Generalized context-free grammars over categories and operads

Noam Zeilberger[1]

Ecole Polytechnique (LIX)

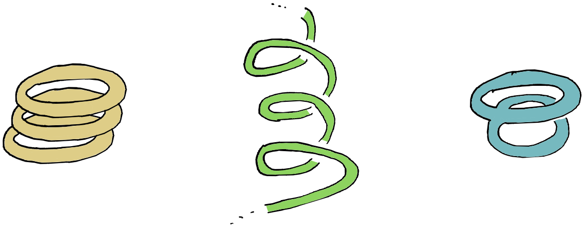
Séminaire d'informatique théorique
Université de Rouen Normandie, 20 November 2025

[1]Based on joint work with Paul-André Melliès. Main reference: *The categorical contours of the Chomsky-Schützenberger representation theorem*, LMCS 21:2, 2025.

0. Thinking fibrationally about deductive systems

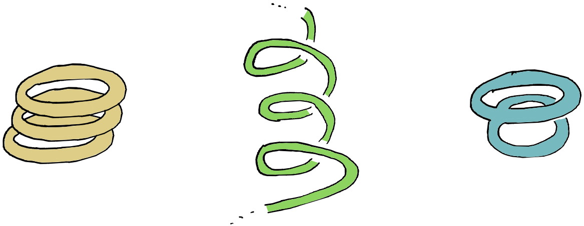
Topological intuitions

Imagine we are analyzing different spaces, more or less complicated.



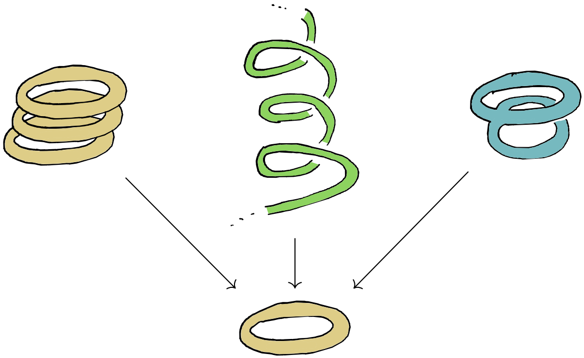
Topological intuitions

Imagine we are analyzing different spaces, more or less complicated. Rather than trying to study these spaces directly, it can be helpful to first project them down to some "simpler" space.



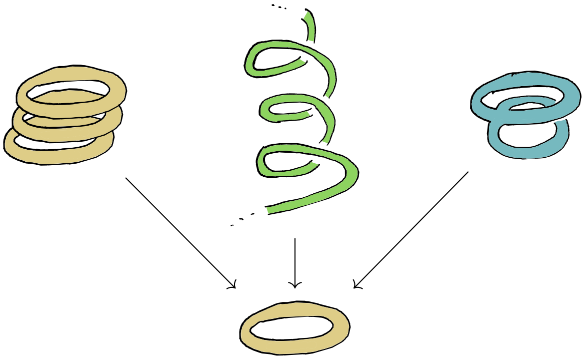
Topological intuitions

Imagine we are analyzing different spaces, more or less complicated. Rather than trying to study these spaces directly, it can be helpful to first project them down to some "simpler" space.



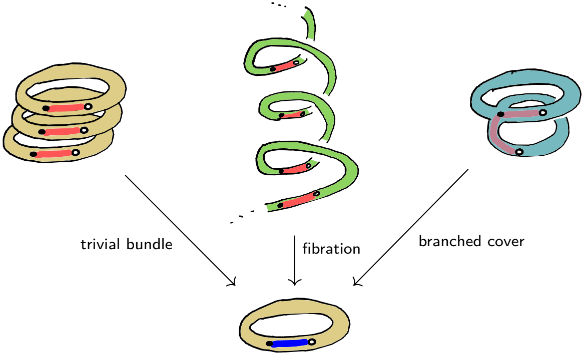
Topological intuitions

Imagine we are analyzing different spaces, more or less complicated. Rather than trying to study these spaces directly, it can be helpful to first project them down to some "simpler" space. We can learn something about our original spaces by the nature of these projections, and sometimes we can even reconstruct the spaces from their fibers.



Topological intuitions

Imagine we are analyzing different spaces, more or less complicated. Rather than trying to study these spaces directly, it can be helpful to first project them down to some "simpler" space. We can learn something about our original spaces by the nature of these projections, and sometimes we can even reconstruct the spaces from their fibers.



Fibrational perspectives on deductive systems

Bill Lawvere (late 1960s): the structure of predicate logic and the nature of quantifiers may be clarified by organizing proofs into bundles lying over spaces of formulas.

[Law69, Law70]. . .

Fibrational perspectives on deductive systems

Bill Lawvere (late 1960s): the structure of predicate logic and the nature of quantifiers may be clarified by organizing proofs into bundles lying over spaces of formulas.

[Law69, Law70]...

This idea may be extended to a variety of computational and deductive systems, including program logics, finite-state automata and context-free grammars.

...[Wal89, dG01, AGN05, MZ15, CP20, MZ25]...

Fibrational perspectives on deductive systems

Bill Lawvere (late 1960s): the structure of predicate logic and the nature of quantifiers may be clarified by organizing proofs into bundles lying over spaces of formulas.

[Law69, Law70]...

This idea may be extended to a variety of computational and deductive systems, including program logics, finite-state automata and context-free grammars.

...[Wal89, dG01, AGN05, MZ15, CP20, MZ25]...

Proof, recognition, and parsing may be thus reduced to lifting problems.

