Fibrational perspectives on logic, automata, and language theory

Noam Zeilberger[1]

Ecole Polytechnique (Palaiseau, France)

World Logic Day @ Tallinn, 10 January 2026

[1]Based on joint work with Bryce Clarke, Paul-André Melliès, and Gabriel Scherer.

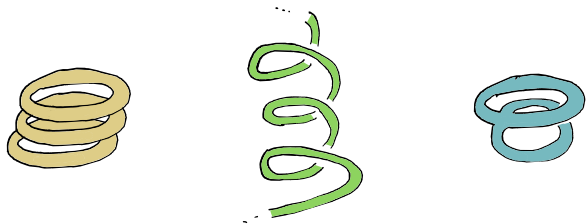
Intuitions

An idea from topology

When studying a space, it can be helpful to first project it down to “simpler” space.

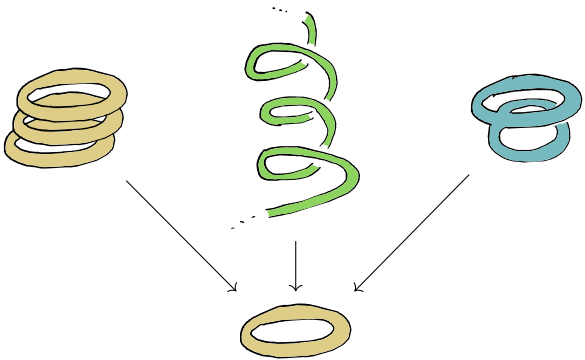
An idea from topology

When studying a space, it can be helpful to first project it down to “simpler” space.



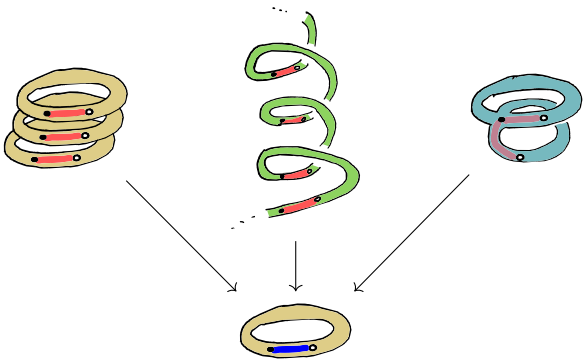
An idea from topology

When studying a space, it can be helpful to first project it down to "simpler" space.



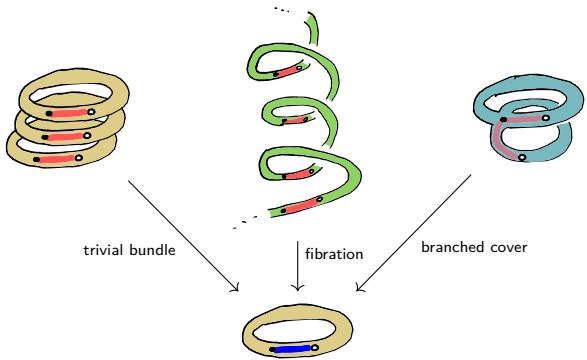
An idea from topology

When studying a space, it can be helpful to first project it down to “simpler” space.



An idea from topology

When studying a space, it can be helpful to first project it down to "simpler" space.



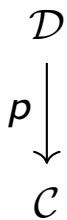
Categories and types

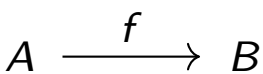
A *category* $\mathcal{C}$ is a collection of objects (points) and a collection of arrows (paths) closed under composition.

A *functor* $p : \mathcal{D} \to \mathcal{C}$ between two categories is a mapping from the objects and arrows of $\mathcal{D}$ to the objects and arrows of $\mathcal{C}$ that respects composition.

We like to think of functors through a type-theoretic lens...

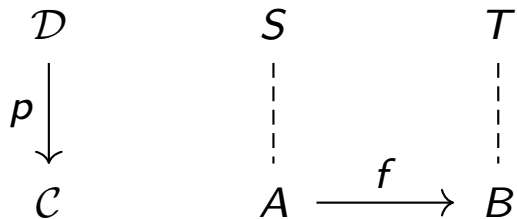
Functors as type refinement systems: some intuition





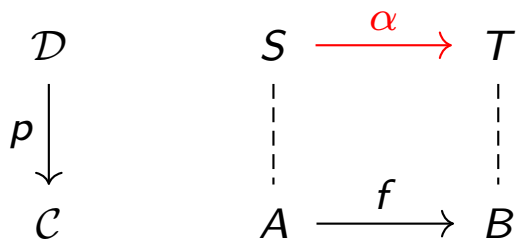
(f a program taking input of type A and producing output of type B)

Functors as type refinement systems: some intuition



(S and T predicates refining A and B)

Functors as type refinement systems: some intuition



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

This idea may be applied to study a wide variety of deductive systems.

We will consider three examples:

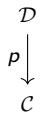
1. Bifibrations
2. Finite-state automata
3. Context-free grammars

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

1. The logic of bifibrations

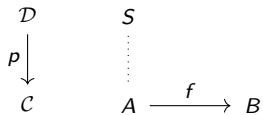
What is a bifibration?

Formally:



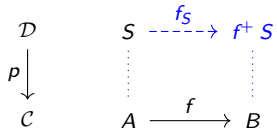
What is a bifibration?

Formally:



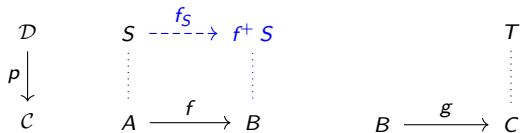
What is a bifibration?

