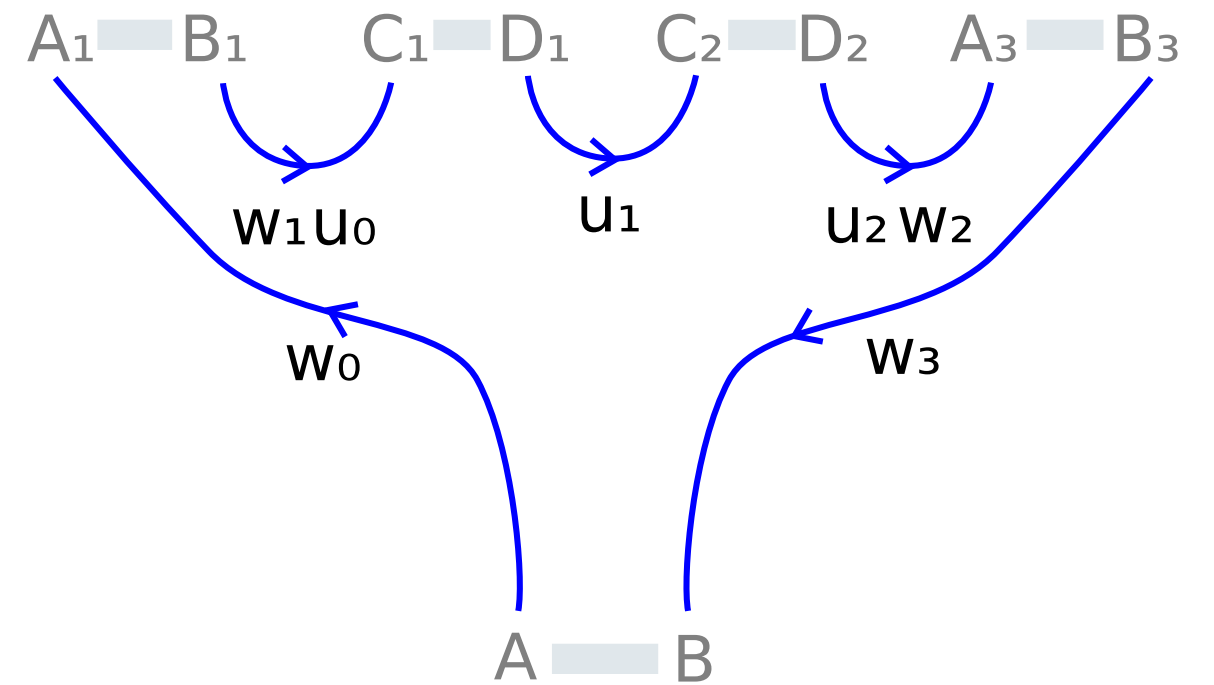
*identity*



*partial composition*



=



# The splicing functor

The operad of spliced arrows construction defines a functor

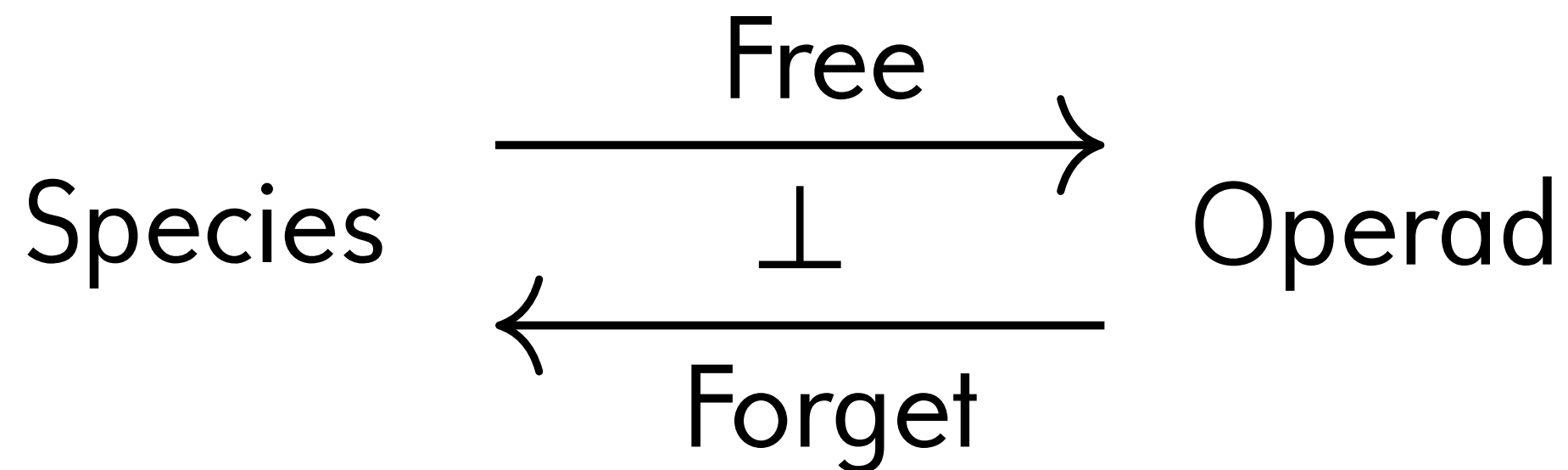
$$\text{Cat} \xrightarrow{W[-]} \text{Operad}$$

since any functor of categories  $F : \mathbb{C} \rightarrow \mathbb{D}$  induces a functor of operads  $W[F] : W[\mathbb{C}] \rightarrow W[\mathbb{D}]$ .

# Free operads

A **species** is a set of "nodes" with colored edges. Any species  $\mathcal{S}$  generates a **free operad** ( $\text{Free } \mathcal{S}$ ) whose operations are trees, with nodes labeled by nodes of  $\mathcal{S}$  while respecting the coloring constraints on edges.

Conversely, any operad  $\mathbb{O}$  has an **underlying species** ( $\text{Forget } \mathbb{O}$ ) with nodes given by operations of  $\mathbb{O}$ , simply forgetting about composition and identity.



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



# Definition

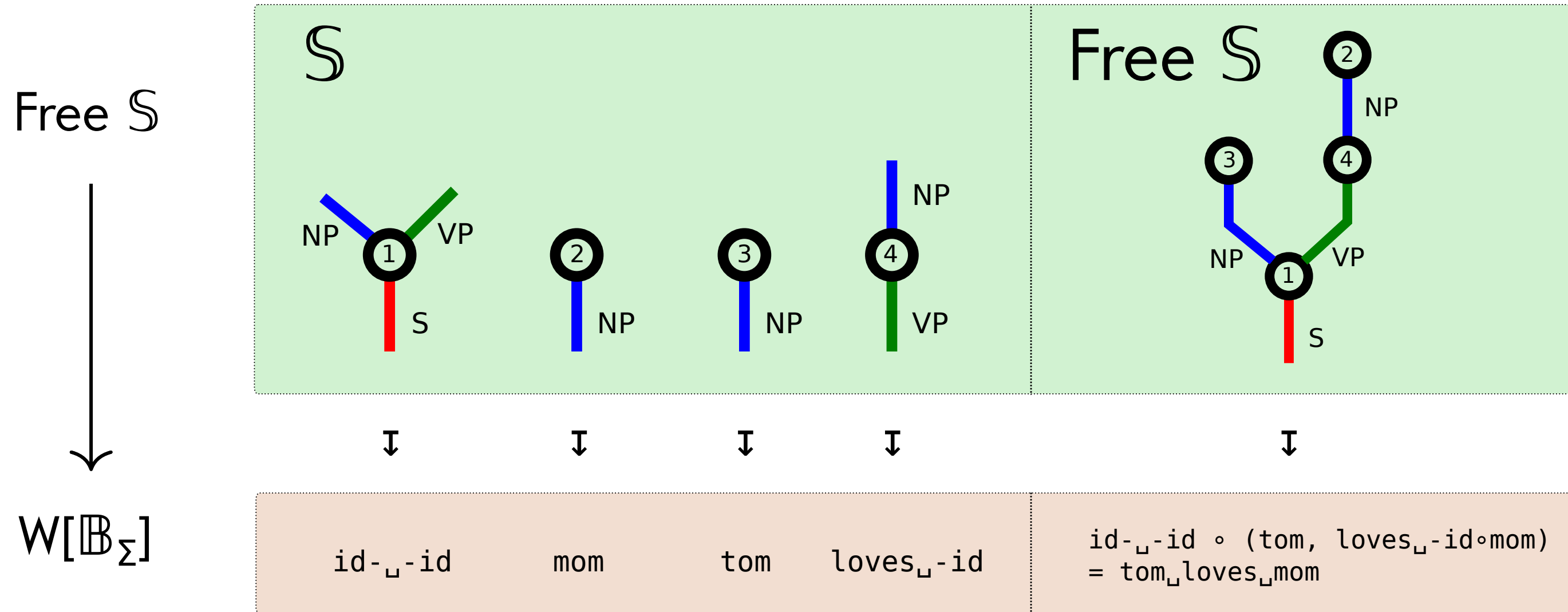
A **context-free grammar of arrows** is a tuple  $G = (\mathbb{C}, \mathbb{S}, S, \varphi)$  consisting of a category  $\mathbb{C}$ , a finite species  $\mathbb{S}$  equipped with a distinguished color  $S \in \mathbb{S}$  and a functor of operads  $p : \text{Free } \mathbb{S} \rightarrow W[\mathbb{C}]$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

The **context-free language of arrows**  $L_G$  generated by the grammar  $G$  is the subset of arrows in  $\mathbb{C}$  which, seen as constants of  $W[\mathbb{C}]$ , are in the image of constants of color  $S$  in  $\text{Free } \mathbb{S}$ , 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 :  $S \rightarrow NP VP$
- 2 :  $NP \rightarrow mom$
- 3 :  $NP \rightarrow tom$
- 4 :  $VP \rightarrow loves NP$



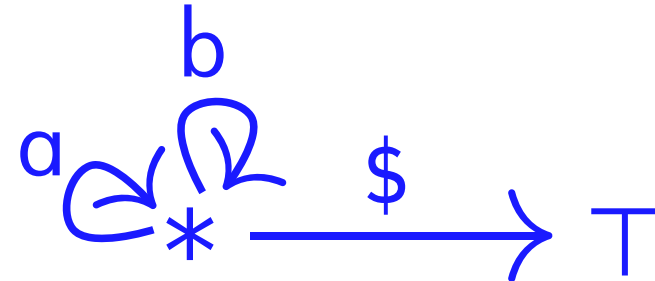
# Refining classical CFGs with "gap types"

A feature of the general definition is that non-terminals are sorted.

We write  $R \sqsubset (A,B)$  to mean  $p(R) = (A,B)$  and say  $R$  *refines* the **gap type**  $(A,B)$ .