Functors as type refinement systems: some terminology and notation [MZ15]

We like to think of a category $\mathcal{D}$ equipped with a functor $p : \mathcal{D} \to \mathcal{C}$ as defining a kind of abstract type system, or "type refinement system". The idea is that the bundle $p : \mathcal{D} \to \mathcal{C}$ provides a source of *additional typing information* for the arrows of $\mathcal{C}$.

Functors as type refinement systems: some terminology and notation [MZ15]

We like to think of a category $\mathcal{D}$ equipped with a functor $p : \mathcal{D} \to \mathcal{C}$ as defining a kind of abstract type system, or "type refinement system". The idea is that the bundle $p : \mathcal{D} \to \mathcal{C}$ provides a source of *additional typing information* for the arrows of $\mathcal{C}$.

Let $R \in \mathcal{D}$ and $A \in \mathcal{C}$ be objects s.t. $p(R) = A$. We write $R \sqsubset A$ and say R **refines** A.

Functors as type refinement systems: some terminology and notation [MZ15]

We like to think of a category $\mathcal{D}$ equipped with a functor $p : \mathcal{D} \to \mathcal{C}$ as defining a kind of abstract type system, or "type refinement system". The idea is that the bundle $p : \mathcal{D} \to \mathcal{C}$ provides a source of *additional typing information* for the arrows of $\mathcal{C}$.

Let $R \in \mathcal{D}$ and $A \in \mathcal{C}$ be objects s.t. $p(R) = A$. We write $R \sqsubset A$ and say R **refines** A.

A **typing judgment** is a triple (S, f, T) of an arrow $f : A \to B$ of $\mathcal{C}$ and a pair of objects $S \sqsubset A$ and $T \sqsubset B$ of $\mathcal{D}$ refining its domain and codomain.

Functors as type refinement systems: some terminology and notation [MZ15]

We like to think of a category $\mathcal{D}$ equipped with a functor $p : \mathcal{D} \to \mathcal{C}$ as defining a kind of abstract type system, or "type refinement system". The idea is that the bundle $p : \mathcal{D} \to \mathcal{C}$ provides a source of *additional typing information* for the arrows of $\mathcal{C}$.

Let $R \in \mathcal{D}$ and $A \in \mathcal{C}$ be objects s.t. $p(R) = A$. We write $R \sqsubset A$ and say R **refines** A.

A **typing judgment** is a triple (S, f, T) of an arrow $f : A \to B$ of $\mathcal{C}$ and a pair of objects $S \sqsubset A$ and $T \sqsubset B$ of $\mathcal{D}$ refining its domain and codomain.

A **derivation** of a typing judgment (S, f, T) is an arrow $\alpha : S \to T$ of $\mathcal{D}$ s.t. $p(\alpha) = f$.

Functors as type refinement systems: some intuition

$$\begin{tikzcd} \mathcal{D} \arrow[d, "p"'] \\ \mathcal{C} \end{tikzcd} \qquad\qquad A \xrightarrow{f} B$$

(f a program taking input of type A and producing output of type B)

Functors as type refinement systems: some intuition

$$
\begin{tikzcd}
\mathcal{D} \arrow[d, "p"'] & S \arrow[d, dash, dashed] & T \arrow[d, dash, dashed] \\
\mathcal{C} & A \arrow[r, "f"] & B
\end{tikzcd}
$$

(S and T predicates refining A and B)

Functors as type refinement systems: some intuition

$$
\begin{tikzcd}
\mathcal{D} \arrow[d, "p"'] & & S \arrow[r, "\alpha"] \arrow[d, dash, dashed] & T \arrow[d, dash, dashed] \\
\mathcal{C} & & A \arrow[r, "f"] & B
\end{tikzcd}
$$

(α a proof that f has a more refined type $S \to T$)

From type systems to languages

Given a functor $p : \mathcal{D} \to \mathcal{C}$ and objects $S \sqsubset A$ and $T \sqsubset B$, the **language of arrows** induced by (p, S, T) is defined as

$$\mathcal{L}_{p,S,T} := \{ p(\alpha) \mid \alpha : S \to T \} \subseteq \mathcal{C}(A, B).$$

We may think of $\mathcal{L}_{p,S,T}$ as the set of programs $f : A \to B$ such that (S, f, T) holds.

Ambiguity

A judgment (S, f, T) may in general have more than one derivation, in other words, a type refinement system may be *ambiguous*.

By definition, $p : \mathcal{D} \to \mathcal{C}$ is **unambiguous** iff p is faithful, i.e., for any pair of arrows $\alpha, \alpha' : S \to T$ with the same source and target, if $p(\alpha) = p(\alpha')$ then $\alpha = \alpha'$.

When p is potentially ambiguous, it can be interesting to consider $\mathcal{L}_{p,S,T}$ as a "proof-relevant language", that is, as a bundle of homsets

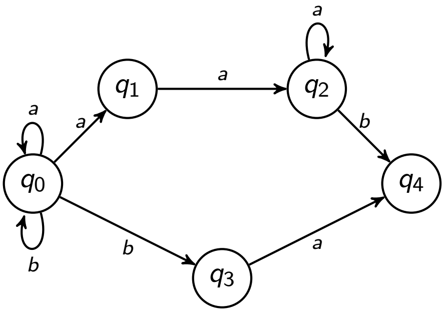
$$\mathcal{L}_{p,S,T} := \mathcal{D}(S, T) \longrightarrow \mathcal{C}(A, B).$$