Formally:



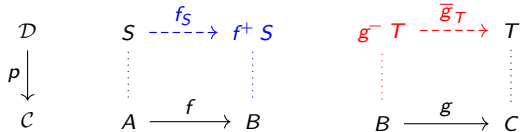
What is a bifibration?

Formally:



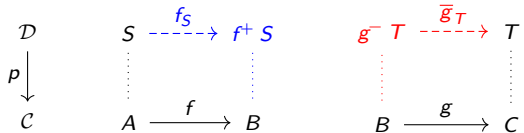
What is a bifibration?

Formally:



What is a bifibration?

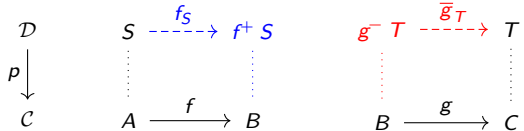
Formally:



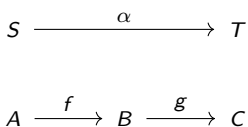
...and these liftings should be universal...

What is a bifibration?

Formally:

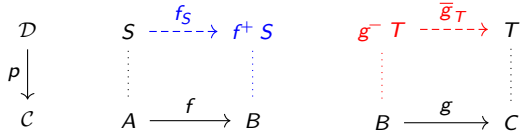


...and these liftings should be universal...



What is a bifibration?

Formally:

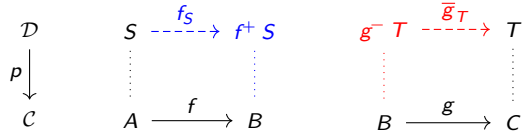


...and these liftings should be universal...

$$S \xrightarrow{f_S} f^+ S \xrightarrow{f \setminus_g \alpha} T \qquad S \xrightarrow{\quad\alpha\quad} T$$
$$=$$
$$A \xrightarrow{f} B \xrightarrow{g} C \qquad A \xrightarrow{f} B \xrightarrow{g} C$$

What is a bifibration?

Formally:

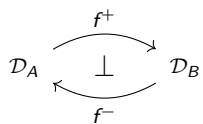


...and these liftings should be universal...

$$\begin{array}{ccccc} S & \xrightarrow{f_S} & f^+ S & \xrightarrow{f \setminus_g \alpha} & T \\ A & \xrightarrow{f} & B & \xrightarrow{g} & C \end{array} = \begin{array}{ccc} S & \xrightarrow{\alpha} & T \\ A & \xrightarrow{f} B \xrightarrow{g} & C \end{array} = \begin{array}{ccccc} S & \xrightarrow{\alpha_f / \overline{g}} & g^- T & \xrightarrow{\overline{g}_T} & T \\ A & \xrightarrow{f} & B & \xrightarrow{g} & C \end{array}$$

Bifibrations as families of adjunctions

A consequence of the definition is that in a bifibration $p : \mathcal{D} \to \mathcal{C}$, pushing and pulling along any given arrow $f : A \to B$ in $\mathcal{C}$ induces an *adjunction* between fiber categories:



$$\frac{S \Longrightarrow_{\mathrm{id}_A} f^- T}{f^+ S \Longrightarrow_{\mathrm{id}_B} T}$$

Indeed, any bifibration $\mathcal{D} \to \mathcal{C}$ can be transformed into a pseudofunctor $\mathcal{C} \to \mathrm{Adj}$, and vice versa. Thus bifibrations can be found almost everywhere there are adjunctions!

Example: Hoare logic

Let $\mathcal{C}$ be the one-object category whose arrows $c : * \longrightarrow *$ are sequential compositions of program commands defined over some global set of variables.

Example: Hoare logic

Let $\mathcal{C}$ be the one-object category whose arrows $c : * \longrightarrow *$ are sequential compositions of program commands defined over some global set of variables.

Let $\mathcal{D}$ be the category whose objects are predicates on the global state, and whose arrows $P \longrightarrow Q$ are given by commands c that satisfy the Hoare triple $\{P\}c\{Q\}$.

Example: Hoare logic

Let $\mathcal{C}$ be the one-object category whose arrows $c : * \to *$ are sequential compositions of program commands defined over some global set of variables.

Let $\mathcal{D}$ be the category whose objects are predicates on the global state, and whose arrows $P \to Q$ are given by commands c that satisfy the Hoare triple $\{P\}c\{Q\}$.

Let $p : \mathcal{D} \to \mathcal{C}$ be the evident forgetful functor.

Example: Hoare logic

Let $\mathcal{C}$ be the one-object category whose arrows $c : * \to *$ are sequential compositions of program commands defined over some global set of variables.

Let $\mathcal{D}$ be the category whose objects are predicates on the global state, and whose arrows $P \to Q$ are given by commands c that satisfy the Hoare triple $\{P\}c\{Q\}$.

Let $p : \mathcal{D} \to \mathcal{C}$ be the evident forgetful functor.