If  $G$  has start symbol  $S \sqsubset (A,B)$  then  $L_G \subseteq \mathbb{C}(A,B)$ .

# Refining classical CFGs with "gap types"

For example, consider the category  $\mathbb{B}_{\Sigma}^{\top} =$   constructed from  $\mathbb{B}_{\Sigma}$  by freely adjoining an object and an arrow.

A CFG over  $\mathbb{B}_{\Sigma}^{\top}$  may include production rules that can only be applied upon ***end of input***, like Knuth's "0th production" rule  $S' \rightarrow S\$$  from the LR parsing paper. (Here  $S \sqsubset (*, *)$  is "classical" while  $S' \sqsubset (*, \top)$  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  $\text{Free } \mathbb{S}$ , if  $p(\alpha) = p(\beta)$  then  $\alpha = \beta$ .
- A non-terminal  $R$  is **nullable** if there exists a constant  $\alpha : R$  of  $\text{Free } \mathbb{S}$  s.t.  $p(\alpha) = \text{id}$ .
- A non-terminal  $R$  is **useful** if there exists a constant  $\alpha : R$  and a unary op  $\beta : R \rightarrow S$ . 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  $L_1 \subseteq \mathbb{C}(A_1, B_1), \dots, L_n \subseteq \mathbb{C}(A_n, B_n)$  are CF, and  $w_0 w_1 \cdots w_n : (A_1, B_1), \dots, (A_n, B_n) \rightarrow (A, B)$  is an operation of  $W[\mathbb{C}]$ , then  $w_0 L_1 w_1 \cdots L_n w_n \subseteq \mathbb{C}(A, B)$  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  $G_1 = (\mathbb{C}, \mathcal{S}_1, S_1, p_1)$  and  $G_2 = (\mathbb{C}, \mathcal{S}_2, S_2, p_2)$  be two CFGs over the same category  $\mathbb{C}$ .

If there is a fully faithful functor of operads  $T : \text{Free } \mathcal{S}_1 \rightarrow \text{Free } \mathcal{S}_2$  such that  $p_1 = T p_2$  and  $T(S_1) = S_2$ , then  $L_{G_1} = L_{G_2}$ , moreover with the grammars generating isomorphic sets of parse trees.

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

(cf. Leermakers 1989)

***A quick digression on  
generalized CFGs and gCFLs***

(from the long version of the paper, not yet online)



# Definition

A **generalized CFG (over an operad)** is a tuple  $G = (\mathbb{O}, \mathbb{S}, S, \varphi)$  consisting of an operad  $\mathbb{O}$ , a finite species  $\mathbb{S}$  equipped with a distinguished color  $S \in \mathbb{S}$  and a functor of operads  $p : \text{Free } \mathbb{S} \rightarrow \mathbb{O}$ . The **language of constants**  $L_G$  generated by the grammar  $G$  is given by the subset of constants  $L_G = \{ p(\alpha) \mid \alpha : S \}$ .

CFGs of arrows are the special case where  $\mathbb{O} = W[\mathbb{C}]$  is a spliced arrow operad.

# A few examples

**Multiple CFGs** (Seki et al.) obtained taking  $\mathbb{O} = L_{\text{aff}} W[\mathbb{C}]$ , where  $L_{\text{aff}} \mathbb{P}$  is the free semi-cartesian (= "affine") strict monoidal operad over  $\mathbb{P}$ . Note the colors of  $L_{\text{aff}} \mathbb{P}$  are given by lists  $[A_1, \dots, A_k]$  of colors in  $\mathbb{P}$ , and operations  $[\Gamma_1], \dots, [\Gamma_n] \rightarrow [A_1, \dots, A_k]$  by pairs  $([f_1, \dots, f_k], \sigma)$  of a list of operations  $f_1 : \Omega_1 \rightarrow A_1, \dots, f_k : \Omega_k \rightarrow A_k$  in  $\mathbb{P}$  and an injection  $\sigma : \Omega_1, \dots, \Omega_k \hookrightarrow \Gamma_1, \dots, \Gamma_n$ .  
(e.g., there is a 3-mCFG generating the language  $a^n \# b^n \# c^n$ )

**Parallel multiple CFGs** (Seki et al.) obtained taking  $\mathbb{O} = L_{\text{cart}} W[\mathbb{C}]$ .

Can also recover more "semantic" examples, e.g., **series-parallel graphs** are generated by a gCFG over  $\mathbb{O} = \text{Set}$  (cf. Courcelle & Engelfriet 2012, §1.1.3).

# Abstracting the notion of language

There is an old idea, that a context-free language may be considered as a minimal solution to a system of (polynomial) equations.

Ginsburg & Rice 1962, Mezei & Wright 1967, Conway 1971

We categorify this idea by first introducing a notion of **model** of a functor  $p : \text{Free } \mathcal{S} \rightarrow \mathbb{O}$  in an arbitrary target functor  $q : \mathbb{E} \rightarrow \mathbb{B}$ , given by a square

$$\begin{array}{ccc} \text{Free } \mathcal{S} & \xrightarrow{[-]'} & \mathbb{E} \\ p \downarrow & & \downarrow q \\ \mathbb{O} & \xrightarrow{[-]} & \mathbb{B} \end{array}$$

satisfying an extra condition ("cones in  $\mathcal{S}$  sent to  $q$ -minimal cones in  $\mathbb{E}$ ").

# Abstracting the notion of language

We can then define the **q-language** generated by a gCFG  $(\mathbb{Q}, \mathbb{S}, S, p)$  as the interpretation  $\llbracket S \rrbracket' \sqsubseteq^q \llbracket A \rrbracket$  of its start symbol  $S \sqsubseteq^p A$  for some *initial model*  $(\llbracket - \rrbracket', \llbracket - \rrbracket) : p \rightarrow q$ , when such a model exists and is hence unique up to canonical iso.

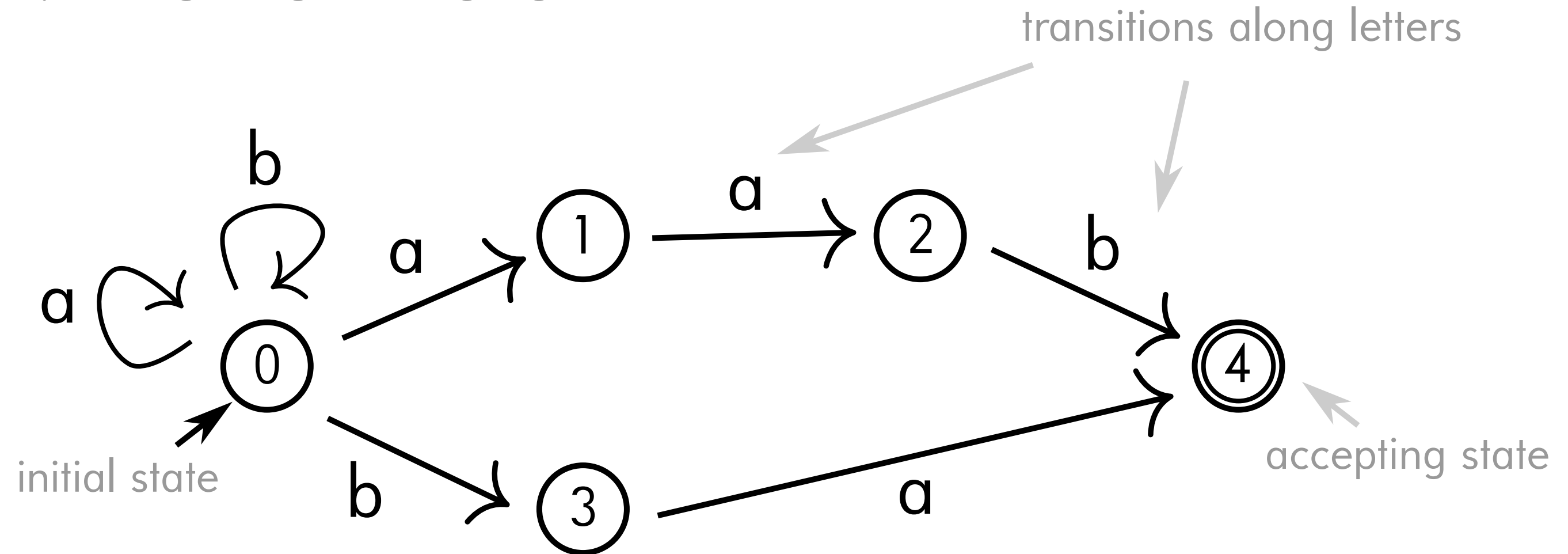
The traditional language is recovered as the q-language for  $q = \text{sub} : \text{Subset} \rightarrow \text{Set}$ . But every gCFG also has an initial model in  $\text{tgt} : \text{Set}^{\rightarrow} \rightarrow \text{Set}$ , which we can see as a ***proof-relevant language*** encoding not just a subset of words but also their parses.

q-languages satisfy good closure properties if  $q$  has sufficient structure.

***Finite-state automata  
over categories and operads***

# Reminder on finite state automata

An **N DFA**: [recognizing the language  $(a+b)^*(abb+ba)$ ]