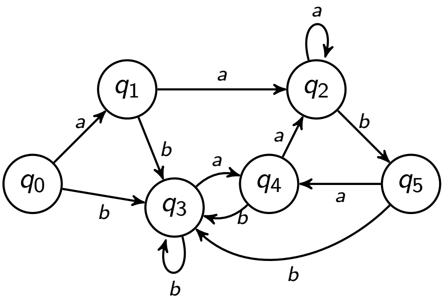
1. Finite-state automata as functors

Automata as graph homomorphisms

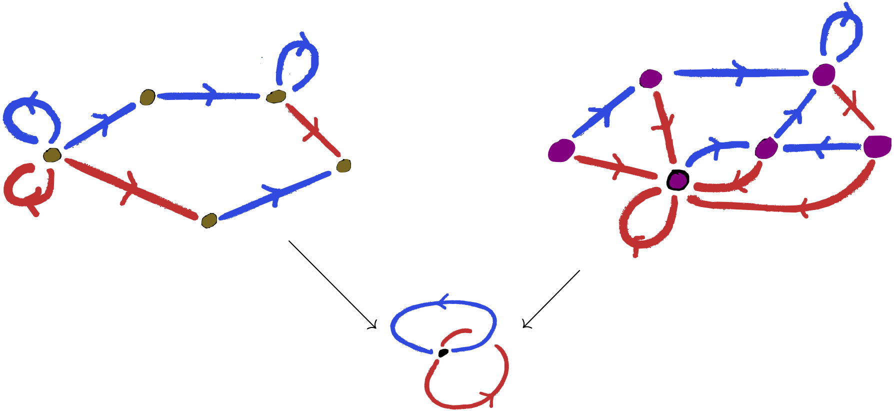
The underlying transition graph of any NFA (without ϵ-transitions) over the alphabet Σ is entirely described by a graph homomorphism $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ into the *bouquet graph* with one node $*$ and a loop $a : * \to *$ for every $a \in \Sigma$.

Automata as graph homomorphisms

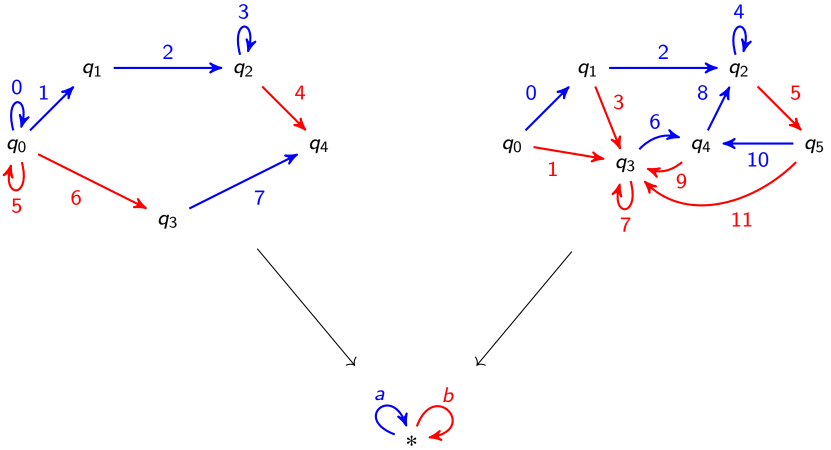




Automata as graph homomorphisms



Automata as graph homomorphisms



Free categories

To any graph $\mathbb{G}$ is associated a **free category** $\mathcal{F}\mathbb{G}$ whose objects are nodes and whose arrows are paths. For example, the free category over

$$\mathbb{G} \quad = \quad \begin{tikzcd} & B \arrow[dr, "b"] & & & \\ A \arrow[ur, "a"] \arrow[dr, "c"'] & & D \arrow[r, "e"] & E \arrow[r, bend left, "f"] & F \arrow[l, bend left, "g"] \\ & C \arrow[ur, "d"'] & & & \end{tikzcd}$$

has $\hom(A, D) = \{ ab, cd \}$ and $\hom(E, E) = (fg)^*$.

Universal property of free categories: any functor $\mathcal{F}\mathbb{G} \to \mathcal{C}$ into a category $\mathcal{C}$ is uniquely determined by a graph homomorphism $\mathbb{G} \to \mathcal{C}$ into the underlying graph of $\mathcal{C}$.

Recognition as a path lifting problem

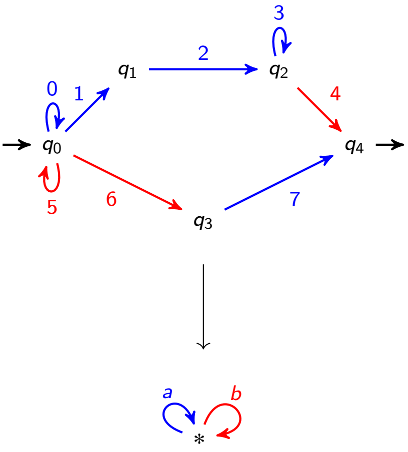
Any *word* $w \in \Sigma^*$ corresponds to a *path* in $\mathbb{B}_\Sigma$, i.e., to an arrow $* \to *$ in $\mathcal{F}\mathbb{B}_\Sigma$.

Any graph homomorphism $\phi : \mathbb{G} \to \mathbb{H}$ induces a corresponding functor between free categories $\mathcal{F}\phi : \mathcal{F}\mathbb{G} \to \mathcal{F}\mathbb{H}$, sending paths in $\mathbb{G}$ to paths in $\mathbb{H}$ of the same length.