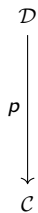
The following are equivalent:

1. $p : \mathcal{D} \to \mathcal{C}$ is a bifibration
2. *weakest preconditions* and *strongest postconditions* exist for all P, c, Q

$$P \Rightarrow wp(c, Q) \quad \iff \quad \{P\}c\{Q\} \quad \iff \quad sp(c, P) \Rightarrow Q$$

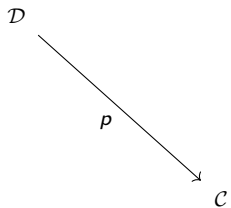
Free bifibrations

Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



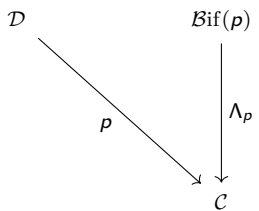
Free bifibrations

Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



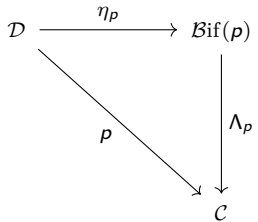
Free bifibrations

Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



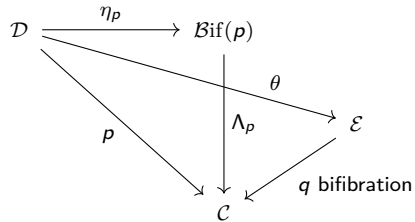
Free bifibrations

Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



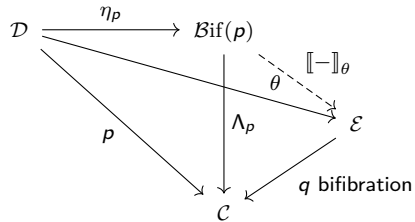
Free bifibrations

Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



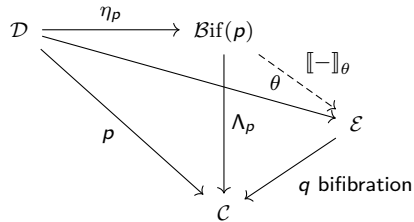
Free bifibrations

Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



Free bifibrations

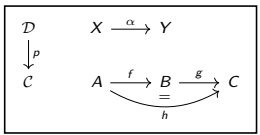
Not every functor is a bifibration, but one can *freely* construct a bifibration over any functor in the following sense:



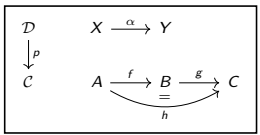
In jww Bryce Clarke and Gabriel Scherer, we gave an explicit construction of $\mathcal{B}\mathrm{if}(p)$

- "The free bifibration on a functor", arxiv:2511.07314

An example derivation in a free bifibration

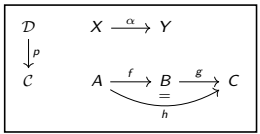


An example derivation in a free bifibration

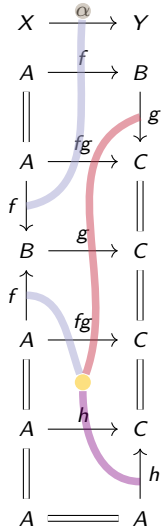


$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \underset{f}{\Longrightarrow} Y}\,\alpha}{X \underset{fg}{\Longrightarrow} g^{+}\, Y}\,\mathrm{R}g^{+}}{f^{+}\, X \underset{g}{\Longrightarrow} g^{+}\, Y}\,\mathrm{L}f^{+}}{f^{-}\, f^{+}\, X \underset{fg}{\Longrightarrow} g^{+}\, Y}\,\mathrm{L}f^{-}}{f^{-}\, f^{+}\, X \underset{h}{\Longrightarrow} g^{+}\, Y}}{f^{-}\, f^{+}\, X \underset{\mathrm{id}_A}{\Longrightarrow} h^{-}\, g^{+}\, Y}\,\mathrm{R}h^{-}$$

An example derivation in a free bifibration

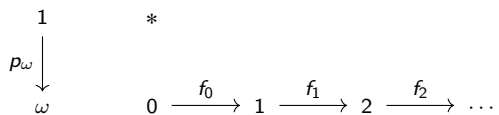


$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \underset{f}{\Longrightarrow} Y}\,\alpha}{X \underset{fg}{\Longrightarrow} g^{+}\,Y}\,\mathrm{R}g^{+}}{f^{+}\,X \underset{g}{\Longrightarrow} g^{+}\,Y}\,\mathrm{L}f^{+}}{f^{-}\,f^{+}\,X \underset{fg}{\Longrightarrow} g^{+}\,Y}\,\mathrm{L}f^{-}}{f^{-}\,f^{+}\,X \underset{h}{\Longrightarrow} g^{+}\,Y}}{f^{-}\,f^{+}\,X \underset{\mathrm{id}_A}{\Longrightarrow} h^{-}\,g^{+}\,Y}\,\mathrm{R}h^{-}$$

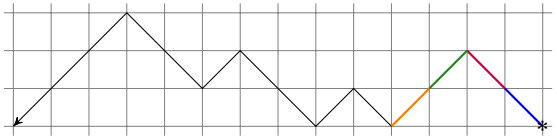


Quiz time

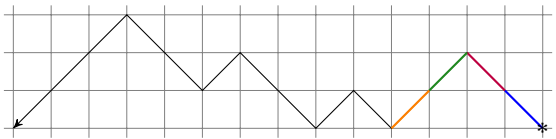
Let ω be the totally ordered set of natural numbers considered as a category.



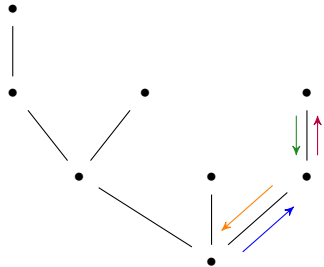
Question: what is the fiber over 0 of the free bifibration on p_ω?