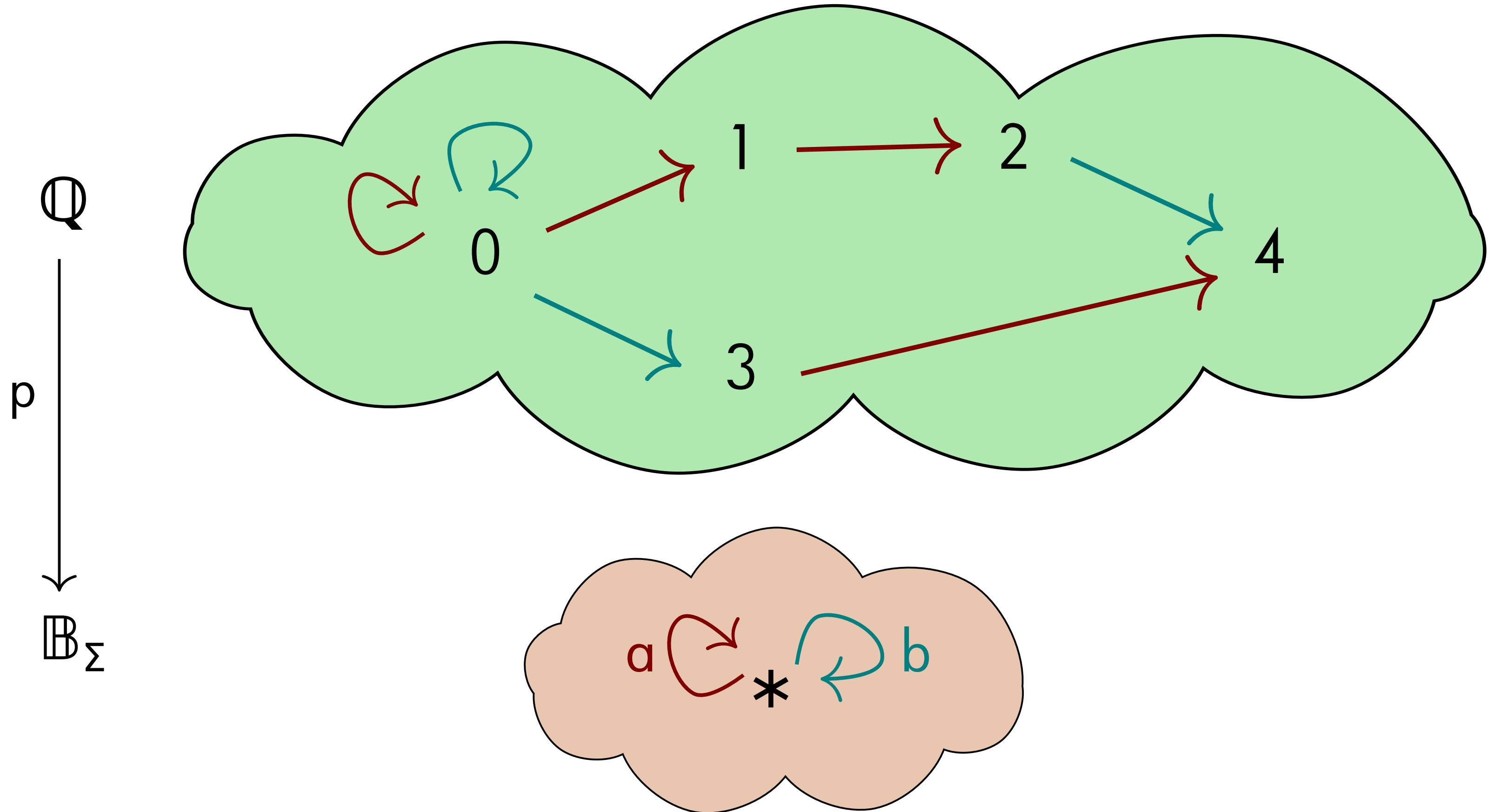


**alphabet**  $\Sigma = \{a,b\}$

**state set**  $Q = \{0,1,2,3,4\}$

*(no  $\epsilon$ -transitions)*

# Representing automata as functors



# Two key properties of NDFAs

Let  $p : \mathbb{D} \rightarrow \mathbb{T}$  be a functor of categories.

$p$  is **finitary** if  $p^{-1}(A)$  and  $p^{-1}(w)$  are finite for every object  $A$  and arrow  $w$  of  $\mathbb{T}$ .

$p$  is **ULF** if for any arrow  $\alpha$  of  $\mathbb{D}$ , if  $p(\alpha) = uv$  for some arrows  $u$  and  $v$  of  $\mathbb{T}$ , there exist unique arrows  $\beta$  and  $\gamma$  of  $\mathbb{D}$  such that  $\alpha = \beta\gamma$  and  $p(\beta) = u$  and  $p(\gamma) = v$ .

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

(Note: ULF = "unique lifting of factorizations" is a generalization of the property of being a discrete (op)fibration. A finitary discrete opfibration  $p : \mathbb{Q} \rightarrow \mathbb{B}_\Sigma$  corresponds to the underlying bare automaton of a *deterministic* finite automaton.)



# Definition

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 definitions of finitary / ULF extend smoothly to functors of operads.

We define an **NDFA over an operad** as a tuple  $M = (\mathbb{O}, \mathbb{Q}, p : \mathbb{Q} \rightarrow \mathbb{O}, q)$  where  $p : \mathbb{Q} \rightarrow \mathbb{O}$  is a finitary ULF functor of operads, and  $q$  a color of  $\mathbb{Q}$ .

When  $\mathbb{O} = \text{Free } \Sigma$  is the free operad over a ranked alphabet  $\Sigma$ , this recovers standard ND **finite state tree automata**. But the above notion is more general!

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

*$\therefore$  any NDFA over a category induces an NDFA over its spliced arrow operad, by the mapping  $(p : \mathbb{Q} \rightarrow \mathbb{C}, q_0, q_f) \mapsto (W[p] : W[\mathbb{Q}] \rightarrow W[\mathbb{C}], (q_0, q_f))$*

# ***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 CFLs 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.

# Pulling back a CFG along a NDFA

General properties of ULF functors imply that the pullback on the left (in Species) is mapped to a pullback on the right (in Operad):

$$\begin{array}{ccc}
 S' & \xrightarrow{\psi} & S \\
 \varphi' \downarrow & \text{pullback} & \downarrow \varphi_G \\
 W[\mathbb{Q}] & \xrightarrow{W[p_M]} & W[\mathbb{C}]
 \end{array}
 \quad \Bigg| \quad
 \begin{array}{ccc}
 \text{Free } S' & \xrightarrow{\text{Free } \psi} & \text{Free } S \\
 p' \downarrow & \text{pullback} & \downarrow p_G \\
 W[\mathbb{Q}] & \xrightarrow{W[p_M]} & W[\mathbb{C}]
 \end{array}$$

This allows us to define the pullback of a CFG  $G$  along a NDFA  $M$  by  $G' = (\mathbb{Q}, S', (S, (q_0, q_f)), p')$ . Note  $G'$  generates a **language of runs** of  $M$ !

Taking the image of  $G'$  along  $p_M$  yields a grammar generating  $L_G \cap L_M$ .