Let $\mathcal{A}$ be an NFA with transition graph $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ and associated functor $p = \mathcal{F}\phi$. Then $\mathcal{A}$ accepts $w \in \Sigma^*$ just in case there is an arrow $\alpha : q_0 \to q_f$ in $\mathcal{F}\mathbb{G}$ such that $p(\alpha) = w$, from an initial state q_0 to an accepting state q_f.

Recognition as a path lifting problem

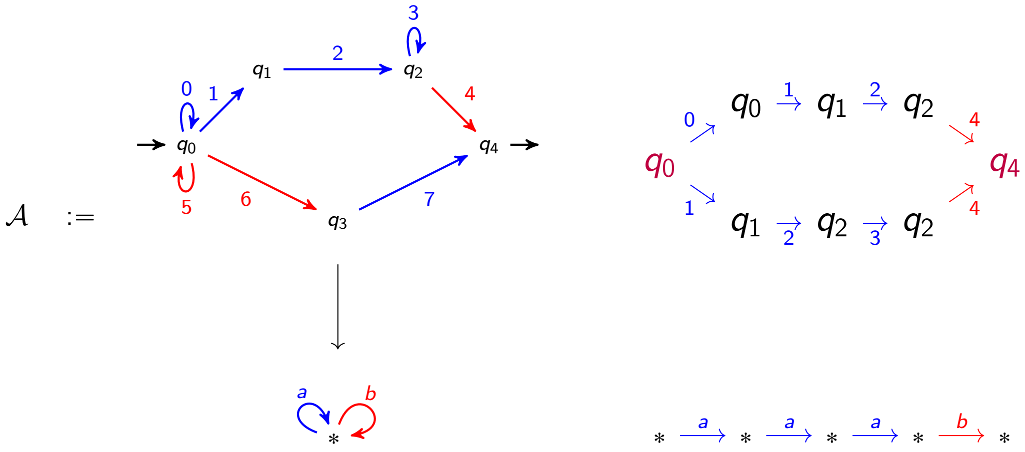
$\mathcal{A} \quad :=$



$q_0 \qquad\qquad\qquad\qquad q_4$

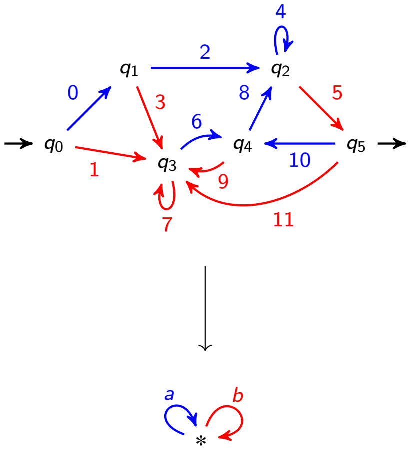
$* \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{b} *$

Recognition as a path lifting problem



Recognition as a path lifting problem

$\mathcal{A} \quad :=$

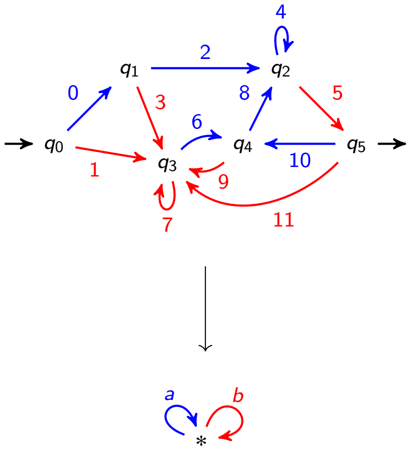


q_0

$$* \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{b} *$$

Recognition as a path lifting problem

$\mathcal{A} \quad :=$

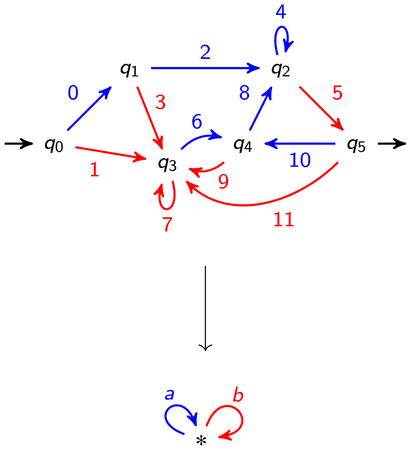


$$q_0 \xrightarrow{0} q_1$$

$$* \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{b} *$$

Recognition as a path lifting problem

$\mathcal{A} \quad :=$

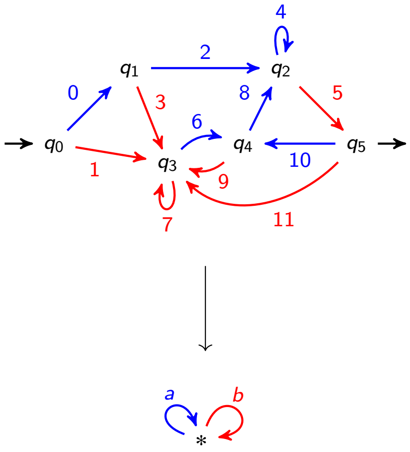


$$q_0 \xrightarrow{0} q_1 \xrightarrow{2} q_2$$

$$* \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{b} *$$

Recognition as a path lifting problem

$\mathcal{A} \quad :=$

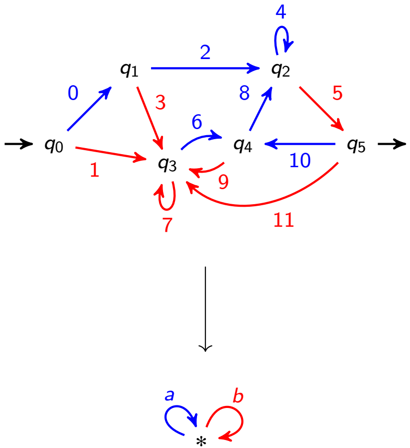


$$q_0 \xrightarrow{0} q_1 \xrightarrow{2} q_2 \xrightarrow{4} q_2$$

$$* \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{b} *$$

Recognition as a path lifting problem

$\mathcal{A} \quad :=$



$$q_0 \xrightarrow{0} q_1 \xrightarrow{2} q_2 \xrightarrow{4} q_2 \xrightarrow{5} q_5$$

$$* \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{a} * \xrightarrow{b} *$$