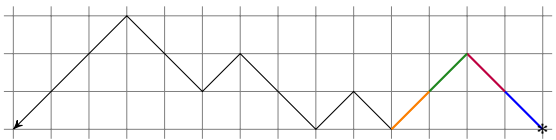


$f^- \; f^- \; f^- \; f^+ \; f^+ \; f^- \; f^+ \; f^+ \; f^- \; f^+ \; f^- \; f^- \; f^+ \; f^+ \; *$

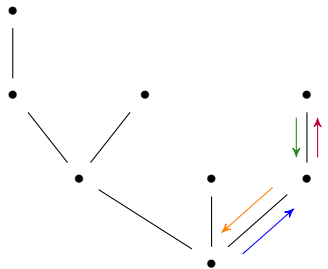


$f^- f^- f^- f^+ f^+ f^- f^+ f^+ f^- f^+ f^- f^- f^+ f^+ *$

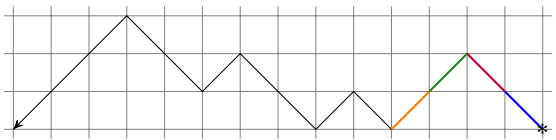




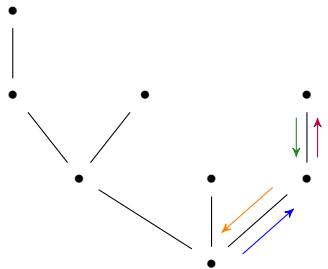
$f^- \; f^- \; f^- \; f^+ \; f^+ \; f^- \; f^+ \; f^+ \; f^- \; f^+ \; f^- \; f^- \; f^+ \; f^+ \; *$



Thus objects of $\mathcal{B}\mathrm{if}(p_\omega)_0$ are isomorphic to Dyck walks $\cong$ plane trees!

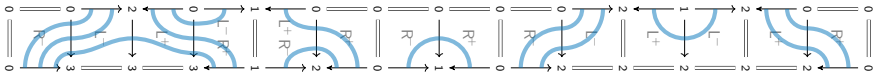


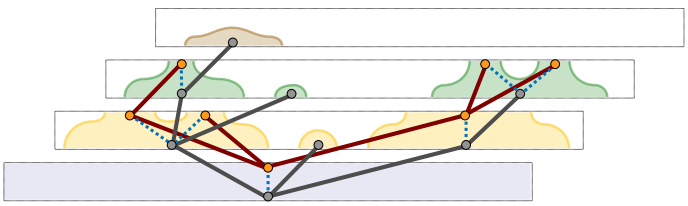
$f^- \; f^- \; f^- \; f^+ \; f^+ \; f^- \; f^+ \; f^+ \; f^- \; f^+ \; f^- \; f^- \; f^+ \; f^+ \; *$

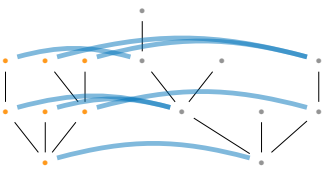


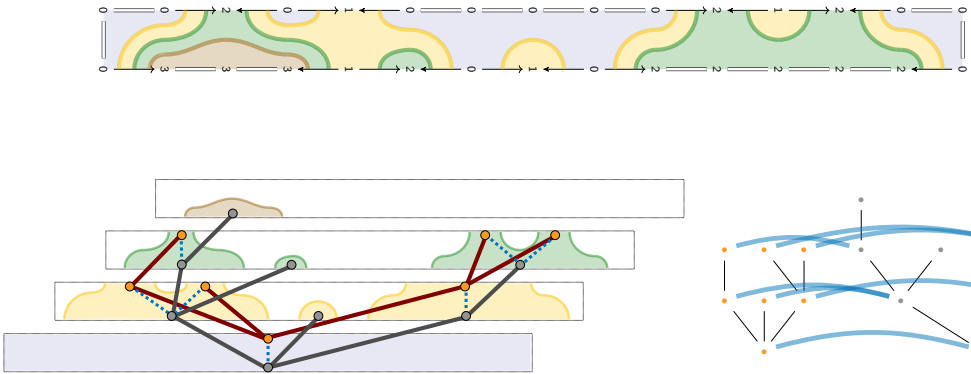
Thus objects of $\mathcal{B}\mathrm{if}(p_\omega)_0$ are isomorphic to Dyck walks $\cong$ plane trees!

But what are its arrows?









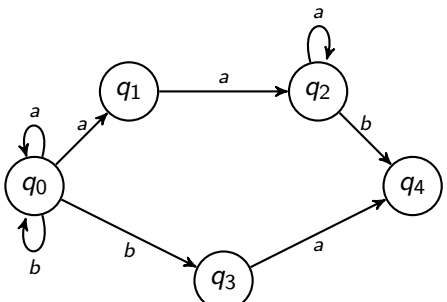
Theorem. $\mathcal{B}\mathrm{if}(p_\omega)_0$ is equivalent to the full subcategory of $[\omega^{\mathrm{op}}, \Delta]$ spanned by the functors such that $T(0) = 1$ and $\exists k.\ T(k) = 0$ (i.e., by the finite trees).