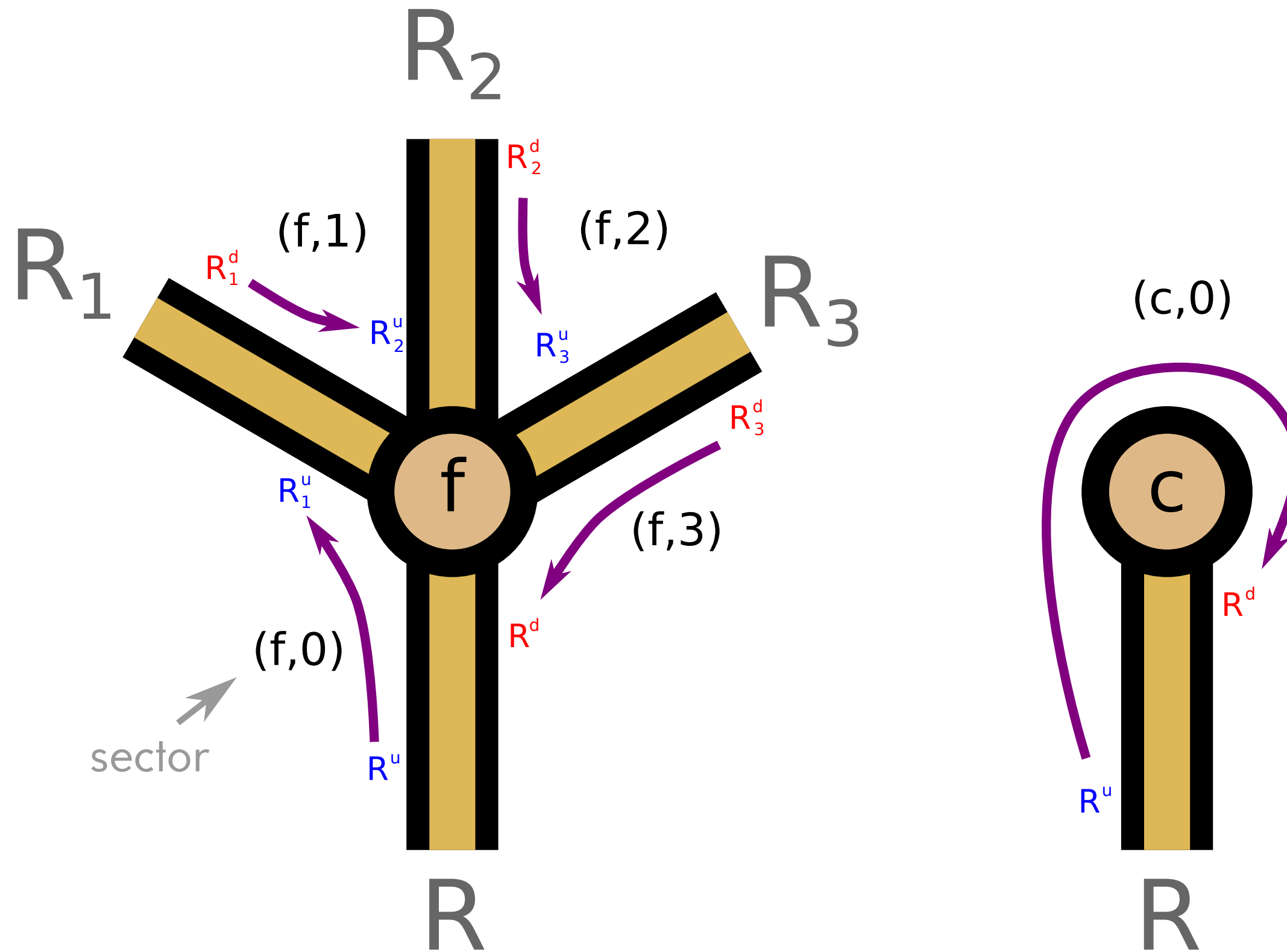
# The contour category of an operad

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

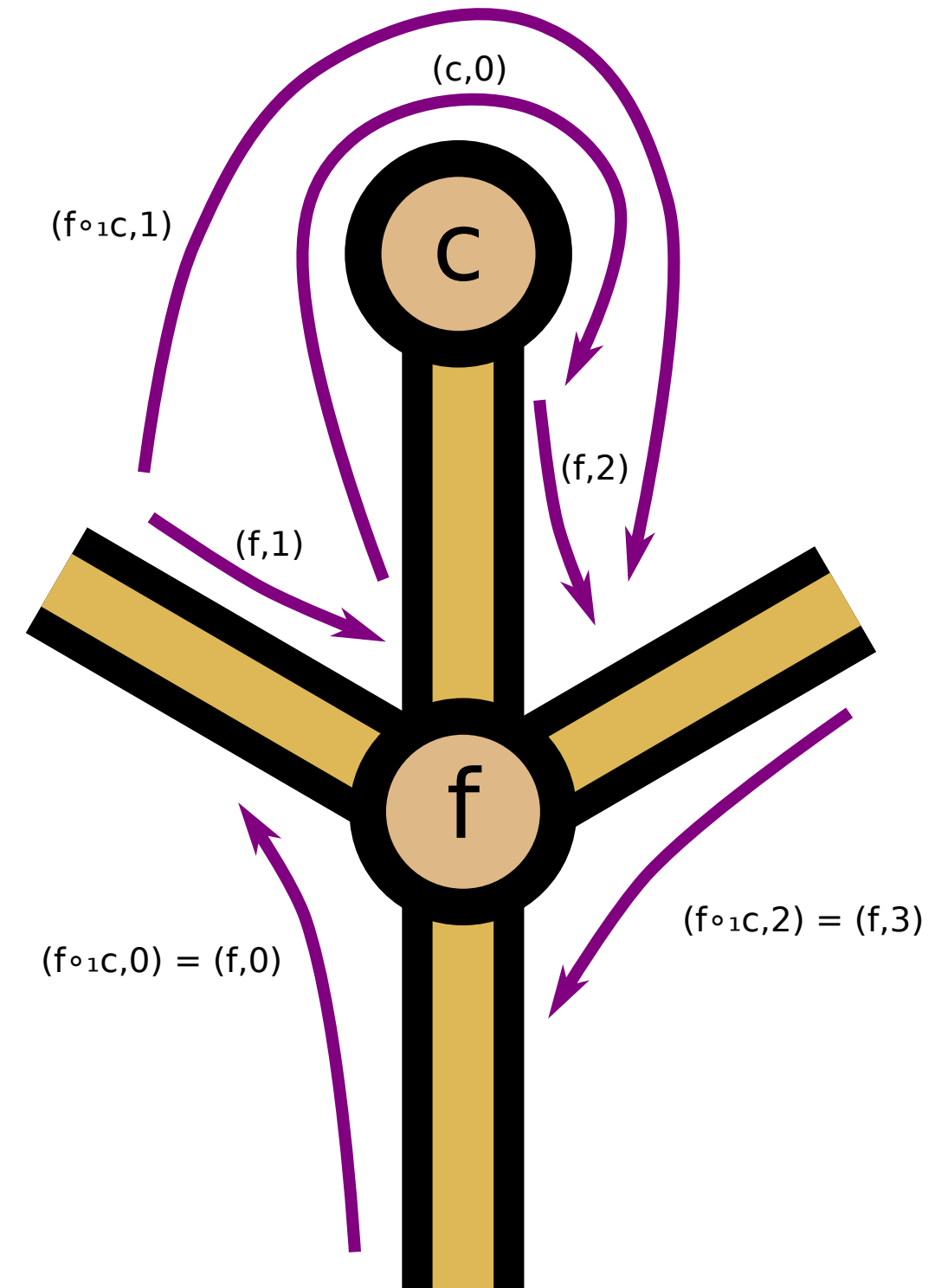
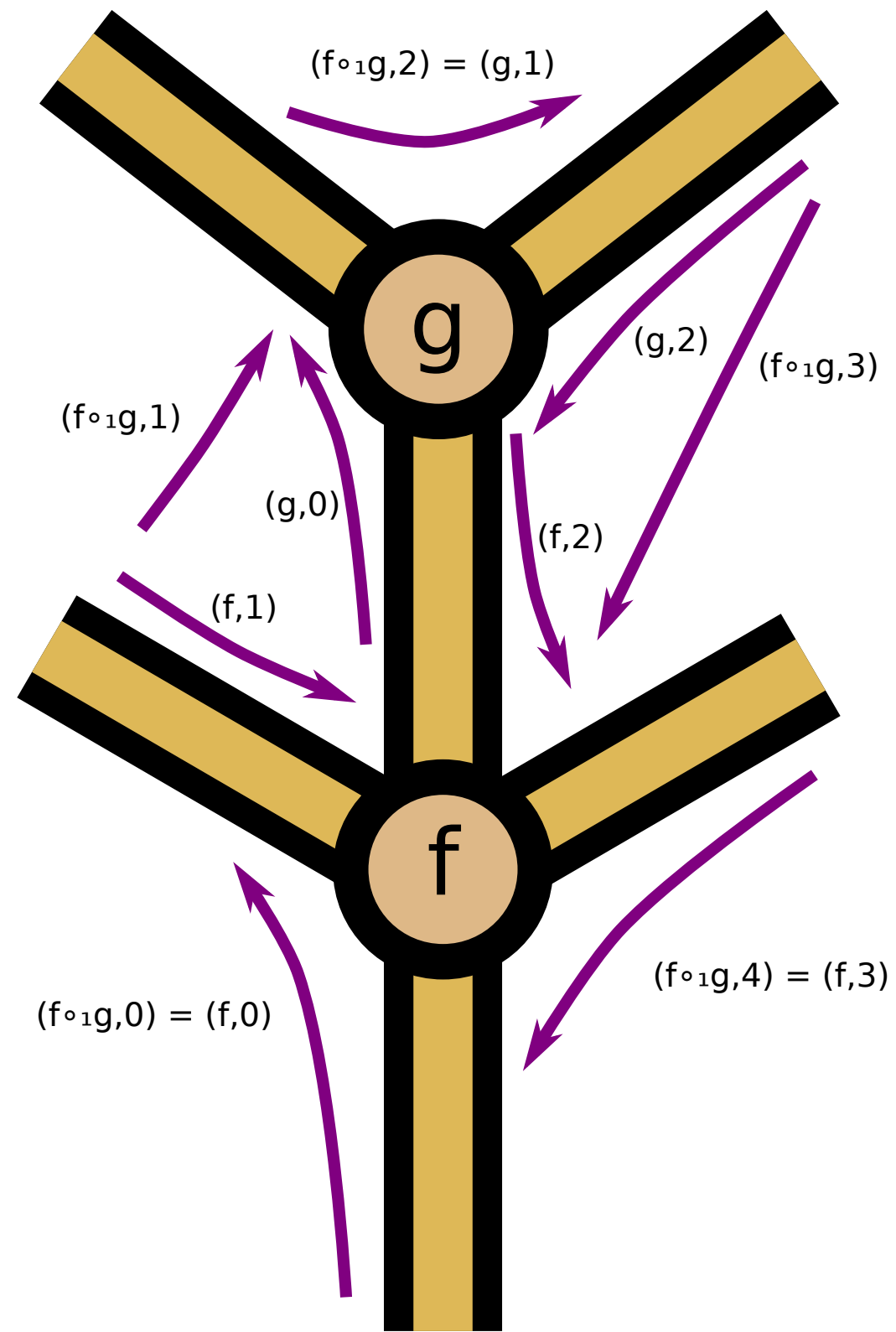
- objects given by *oriented colors*  $R^\varepsilon$  consisting of a color  $R$  of  $\mathbb{O}$  and an orientation  $\varepsilon \in \{u,d\}$  ("up" or "down");
- arrows generated by pairs  $(f,i)$  of an operation  $f : R_1, \dots, R_n \rightarrow R$  of  $\mathbb{O}$  and an index  $0 \leq i \leq n$ , defining an arrow  $R_i^d \rightarrow R_{i+1}^u$  where  $R_0^d = R^u$  and  $R_{n+1}^u = R^d$ ;

subject to some reasonable equations with a geometric interpretation...

# The contour category of an operad



# The contour category of an operad





# The contour / splicing adjunction

This construction provides a left adjoint to the splicing construction:

$$\text{Operad} \begin{array}{c} \xrightarrow{C[-]} \\ \perp \\ \xleftarrow{W[-]} \end{array} \text{Cat}$$

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

The unit and counit have nice descriptions:

$$\eta : \mathbb{O} \rightarrow W[C[\mathbb{O}]]$$

$$R \mapsto (R^u, R^d)$$

$$f \mapsto (f, 0) \cdots (f, n)$$

$$\varepsilon : C[W[\mathbb{C}]] \rightarrow \mathbb{C}$$

$$(A, B)^u \mapsto A \quad (A, B)^d \mapsto B$$

$$(w_0 \cdots w_n, i) \mapsto w_i$$

# 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

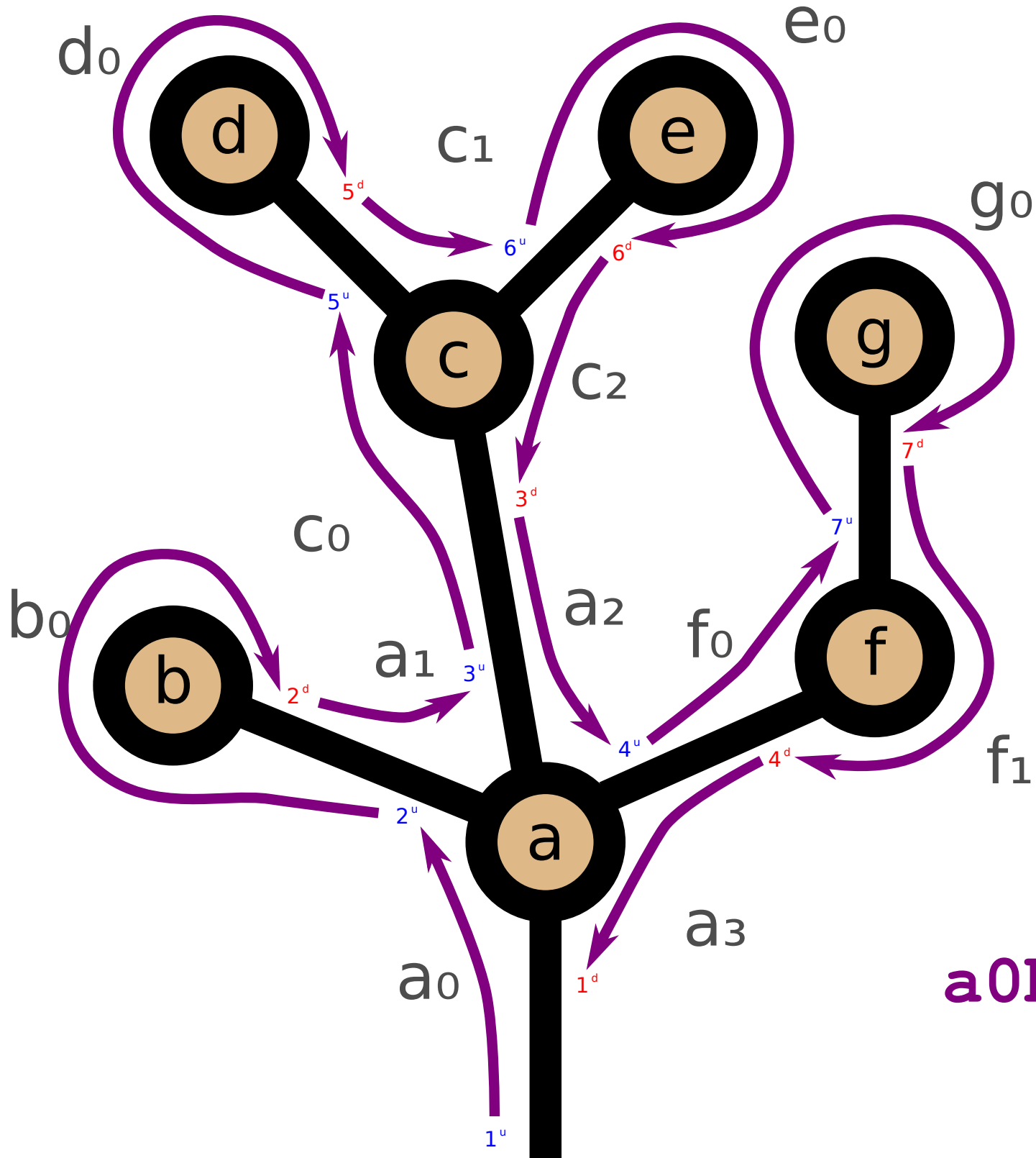
$$\text{Free } \mathcal{S} \xrightarrow{\eta_{\mathcal{S}}} W[\mathbb{C}[\text{Free } \mathcal{S}]] \xrightarrow{W[q]} W[\mathbb{C}]$$