Fibrational properties

A homomorphism of finite graphs $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ represents the transition graph of a *complete DFA* just in case the functor $\mathcal{F}\phi : \mathcal{F}\mathbb{G} \to \mathcal{F}\mathbb{B}_\Sigma$ is a *discrete opfibration*. Such functors are in 1-to-1 correspondence with covariant presheaves $\mathcal{F}\mathbb{B}_\Sigma \to \mathrm{FinSet}$.

Fibrational properties

A homomorphism of finite graphs $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ represents the transition graph of a *complete DFA* just in case the functor $\mathcal{F}\phi : \mathcal{F}\mathbb{G} \to \mathcal{F}\mathbb{B}_\Sigma$ is a *discrete opfibration*. Such functors are in 1-to-1 correspondence with covariant presheaves $\mathcal{F}\mathbb{B}_\Sigma \to \mathrm{FinSet}$.

More generally, a functor $p : \mathcal{D} \to \mathcal{F}\mathbb{B}_\Sigma$ represents the transition graph of an NFA just in case it has *finite fibers* and *unique lifting of factorizations*.

Fibrational properties

A homomorphism of finite graphs $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ represents the transition graph of a *complete DFA* just in case the functor $\mathcal{F}\phi : \mathcal{F}\mathbb{G} \to \mathcal{F}\mathbb{B}_\Sigma$ is a *discrete opfibration*. Such functors are in 1-to-1 correspondence with covariant presheaves $\mathcal{F}\mathbb{B}_\Sigma \to \mathrm{FinSet}$.

More generally, a functor $p : \mathcal{D} \to \mathcal{F}\mathbb{B}_\Sigma$ represents the transition graph of an NFA just in case it has *finite fibers* and *unique lifting of factorizations*. Such functors are in 1-to-1 correspondence with covariant presheaves valued in the *bicategory of spans of finite sets*, that is, (pseudo)functors $\mathcal{F}\mathbb{B}_\Sigma \to \mathrm{FinSpan}$.

Fibrational properties

A homomorphism of finite graphs $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ represents the transition graph of a *complete DFA* just in case the functor $\mathcal{F}\phi : \mathcal{F}\mathbb{G} \to \mathcal{F}\mathbb{B}_\Sigma$ is a *discrete opfibration*. Such functors are in 1-to-1 correspondence with covariant presheaves $\mathcal{F}\mathbb{B}_\Sigma \to \mathrm{FinSet}$.

More generally, a functor $p : \mathcal{D} \to \mathcal{F}\mathbb{B}_\Sigma$ represents the transition graph of an NFA just in case it has *finite fibers* and *unique lifting of factorizations*. Such functors are in 1-to-1 correspondence with covariant presheaves valued in the *bicategory of spans of finite sets*, that is, (pseudo)functors $\mathcal{F}\mathbb{B}_\Sigma \to \mathrm{FinSpan}$.

This suggests two different ways of generalizing automata to arbitrary base categories:

1. As functors $p : \mathcal{F}\mathbb{G} \to \mathcal{C}$ from a finitely generated free category.
2. As finitary ULF functors $p : \mathcal{Q} \to \mathcal{C}$.

We will eventually take position (2), but adapt (1) for CFGs...

2. Context-free grammars as functors

From categories to operads

An operad (aka multicategory) is like a category where arrows can take multiple objects as input.

$$f : A_1, \ldots, A_n \to B \qquad (n \geqslant 0)$$

(We work with *planar* operads, which have a linear order on inputs, and no exchange.)

Any functor of operads $p : \mathcal{D} \to \mathcal{O}$ equipped with an object $S \sqsubset A$ induces a **language of constants** defined by

$$\mathcal{L}_{p,S} := \{\, p(\alpha) \mid \alpha : S \,\} \subseteq \mathcal{O}(\cdot; A).$$