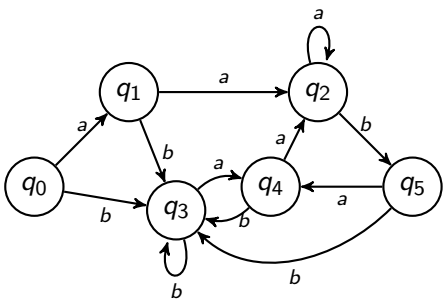
2. Finite-state automata over categories

Automata as graph homomorphisms

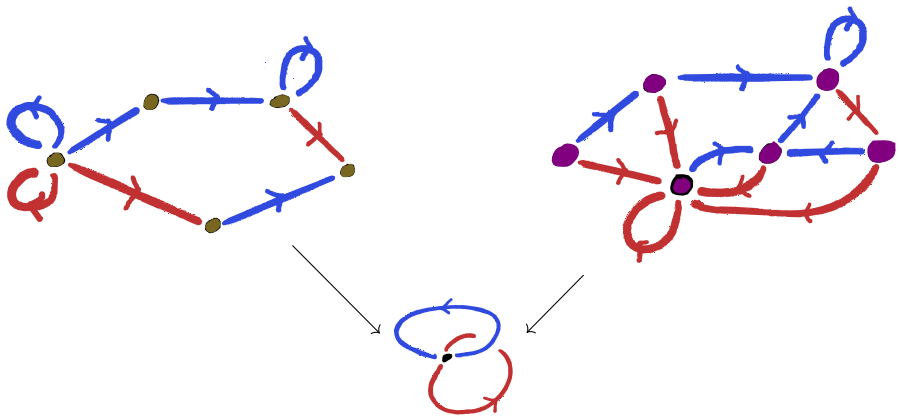
The underlying transition graph of any NFA (without ϵ-transitions) over the alphabet Σ is entirely described by a graph homomorphism $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ into the *bouquet graph* with one node $*$ and a loop $a : * \to *$ for every $a \in \Sigma$.

Automata as graph homomorphisms

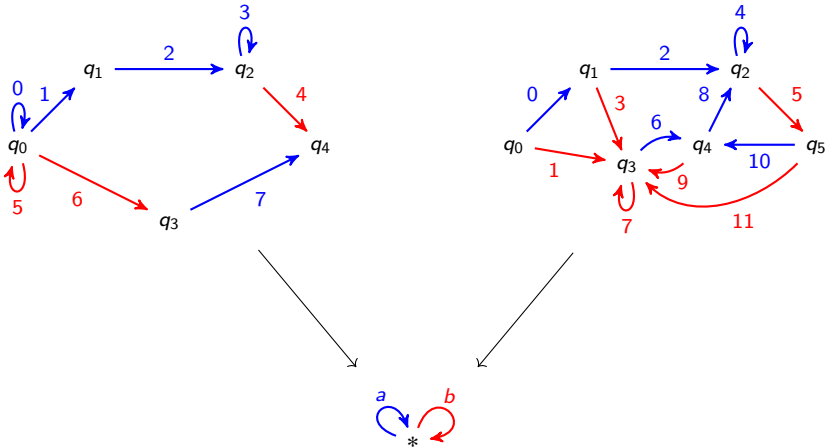




Automata as graph homomorphisms



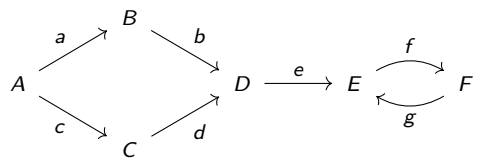
Automata as graph homomorphisms



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

$\mathbb{G}$ =



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

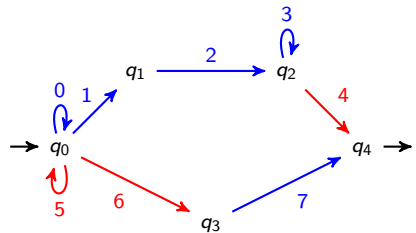
Any *word* $w \in \Sigma^*$ corresponds to a *path* in $\mathbb{B}_\Sigma$, i.e., to an arrow $* \to *$ in $\mathcal{F}\mathbb{B}_\Sigma$.

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.