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

The CFG  $\text{Univ}_{\mathcal{S}, \mathcal{S}} = (\mathbb{C}[\text{Free } \mathcal{S}], \mathcal{S}, \mathcal{S}, \eta_{\mathcal{S}})$  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 $\mathcal{S}$



$\mathcal{S}$		
a	: 2, 3, 4	$\rightarrow 1$
b	: 2	
c	: 5, 6	$\rightarrow 3$
d	: 5	
e	: 6	
f	: 7	$\rightarrow 4$
g	: 7	

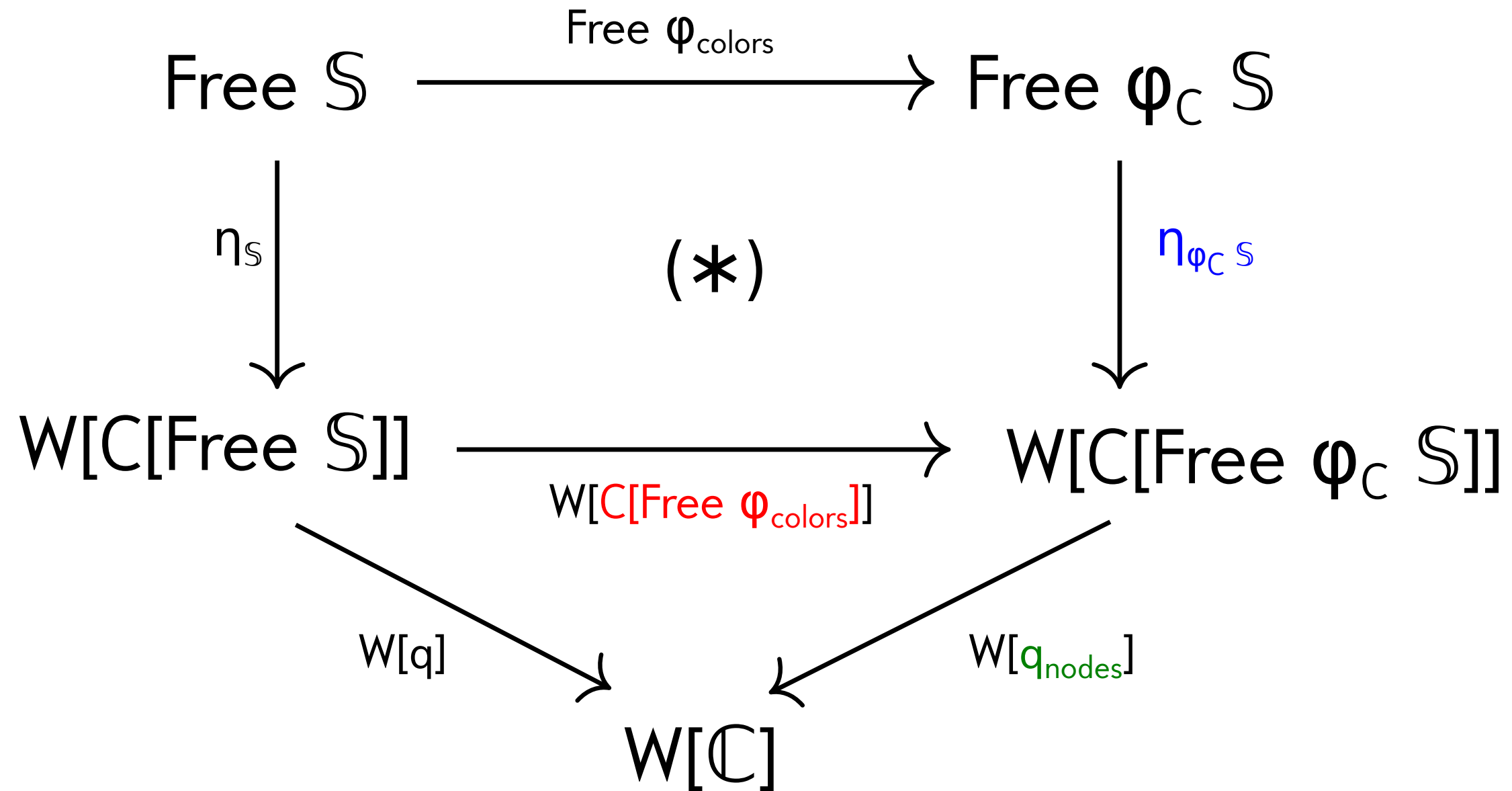
a0b0a1c0d0c1e0c2a2f0g0f1a3 : 1<sup>u</sup>  $\rightarrow$  1<sup>d</sup>