Parsing as a lifting problem

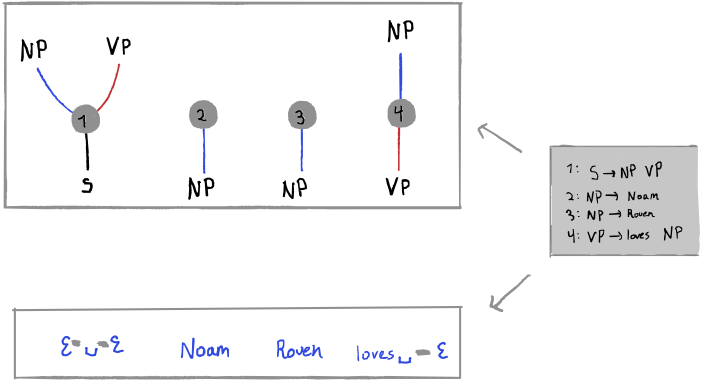
$S \to NP\ VP$

$NP \to$ Noam

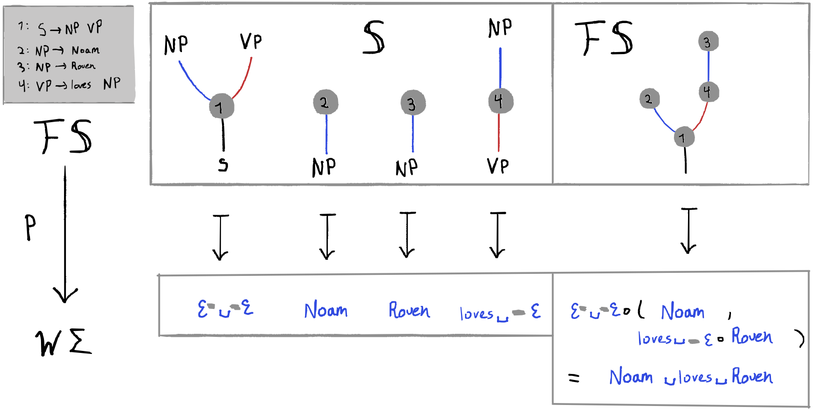
$NP \to$ Rouen

$VP \to$ loves NP

Parsing as a lifting problem



Parsing as a lifting problem



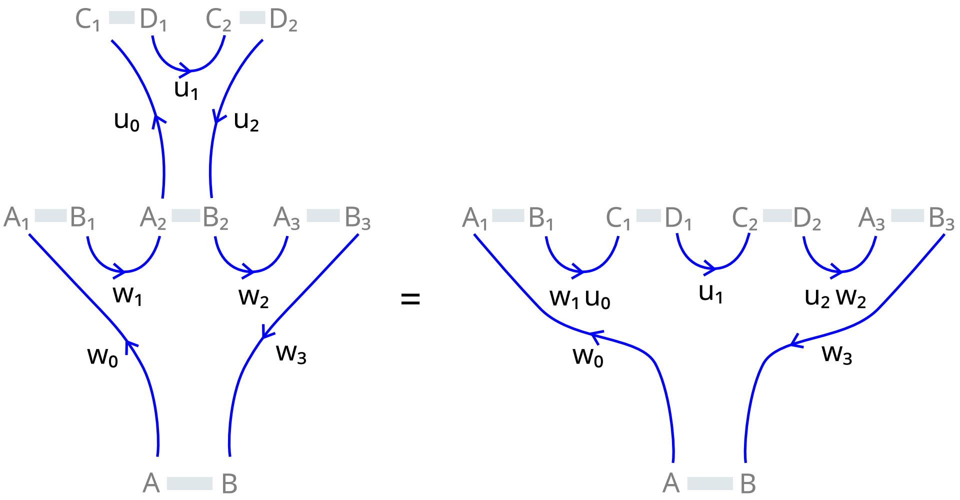
Cf. RFC Walters, "A note on context-free grammars", JPAA 62(2): 199-203, 1989.

The splicing construction

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

- objects are pairs (A, B) of objects of $\mathcal{C}$;
- n-operations $f : (A_1, B_1), \ldots, (A_n, B_n) \to (A, B)$ given by sequences $f = w_0 - w_1 - \ldots - w_n$ of $n+1$ arrows $w_i : B_i \to A_{i+1}$ in $\mathcal{C}$ ($B_0 = A$, $A_{n+1} = B$);
- composition performed by “splicing into the gaps”...

The splicing construction



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

Context-free grammar over a category

A **CFG over a category** $\mathcal{C}$ is a tuple $G = (\dot{\mathbb{S}}, p)$ consisting of a pointed finite species $\dot{\mathbb{S}} = (\mathbb{S}, S \in \mathbb{S})$ and a functor $p : \mathcal{F}\mathbb{S} \to \mathcal{W}\mathcal{C}$. The context-free language generated by G is the set $\mathcal{L}_G := \mathcal{L}_{p,S} = \{\, p(\alpha) \mid \alpha : S \,\} \subseteq \mathcal{C}(A, B)$ where $p(S) = (A, B)$.

Context-free grammar over a category

A **CFG over a category** $\mathcal{C}$ is a tuple $G = (\dot{\mathbb{S}}, p)$ consisting of a pointed finite species $\dot{\mathbb{S}} = (\mathbb{S}, S \in \mathbb{S})$ and a functor $p : \mathcal{F}\mathbb{S} \to \mathcal{W}\mathcal{C}$. The context-free language generated by G is the set $\mathcal{L}_G := \mathcal{L}_{p,S} = \{\, p(\alpha) \mid \alpha : S \,\} \subseteq \mathcal{C}(A, B)$ where $p(S) = (A, B)$.

Example: a classical CFG is just a CFG over $\mathcal{F}\mathbb{B}_\Sigma$.

Context-free grammar over a category

A **CFG over a category** $\mathcal{C}$ is a tuple $G = (\dot{\mathbb{S}}, p)$ consisting of a pointed finite species $\dot{\mathbb{S}} = (\mathbb{S}, S \in \mathbb{S})$ and a functor $p : \mathcal{F}\mathbb{S} \to \mathcal{W}\mathcal{C}$. The context-free language generated by G is the set $\mathcal{L}_G := \mathcal{L}_{p,S} = \{ p(\alpha) \mid \alpha : S \} \subseteq \mathcal{C}(A, B)$ where $p(S) = (A, B)$.

Example: a classical CFG is just a CFG over $\mathcal{F}\mathbb{B}_\Sigma$.