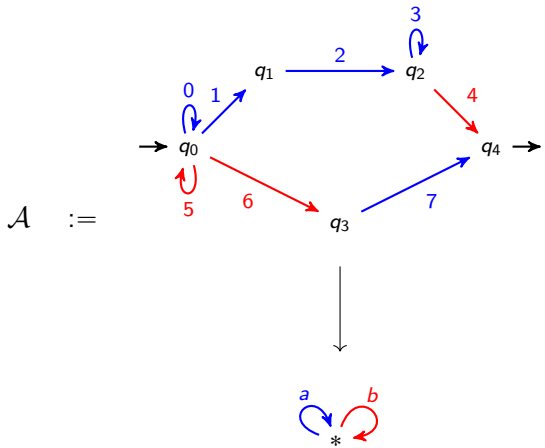
Recognition as a path lifting problem

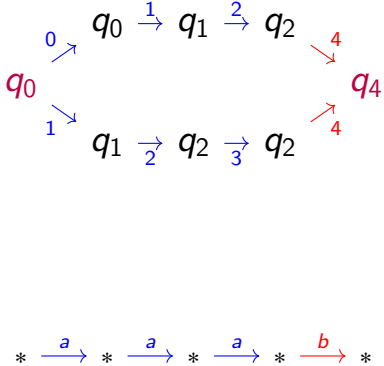
$\mathcal{A} :=$



q_0 q_4

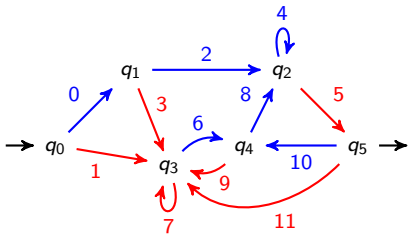
Recognition as a path lifting problem



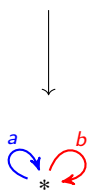


Recognition as a path lifting problem

$\mathcal{A}$:=

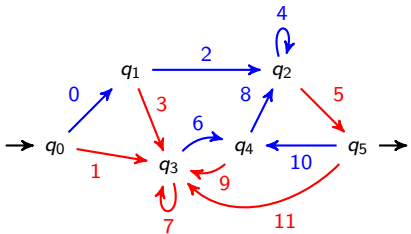


q_0



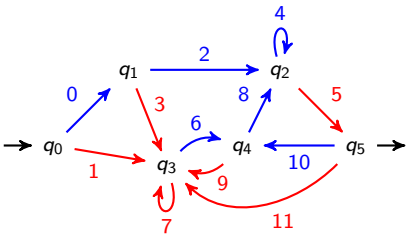
Recognition as a path lifting problem

$\mathcal{A}$:=



$q_0 \xrightarrow{0} q_1$

Recognition as a path lifting problem

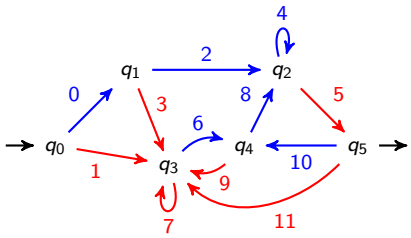


$q_0 \xrightarrow{0} q_1 \xrightarrow{2} q_2$

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

Recognition as a path lifting problem

$\mathcal{A}$:=

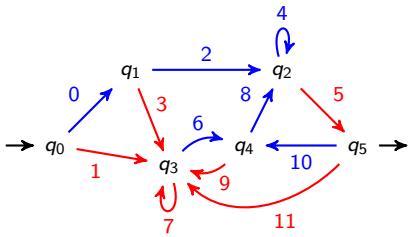


$q_0 \xrightarrow{0} q_1 \xrightarrow{2} q_2 \xrightarrow{4} q_2$

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

Recognition as a path lifting problem

$\mathcal{A}$:=



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

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

A key property of automata

Deterministic automata are represented by *discrete opfibrations* over $\mathcal{F}\mathbb{B}_{\Sigma}$.

A key property of automata

Deterministic automata are represented by *discrete opfibrations* over $\mathcal{F}\mathbb{B}_\Sigma$.

But more generally, when does a functor $p : \mathcal{D} \to \mathcal{F}\mathbb{B}_\Sigma$ represent an NFA?

A key property of automata

Deterministic automata are represented by *discrete opfibrations* over $\mathcal{F}\mathbb{B}_\Sigma$.

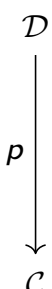
But more generally, when does a functor $p : \mathcal{D} \to \mathcal{F}\mathbb{B}_\Sigma$ represent an NFA?

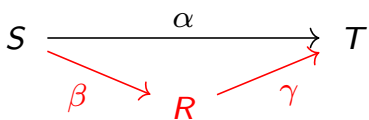
Proposition. The following are equivalent:

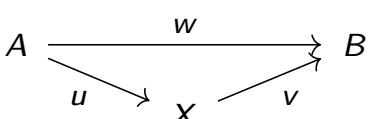
1. $\mathcal{D} = \mathcal{F}\mathbb{G}$ and $p = \mathcal{F}\phi$ where $\phi : \mathbb{G} \to \mathbb{B}_\Sigma$ is the transition graph of a NFA.
2. p has finite fibers and the **unique lifting of factorizations** property.

Unique lifting of factorizations

Let α be an arrow in $\mathcal{D}$ with image $p(\alpha) = w$ in $\mathcal{C}$. If w factors as uv, then there exist unique β and γ in $\mathcal{D}$ such that $\alpha = \beta\gamma$ and $p(\beta) = u$ and $p(\gamma) = v$.



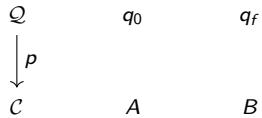




Definition: NFA over a category $\mathcal{C}$

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

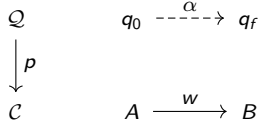
- a category $\mathcal{Q}$
- a finitary ULF functor $p : \mathcal{Q} \to \mathcal{C}$
- a pair of objects $q_0, q_f \in \mathcal{Q}$



Definition: NFA over a category $\mathcal{C}$

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

- a category $\mathcal{Q}$
- a finitary ULF functor $p : \mathcal{Q} \to \mathcal{C}$
- a pair of objects $q_0, q_f \in \mathcal{Q}$



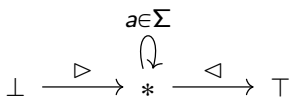
Such an automaton recognizes a **regular language of arrows** $\mathcal{L}_{\mathcal{A}}$ defined as the image of the homset $\mathcal{Q}(q_0, q_f)$ along p, that is,

$$\mathcal{L}_{\mathcal{A}} \stackrel{\text{def}}{=} \{\, p(\alpha) \mid \alpha : q_0 \to q_f \,\} \subseteq \mathcal{C}(A, B)$$

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

Example: automata over free categories

Let $\mathbb{B}_{\Sigma}^{\triangleright\triangleleft}$ be the graph obtained from $\mathbb{B}_{\Sigma}$ by adjoining a pair of initial and final nodes:

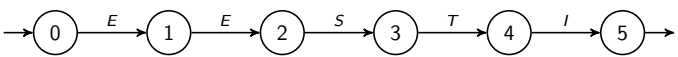


An automaton over $\mathcal{F}\mathbb{B}_{\Sigma}^{\triangleright\triangleleft}$ processes a word with explicit begin/end markers.

More generally, an automaton over the free category $\mathcal{F}\mathbb{H}$ on an arbitrary graph $\mathbb{H}$ recognizes a language of paths, which may be considered as "typed words".