# 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.

# The proof in a diagram

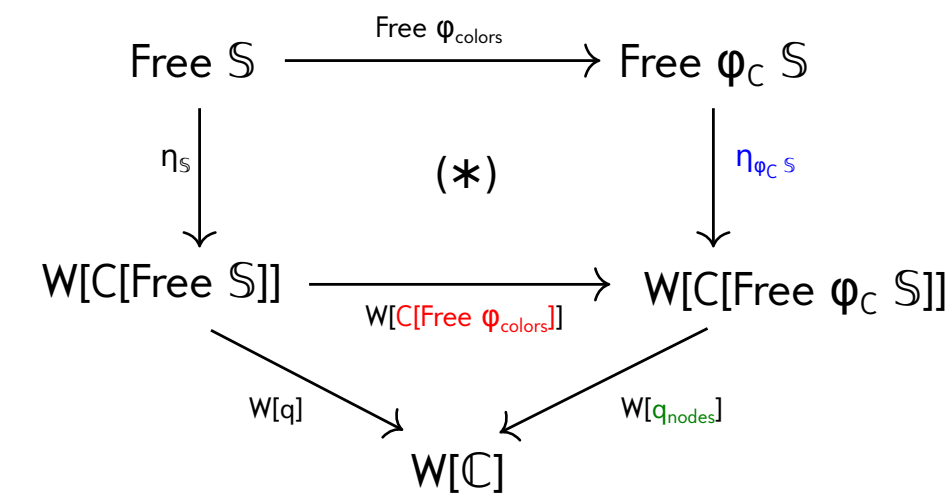
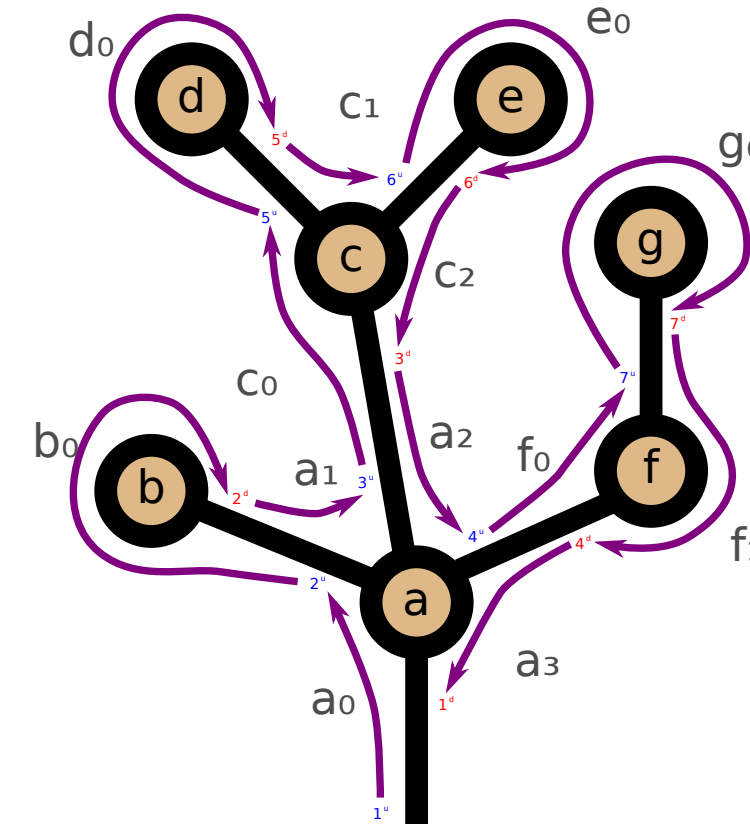
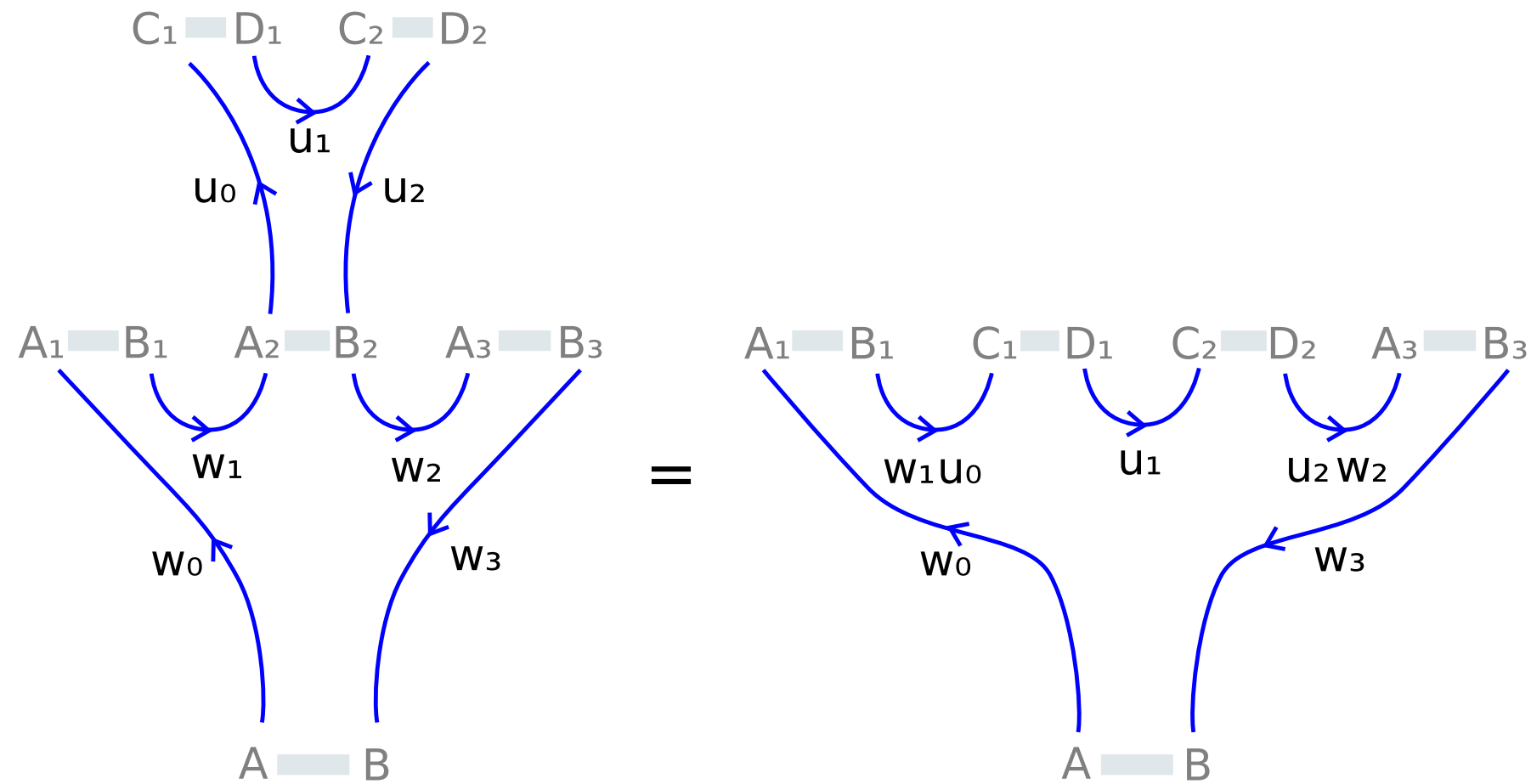
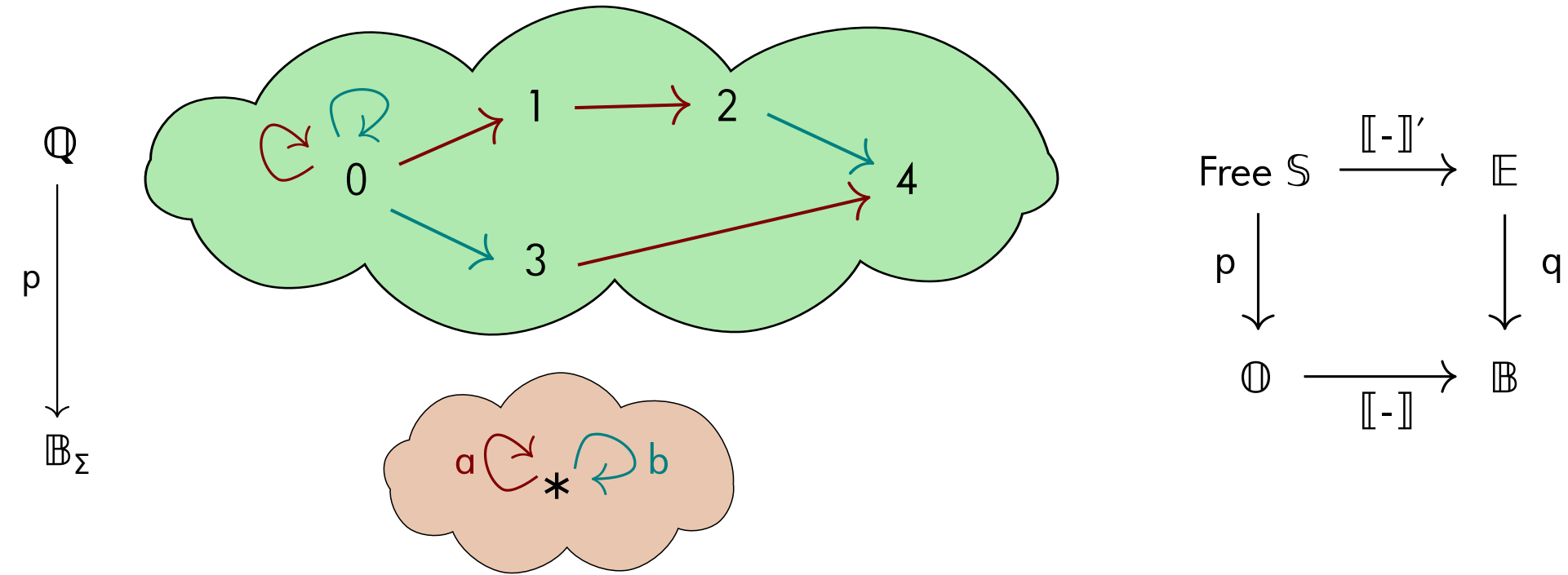
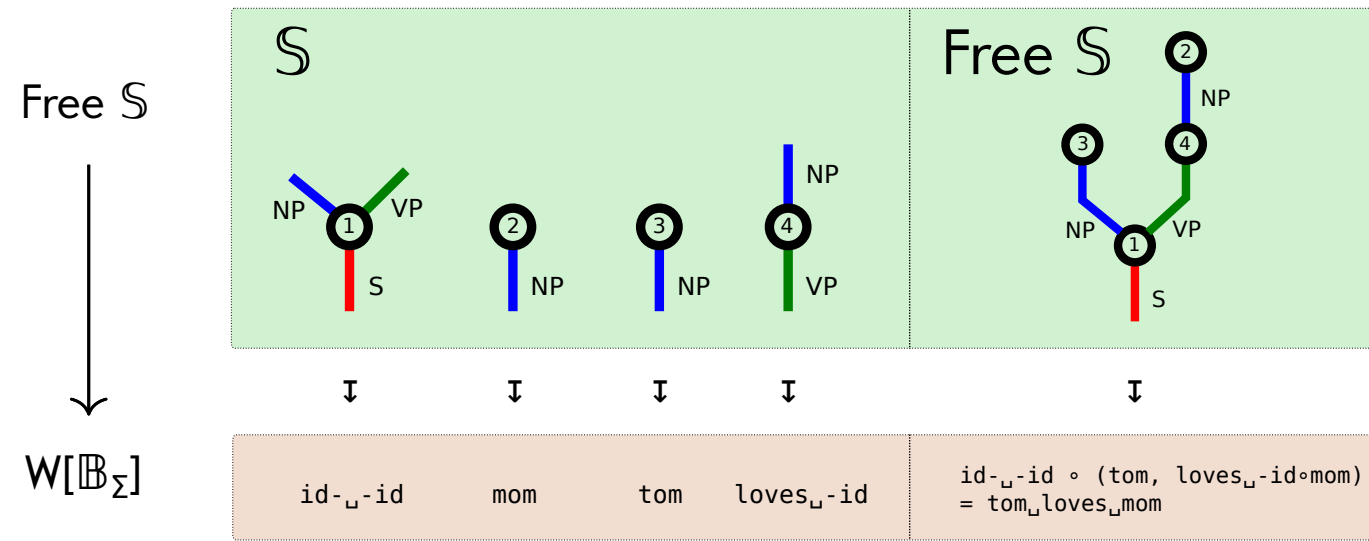


$$L_G = q L_{\mathcal{S}, \mathcal{S}} = q_{\text{nodes}} C[\varphi_{\text{colors}}] L_{\mathcal{S}, \mathcal{S}} = q_{\text{nodes}} (L_{\varphi_{\mathbb{C}} \mathcal{S}, \mathcal{S}} \cap L_{\mathbb{M} \text{colors}})$$

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

# ***Conclusion***

# Summary!



# Current / future directions

## Parsing (& typing & proving)

The MFPS paper includes brief discussion of generalized **CYK parsing**.  
One of our original goals (back in 2017!) was to understand **LR parsing**.  
We are revisiting LR & **Earley** parsing in the fresh light of contour categories.  
Hope to eventually gain better understanding of **type inference** and **proof search**.