Example: let $\mathbb{B}_\Sigma^{\hat{}\$} :=$
$$\begin{tikzcd} \perp \arrow[r, "\hat{}"] & * \arrow[loop above, "a\in\Sigma"] \arrow[r, "\$"] & \top \end{tikzcd}$$
. A CFG over $\mathcal{F}\mathbb{B}_\Sigma^{\hat{}\$}$ can have productions that are only applicable at the beginning/end of the string.

3. Generalized NFAs and CFGs over operads

Definitions

Let $\mathcal{O}$ be any operad.

A **NFA over** $\mathcal{O}$ is a tuple $M = (\dot{\mathcal{Q}}, p)$ of a pointed operad $(\mathcal{Q}, q_r \in \mathcal{Q})$ and a functor $p : \mathcal{Q} \to \mathcal{O}$ satisfying the finite fiber and ULF properties.

Definitions

Let $\mathcal{O}$ be any operad.

A **NFA over** $\mathcal{O}$ is a tuple $M = (\dot{\mathcal{Q}}, p)$ of a pointed operad $(\mathcal{Q}, q_r \in \mathcal{Q})$ and a functor $p : \mathcal{Q} \to \mathcal{O}$ satisfying the finite fiber and ULF properties.

Example: a nondeterministic tree automaton on a graded alphabet Σ is just an NFA over the free operad $\mathcal{F}\Sigma$.

Definitions

Let $\mathcal{O}$ be any operad.

A **NFA over** $\mathcal{O}$ is a tuple $M = (\dot{\mathcal{Q}}, p)$ of a pointed operad $(\mathcal{Q}, q_r \in \mathcal{Q})$ and a functor $p : \mathcal{Q} \to \mathcal{O}$ satisfying the finite fiber and ULF properties.

Example: a nondeterministic tree automaton on a graded alphabet Σ is just an NFA over the free operad $\mathcal{F}\Sigma$.

A **CFG over** $\mathcal{O}$ is a tuple $G = (\dot{\mathbb{S}}, p)$ of a pointed finite species $\dot{\mathbb{S}} = (\mathbb{S}, S \in \mathbb{S})$ and an arbitrary functor $p : \mathcal{F}\mathbb{S} \to \mathcal{O}$.

Definitions

Let $\mathcal{O}$ be any operad.

A **NFA over** $\mathcal{O}$ is a tuple $M = (\dot{\mathcal{Q}}, p)$ of a pointed operad $(\mathcal{Q}, q_r \in \mathcal{Q})$ and a functor $p : \mathcal{Q} \to \mathcal{O}$ satisfying the finite fiber and ULF properties.

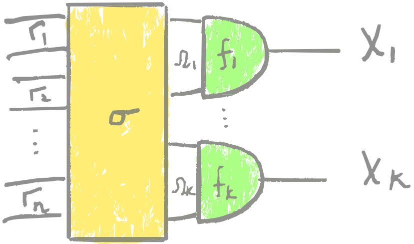
Example: a nondeterministic tree automaton on a graded alphabet Σ is just an NFA over the free operad $\mathcal{F}\Sigma$.

A **CFG over** $\mathcal{O}$ is a tuple $G = (\dot{\mathbb{S}}, p)$ of a pointed finite species $\dot{\mathbb{S}} = (\mathbb{S}, S \in \mathbb{S})$ and an arbitrary functor $p : \mathcal{F}\mathbb{S} \to \mathcal{O}$.

Example: k-multiple and k-parallel CFGs in the sense of Seki et al. (1991) are CFGs over free semi-cartesian / free cartesian operads. . .

A few constructions on operads

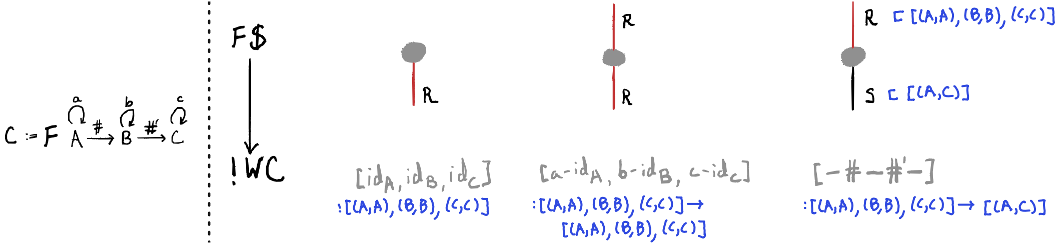
Given an operad $\mathcal{O}$, the *free symmetric monoidal operad* over $\mathcal{O}$ is the operad $!_{\mathrm{sym}}\mathcal{O}$ whose objects are lists $[X_1, \ldots, X_k]$ of objects of $\mathcal{O}$, and whose *n*-ary operations $[\Gamma_1], \ldots, [\Gamma_n] \to [X_1, \ldots, X_k]$ are pairs $([f_1, \ldots, f_k], \sigma)$ of a list of operations $f_1 : \Omega_1 \to X_1, \ldots, f_k : \Omega_k \to X_k$ and a bijection $\sigma : \Omega_1, \ldots, \Omega_k \xrightarrow{\sim} \Gamma_1, \ldots, \Gamma_n$.



The free semi-cartesian monoidal operad $!_{\mathrm{aff}}\mathcal{O}$ / free cartesian monoidal operad $!_{\mathrm{cart}}\mathcal{O}$ are defined in the same way but letting σ be any injection / function.