Example: singleton automaton

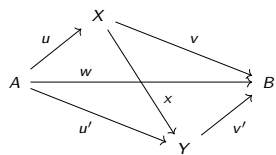
For any $w \in \Sigma^*$ s.t. $|w| = n$, there is an $(n+1)$-state automaton recognizing $\{ w \}$:



This is a special case of the following general construction...

Example: singleton automaton

Let $w : A \to B$ be an arrow of a category $\mathcal{C}$. The category Fact_w has objects given by triples $(X,\ A \xrightarrow{u} X \xrightarrow{v} B\ )$ such that $w = uv$, and arrows $(X, u, v) \to (Y, u', v')$ given by arrows $x : X \to Y$ making the diagram commute:



Let $p_w : \mathrm{Fact}_w \to \mathcal{C}$ be the middle projection $(X, u, v) \mapsto X$. Note p_w is ULF.

Then $(\mathcal{Q}, p, q_0, q_f) := (\mathrm{Fact}_w, p_w, (\mathrm{id}_A, w), (w, \mathrm{id}_B))$ defines an automaton recognizing the singleton language $\{\, w \,\} \subset \mathcal{C}(A, B)$. (This automaton is not necessarily finite state; we say $\mathcal{C}$ has *finitary factorizations* if p_w has finite fibers for all $w : A \to B \in \mathcal{C}$.)

3. Context-free grammars over categories and operads

From categories to operads

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

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

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

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

where $A = p(S)$.

Parsing as a lifting problem

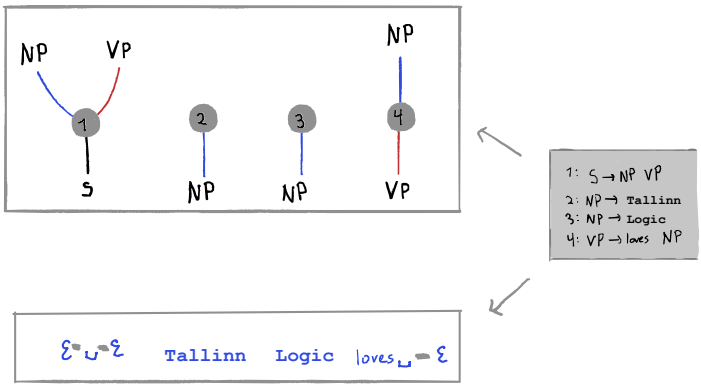
$S \to NP\ VP$

$NP \to \texttt{Tallinn}$

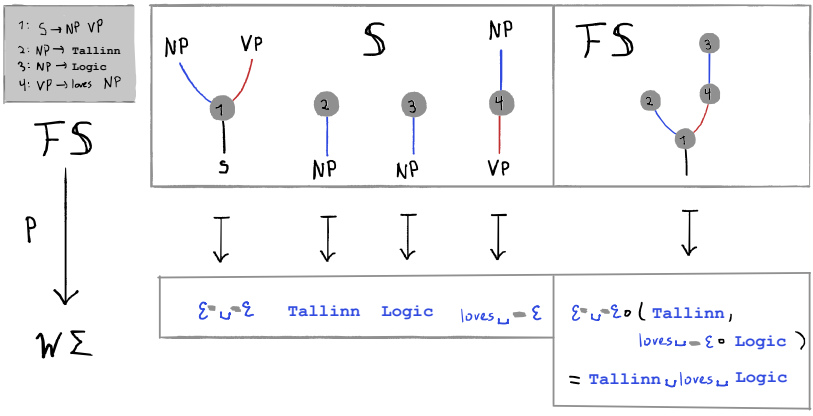
$NP \to \texttt{Logic}$

$VP \to \text{loves}\ NP$

Parsing as a lifting problem



Parsing as a lifting problem

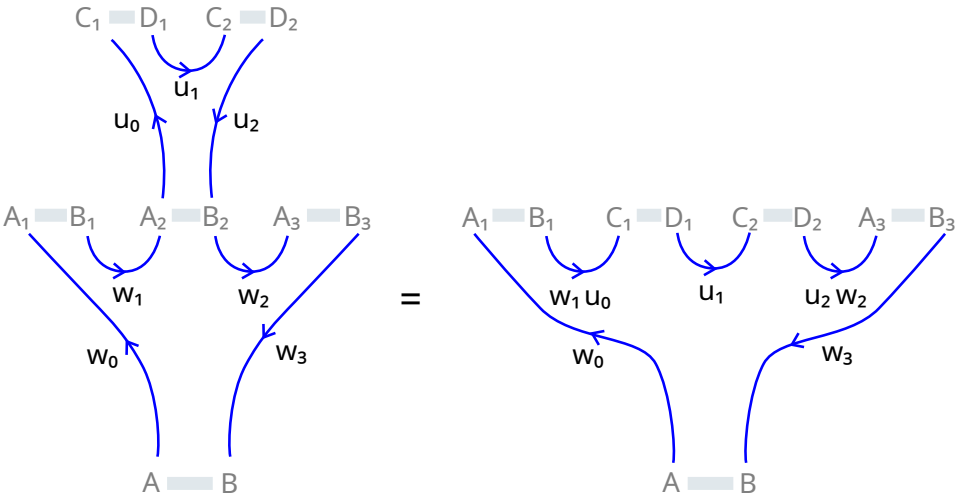


The splicing construction

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



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

Definition: CFG over a category $\mathcal{C}$

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

- a finite signature $\mathbb{S}$ of objects and operations
- a distinguished object $S \in \mathbb{S}$
- a functor $p : \mathcal{F}\mathbb{S} \to \mathcal{W}\mathcal{C}$

The grammar generates a **context-free language of arrows** given by

$$\mathcal{L}_G \stackrel{\text{def}}{=} \mathcal{L}_{p,S} = \{\, p(\alpha) \mid \alpha : S \,\} \subseteq \mathcal{C}(A, B)$$

where $p(S) = (A, B)$.