## The contour / splicing adjunction

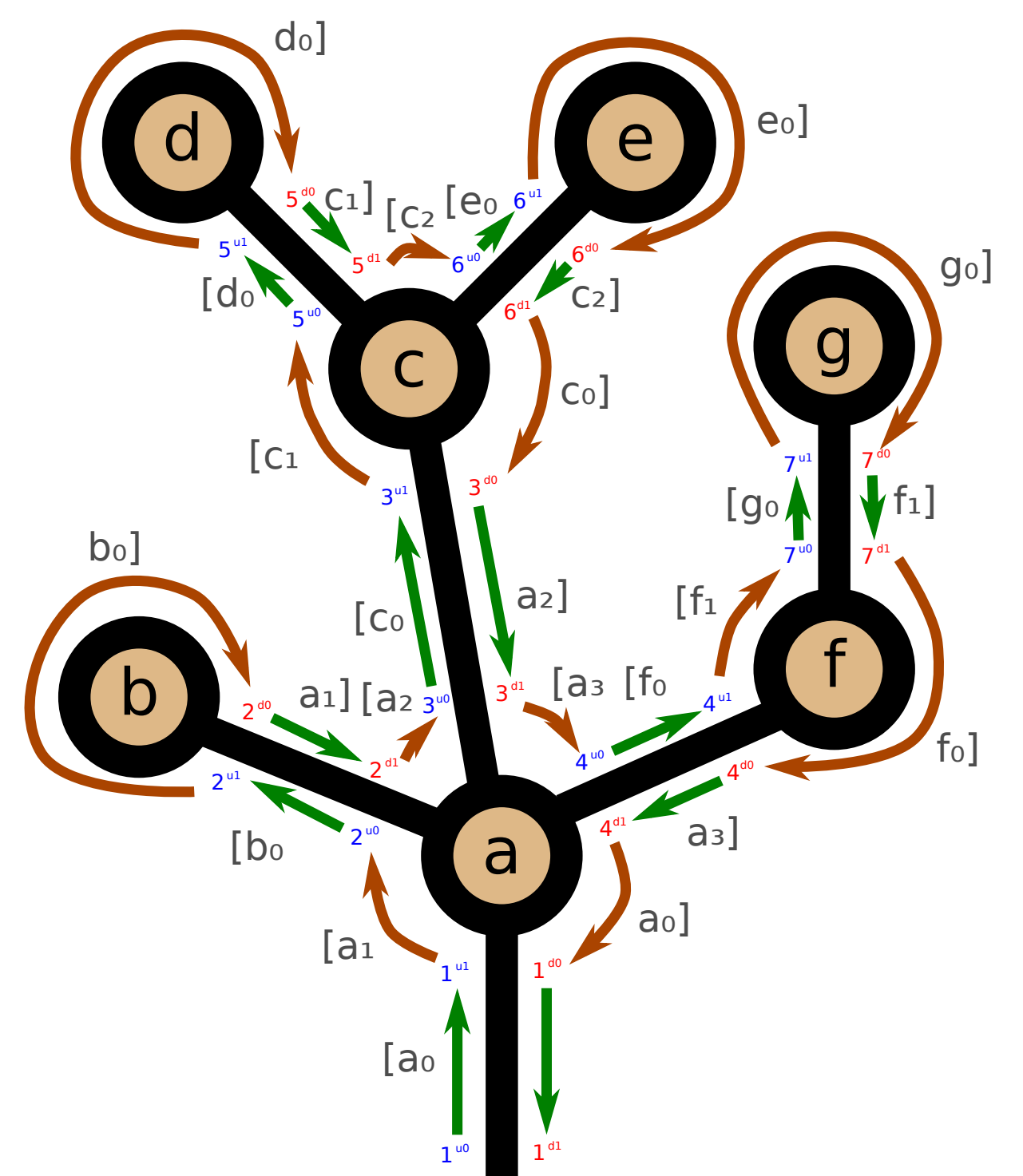
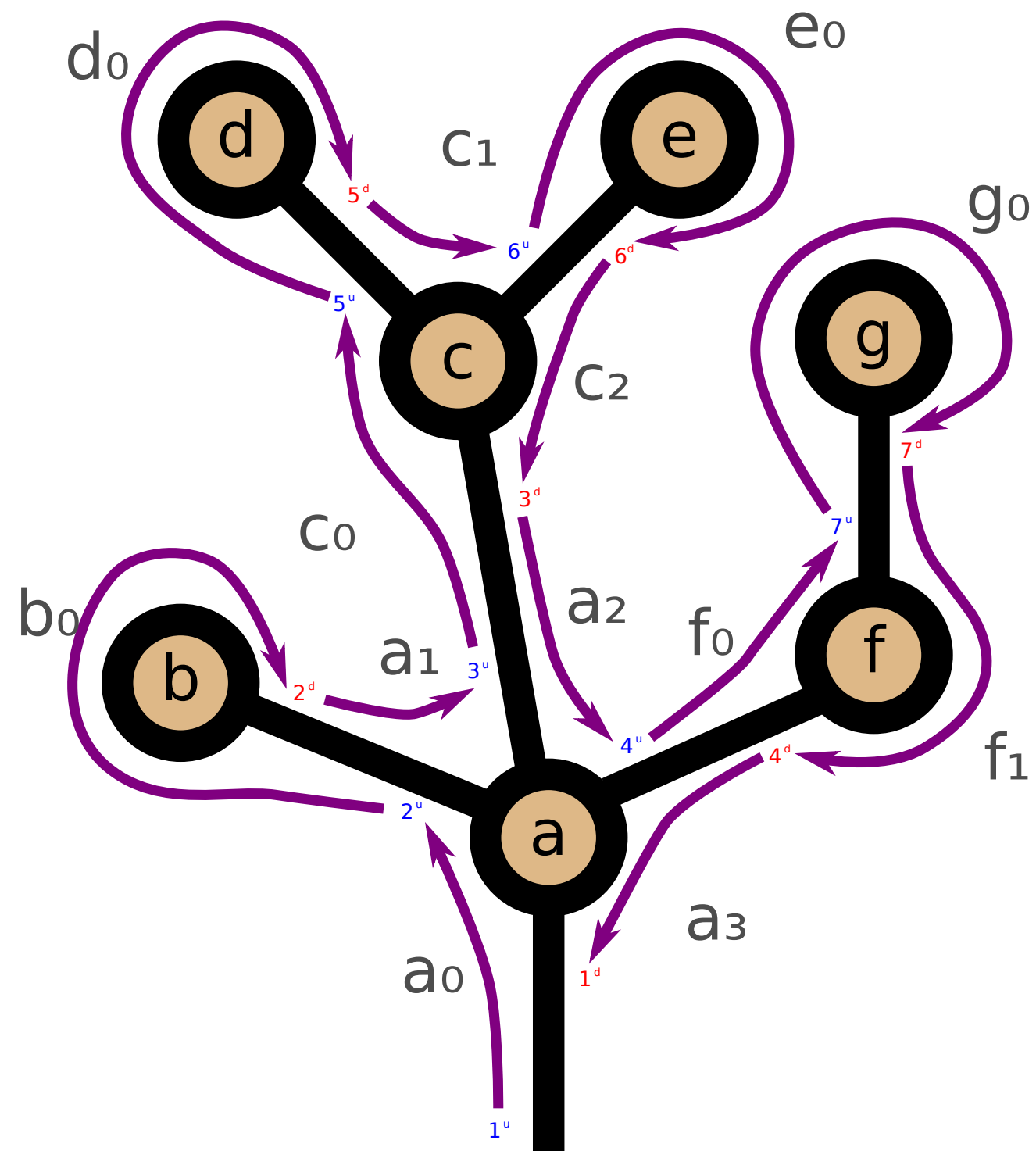
This seems to be a contribution of language theory to category theory!  
More to the story:  $W[\mathbb{C}]$  naturally extends to a **cyclic polycategory** (jww Peter Faul).  
Apparent link with permutation reps of graphs on surfaces (**combinatorial maps**).  
The adjunction has been recently extended and applied to **process algebra**.

Matt Earnshaw, James Hefford, Mario Román  
The Produoidal Algebra of Process Decomposition, arXiv:2301.11867