Generating mildly context-sensitive languages

The following 3-mCFG generates the language $a^n \# b^n \#' c^n$.



Closure properties of generalized CFLs

1. If $\mathcal{L}_1, \ldots, \mathcal{L}_k \subseteq \mathcal{O}(A)$ are context-free, then so is their union $\bigcup_{i=1}^{k} \mathcal{L}_i \subseteq \mathcal{O}(A)$.
2. If $\mathcal{L}_1 \subseteq \mathcal{O}(A_1), \ldots, \mathcal{L}_n \subseteq \mathcal{O}(A_n)$ are CF, and if $f : A_1, \ldots, A_n \to A$ is an operation of $\mathcal{O}$, then $\{ f(u_1, \ldots, u_n) \mid u_1 \in \mathcal{L}_1, \ldots, u_n \in \mathcal{L}_n \} \subseteq \mathcal{O}(A)$ is CF.
3. If $\mathcal{L} \subseteq \mathcal{O}(A)$ is CF and $F : \mathcal{O} \to \mathcal{P}$ is a functor of operads, then the functorial image $F(L) \subseteq \mathcal{P}(F(A))$ is also CF.
4. If $\mathcal{L} \subseteq \mathcal{O}(A)$ is CF and $\mathcal{R} \subseteq \mathcal{O}(A)$ is regular, then $\mathcal{L} \cap \mathcal{R} \subseteq \mathcal{O}(A)$ is CF.

Closure properties of generalized CFLs

1. If $\mathcal{L}_1, \ldots, \mathcal{L}_k \subseteq \mathcal{O}(A)$ are context-free, then so is their union $\bigcup_{i=1}^{k} \mathcal{L}_i \subseteq \mathcal{O}(A)$.
2. If $\mathcal{L}_1 \subseteq \mathcal{O}(A_1), \ldots, \mathcal{L}_n \subseteq \mathcal{O}(A_n)$ are CF, and if $f : A_1, \ldots, A_n \to A$ is an operation of $\mathcal{O}$, then $\{\, f(u_1, \ldots, u_n) \mid u_1 \in \mathcal{L}_1, \ldots, u_n \in \mathcal{L}_n \,\} \subseteq \mathcal{O}(A)$ is CF.
3. If $\mathcal{L} \subseteq \mathcal{O}(A)$ is CF and $F : \mathcal{O} \to \mathcal{P}$ is a functor of operads, then the functorial image $F(L) \subseteq \mathcal{P}(F(A))$ is also CF.
4. If $\mathcal{L} \subseteq \mathcal{O}(A)$ is CF and $\mathcal{R} \subseteq \mathcal{O}(A)$ is regular, then $\mathcal{L} \cap \mathcal{R} \subseteq \mathcal{O}(A)$ is CF.

(4) is a corollary of the following pullback construction: given a gCFG $G = (\dot{\mathbb{S}}, p_G)$ and a gNFA $M = (\dot{\mathcal{Q}}, p_M)$ with $p_G(S) = p_M(q_r)$, we can construct a gCFG $M^{-1}(G)$ over $\mathcal{Q}$ generating the language $p_M^{-1}(\mathcal{L}_G) \cap \mathcal{Q}(q_r)$. (cf. the Bar-Hillel construction)

4. Perspectives

Perspectives

Context-free grammars and finite-state automata are complementary notions from formal language theory.

Perspectives

Context-free grammars and finite-state automata are complementary notions from formal language theory. Their representation (and generalization) as functors of operads seems to have some explanatory power.

Perspectives

Context-free grammars and finite-state automata are complementary notions from formal language theory. Their representation (and generalization) as functors of operads seems to have some explanatory power.

Current projects include extending the splicing construction to account for context-free languages beyond languages of words (cf. Matthew Earnshaw's PhD thesis), as well as developing the notion of "proof-relevant language".

Perspectives

Context-free grammars and finite-state automata are complementary notions from formal language theory. Their representation (and generalization) as functors of operads seems to have some explanatory power.

Current projects include extending the splicing construction to account for context-free languages beyond languages of words (cf. Matthew Earnshaw's PhD thesis), as well as developing the notion of "proof-relevant language".

A long term goal is to transfer knowledge between category theory, automata theory, parsing & linguistics, type systems...

Very selected bibliography
(see [MZ25] for more)

[AGN05] Samson Abramsky, Simon Gay, and Rajagopal Nagarajan. Specification structures and propositions-as-types for concurrency. In *Logics for Concurrency: Structure versus Automata*, pages 5–40. Springer, 2005.

[CP20] Thomas Colcombet and Daniela Petrişan. Automata minimization: a functorial approach. *Logical Methods in Computer Science*, 16(1):32:1–32:28, 2020.

[dG01] Philippe de Groote. Towards abstract categorial grammars. In *Association for Computational Linguistic, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference, July 9-11, 2001, Toulouse, France*, pages 148–155. Morgan Kaufmann Publishers, 2001.

[Law69] F. W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.

[Law70] F. W. Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. In *Proceedings of the AMS Symposium on Pure Mathematics XVII*, pages 1–14, 1970.

[MZ15] Paul-André Melliès and Noam Zeilberger. Functors are type refinement systems. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 3–16. ACM, 2015.

[MZ25] Paul-André Melliès and Noam Zeilberger. The categorical contours of the Chomsky-Schützenberger representation theorem. *Logical Methods in Computer Science*, 21:12:1–12:49, 2025.

[Wal89] R. F. C. Walters. A note on context-free languages. *Journal of Pure and Applied Algebra*, 62(2):199–203, 1989.