Intersecting a grammar with an automaton

Theorem. For any CFG $G = (\mathbb{S}, S, p_G)$ and NFA $M = (\mathcal{Q}, p_M, q_0, q_f)$ over the same category, with $p_G(S) = (p_M(q_0), p_M(q_f))$, there is a CFG generating $\mathcal{L}_G \cap \mathcal{L}_M$.

Intersecting a grammar with an automaton

Theorem. For any CFG $G = (\mathbb{S}, S, p_G)$ and NFA $M = (\mathcal{Q}, p_M, q_0, q_f)$ over the same category, with $p_G(S) = (p_M(q_0), p_M(q_f))$, there is a CFG generating $\mathcal{L}_G \cap \mathcal{L}_M$.

Proof sketch:

1. a finitary ULF functor of categories $p_M : \mathcal{Q} \to \mathcal{C}$ induces a finitary ULF functor of operads $\mathcal{W}\, p_M : \mathcal{W}\,\mathcal{Q} \to \mathcal{W}\,\mathcal{C}$.
2. the pullback on the right is induced by a pullback on the left.

$$\begin{array}{ccc} \mathbb{S}' & \xrightarrow{\psi'} & \mathbb{S} \\ {\scriptstyle \phi'_G}\downarrow & \lrcorner & \downarrow{\scriptstyle \phi_G} \\ \mathcal{W}\,\mathcal{Q} & \xrightarrow[\mathcal{W}\,p_M]{} & \mathcal{W}\,\mathcal{C} \end{array} \in \text{Sig} \qquad \begin{array}{ccc} \mathcal{F}\mathbb{S}' & \xrightarrow{\mathcal{F}\psi'} & \mathcal{F}\mathbb{S} \\ {\scriptstyle p'_G}\downarrow & \lrcorner & \downarrow{\scriptstyle p_G} \\ \mathcal{W}\,\mathcal{Q} & \xrightarrow[\mathcal{W}\,p_M]{} & \mathcal{W}\,\mathcal{C} \end{array} \in \text{Oper}$$

Going further

The operad $\mathcal{WC}$ can be replaced by an arbitrary base operad $\mathcal{O}$ to define a notion of generalized CFG over an operad. Similarly, one can define generalized NFA over arbitrary operads by considering finitary ULF functors of operads.

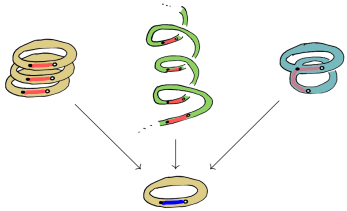
Going further

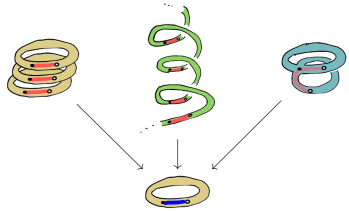
The operad $\mathcal{W}\mathcal{C}$ can be replaced by an arbitrary base operad $\mathcal{O}$ to define a notion of generalized CFG over an operad. Similarly, one can define generalized NFA over arbitrary operads by considering finitary ULF functors of operads.

Although the functors representing grammars and automata typically are *not* bifibrations, it can be interesting to *model* them in bifibrations. In particular, the language generated by a CFG extends to a morphism in $\mathrm{Oper}^{\to}$ from $p_G : \mathcal{F}\mathbb{S} \to \mathcal{W}\Sigma$ to the *bifibration of subsets of sets*.

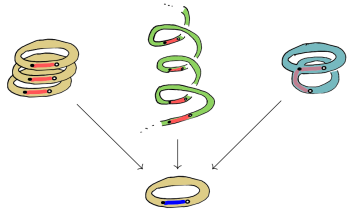
$$\begin{array}{ccc} \mathcal{F}\mathbb{S} & \xrightarrow{[\![-]\!]} & \mathrm{SubSet} \\ \downarrow{\scriptstyle p_G} & & \downarrow{\scriptstyle \pi} \\ \mathcal{W}\Sigma & \xrightarrow[[\![-]\!]]{} & \mathrm{Set} \end{array}$$

Concluding perspectives



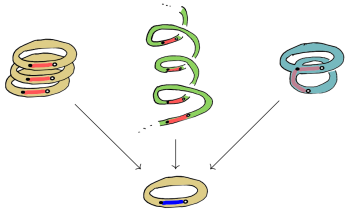


Type systems, automata, and grammars may be considered as bundles of categories or operads, in a way that allows to treat them uniformly.



Type systems, automata, and grammars may be considered as bundles of categories or operads, in a way that allows to treat them uniformly.

Fibrational properties as well as free constructions play an important role, and have deep connections to logic.



Type systems, automata, and grammars may be considered as bundles of categories or operads, in a way that allows to treat them uniformly.

Fibrational properties as well as free constructions play an important role, and have deep connections to logic.

References:

- w/B. Clarke & G. Scherer: "The free bifibration on a functor", arxiv:2511.07314
- w/P.-A. Melliès: "Functors are type refinement systems" (POPL 2015); "The categorical contours of the Ch.-Sch. repr. theorem" (LMCS 21:2, 2025)