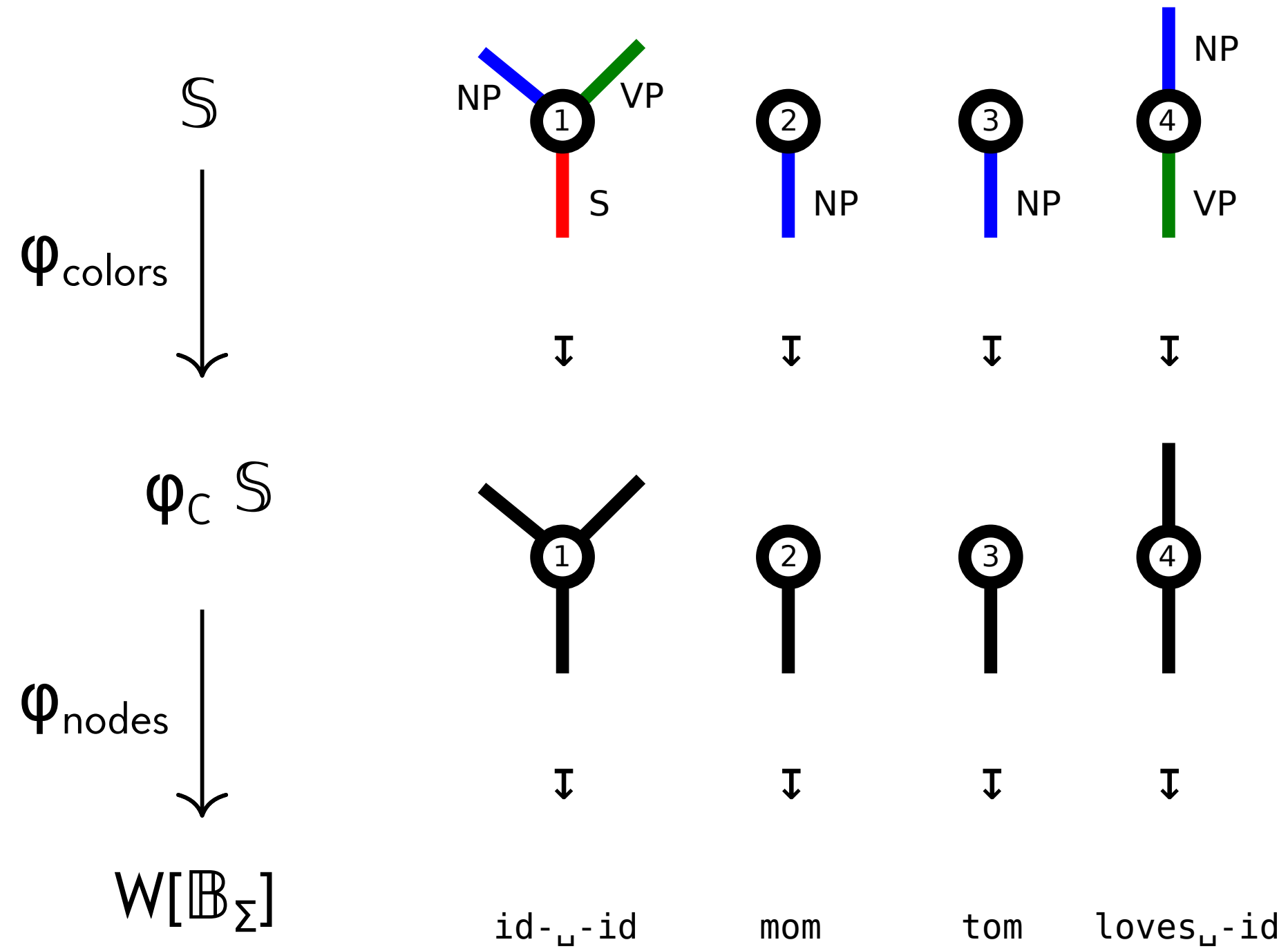


***Extra slides***

# From contour words to Dyck words



# Colors / nodes factorization



# Translation of corners

$$C[\varphi_c S] \longrightarrow \mathbb{B}_\Sigma$$

$1_0 \mapsto \text{id}$

$1_1 \mapsto \sqcup$

$1_2 \mapsto \text{id}$

$2_0 \mapsto \text{mom}$

$3_0 \mapsto \text{tom}$

$4_0 \mapsto \text{loves}_{\sqcup}$

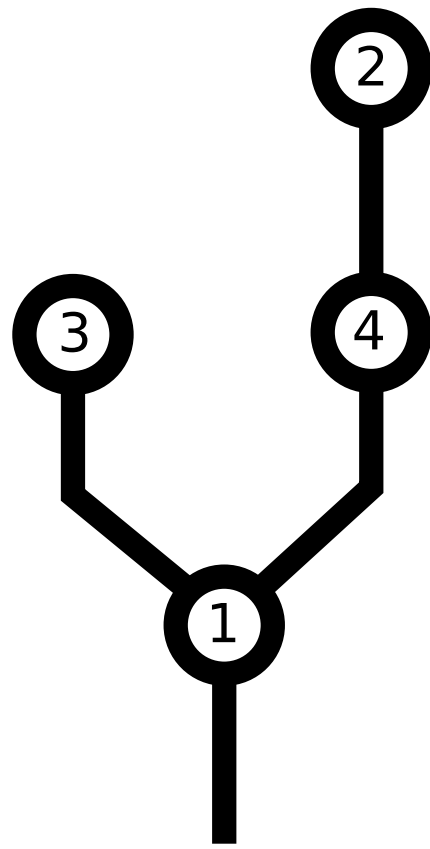
$4_1 \mapsto \text{id}$

# Uncolored tree contour words

Free  $\varphi_c S$

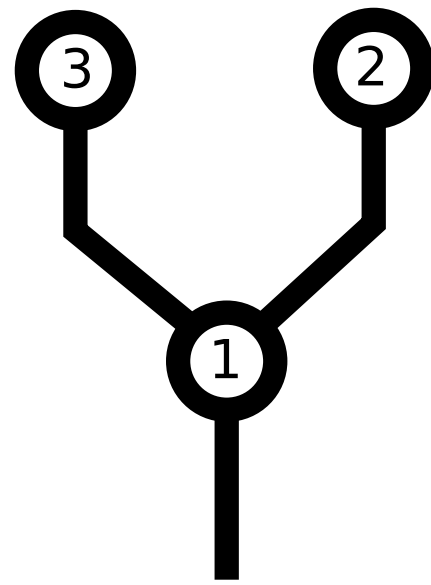


$W[\mathbb{B}_\Sigma]$



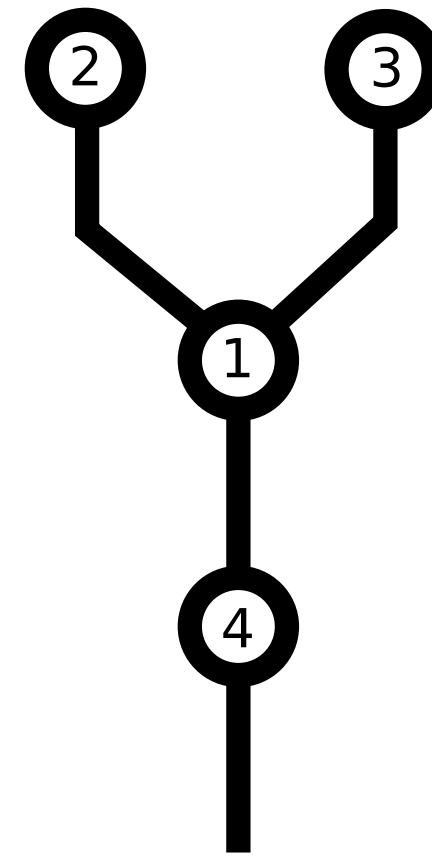
$1_0 3_0 1_1 4_0 2_0 4_1 1_2$

tom loves mom



$1_0 3_0 1_1 2_0 1_2$

tom mom

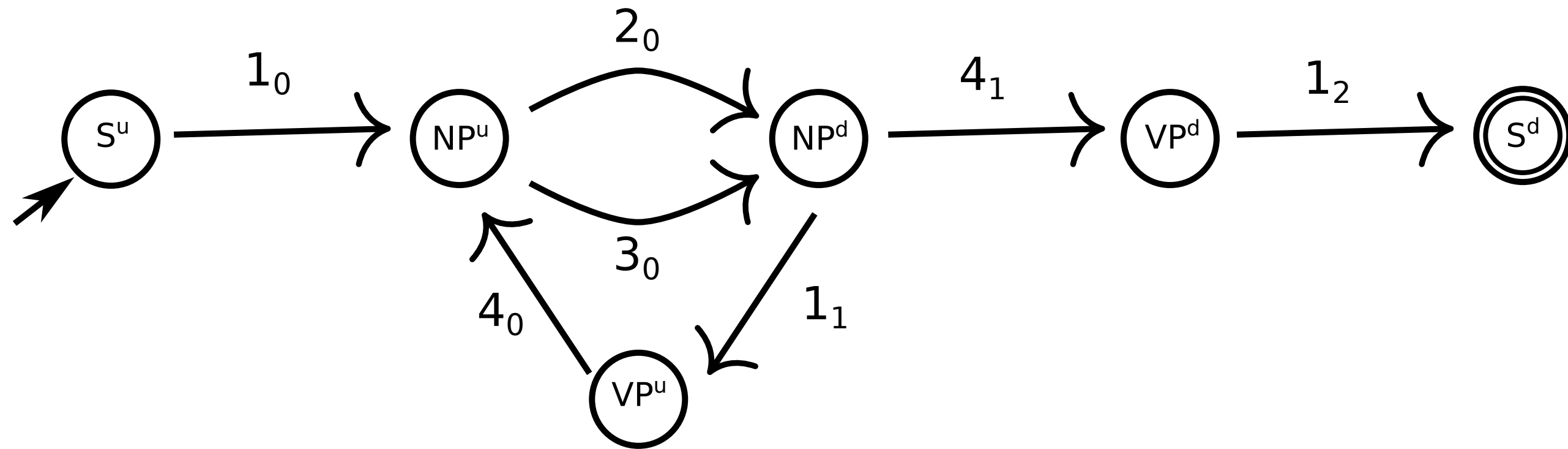


$4_0 1_0 2_0 1_1 3_0 1_2 4_1$

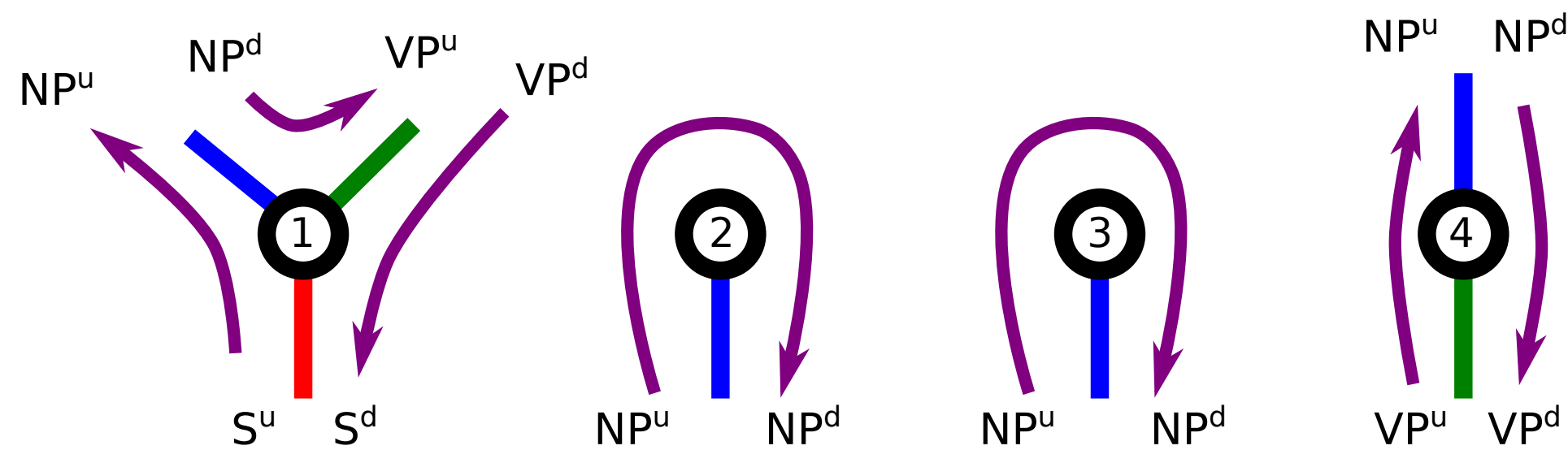
loves mom tom

...

# Coloring automaton



$C[S]$   
 $\downarrow$   
 $C[\varphi_c S]$



# Species (some terminology)

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

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

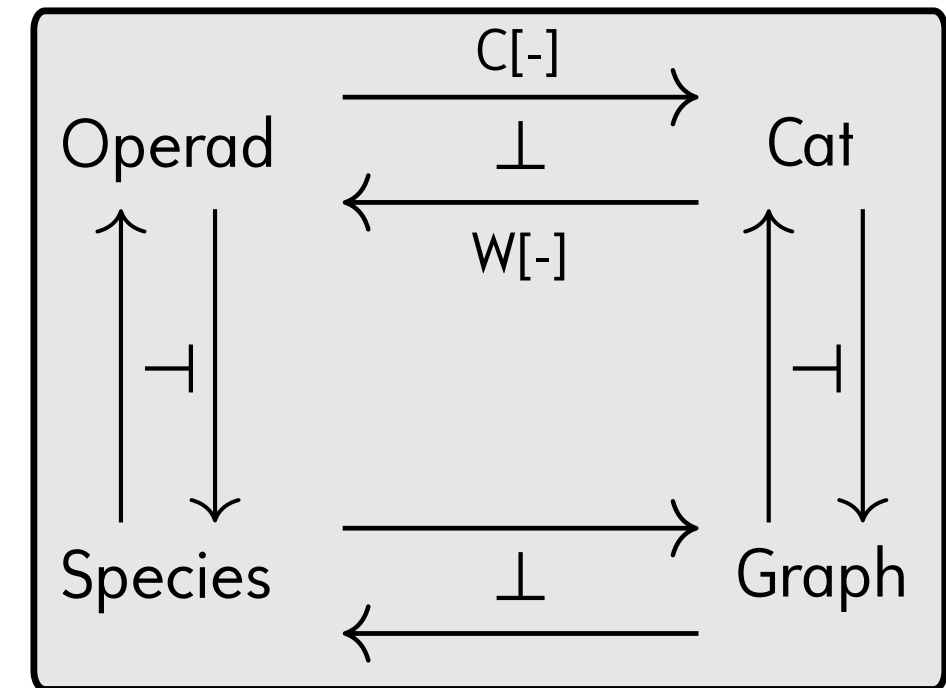
with the following interpretation:  $C$  is a set of "colors",  $V$  a set of "nodes", and  $i : V \rightarrow C^*$  and  $o : V \rightarrow C$  return respectively the list of input colors and the unique output color of each node. We say a species is **finite** (aka "polynomial") iff both  $C$  and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

$$\begin{array}{ccccc} C^* & \leftarrow^i & V & \xrightarrow{o} & C \\ \downarrow \varphi_{C^*} & & \downarrow \varphi_V & & \downarrow \varphi_C \\ D^* & \leftarrow^{i'} & W & \xrightarrow{o'} & D \end{array}$$

# Free contour categories

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

We sometimes write  $C[\mathcal{S}]$  as another name for this category.



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

\*Note that the word "corner" comes from the theory of planar maps, but in parsing theory, corners are called "dotted rules"!