Effect-Driven Parsing

Formal studies on a categorical approach to semantic parsing

Matthieu Boyer

Yale ENS | PSL★

September 9 2025

École Normale Supérieure | Yale University

Context

- Categorical formalization of a type and effects system for natural-language semantics (following [BC25]).

- Develop a graphical, type-driven parsing formalism that derives sentence meaning compositionally from word meanings.

Typed Semantics for Natural Languages

Expression	Type	λ-Term
planet	$\mathtt{e} \to \mathtt{t}$	$\lambda x.\mathbf{planet}\,x$
	Generalizes to **common nouns**	
Jupiter	$\mathtt{e}$	$\mathbf{j} \in \mathrm{Var}$
	Generalizes to **proper nouns**	
chase	$\mathtt{e} \to \mathtt{e} \to \mathtt{t}$	$\lambda o.\lambda s.\mathbf{chase}\,(o)\,(s)$
	Generalizes to **transitive verbs**	

Syntactic Types and Semantic Types

- Syntax **a cat** and **the cat** should have the same type: e.

Syntactic Types and Semantic Types

- Syntax **a cat** and **the cat** should have the same type: e.

- No single canonical **cat** exists: that type cannot be e.

Syntactic Types and Semantic Types

- Syntax **a cat** and **the cat** should have the same type: e.

- No single canonical **cat** exists: that type cannot be e.

We will use **(side-)effects** to do the difference between them:

$$\mathbf{a\ cat} = \{c \mid \mathbf{cat}\,c\} \qquad \text{(Set)}$$

$$\mathbf{the\ cat} = x \text{ if } \mathbf{cat}^{-1}(\top) = \{x\} \text{ else } \# \qquad \text{(Maybe)}$$

Effects as Functors

Traditionally ([Mog89]), monads model side-effects.

Effects as Functors

Traditionally ([Mog89]), monads model side-effects.

Here, functors suffice and lighter structures are useful.

String Diagrams

String Diagrams are a formalism ([HM23]) for visualising multi-threading computations.

Theoretically, the dual of a diagram in a 2-category.

String Diagrams

String Diagrams are a formalism ([HM23]) for visualising multi-threading computations.

Theoretically, the dual of a diagram in a 2-category.

Used as a tool for parsing in [CSC10].

École Normale Supérieure | Yale University

Other Categorical Theories

[Mar] and [SM25] use Hopf algebras to give a model for parsing.

[MZ25] use operads to prove results on CFGs.

Notations

- Language $\mathcal{L}$ of denotationally composed words.
- Base typing CCC $\mathcal{C}$; Effects: functors $\mathcal{F}(\mathcal{L})$.

Notations

- Language $\mathcal{L}$ of denotationally composed words.

- Base typing CCC $\mathcal{C}$; Effects: functors $\mathcal{F}(\mathcal{L})$.

- Typing CCC: $\bar{\mathcal{C}} = $ closure of $\mathcal{C}$ under $\mathcal{F}(\mathcal{L})^*$, products and exponentials.

Intuition: all (effect-sequence, base type) combos, with functions/products.

Intuitionistic-style Typing Judgements

We then have typing judgements for basic combinations:

$$\frac{\Gamma \vdash x : \tau \qquad \Gamma \vdash F \in \mathcal{F}(\mathcal{L})}{\Gamma \vdash Fx : F\tau}\mathsf{Cons}$$

$$\frac{\Gamma \vdash x : F\tau_1 \qquad \Gamma \vdash \varphi : \tau_1 \rightarrow \tau_2}{\Gamma \vdash \varphi x : F\tau_2}\mathtt{fmap}$$

Intuitionistic-style Typing Judgements

We then have typing judgements for basic combinations:

$$\frac{\Gamma \vdash x : \tau_1 \qquad \Gamma \vdash \varphi : \tau_1 \to \tau_2}{\Gamma \vdash \varphi x : \tau_2} \mathsf{App}$$

$$\frac{\Gamma \vdash x : A\tau_1 \qquad \Gamma \vdash \varphi : A\left(\tau_1 \to \tau_2\right)}{\Gamma \vdash \varphi x : A\tau_2} \texttt{<*>}$$

Intuitionistic-style Typing Judgements

Typing judgements for natural transformations:

$$\frac{\Gamma \vdash x : \tau}{\Gamma \vdash x : A\tau}\texttt{pure/return}$$

$$\frac{\Gamma \vdash x : MM\tau}{\Gamma \vdash x : M\tau}\texttt{>>=}$$

$$\forall F \overset{\theta}{\Longrightarrow} G, \qquad \frac{\Gamma \vdash x : F\tau \qquad \Gamma \vdash G : S' \subseteq \star \qquad \tau \in S'}{\Gamma \vdash x : G\tau}\texttt{nat}$$

Effect-Driven Parsing
Category-theoretical type system
Introducing a language
Yale | ENS | PSL ★

Presentation

To present a language in our formalism, we need:

- A syntax;

- A typed dictionary using effects;

- A typed lexicon of non-verbal constructs.

Effect-Driven Parsing
Category-theoretical type system
Introducing a language

Introducing Higher-Order Constructs

We implement higher-order semantics (e.g. the future and plural) via functors.

Effect-Driven Parsing
└ Category-theoretical type system
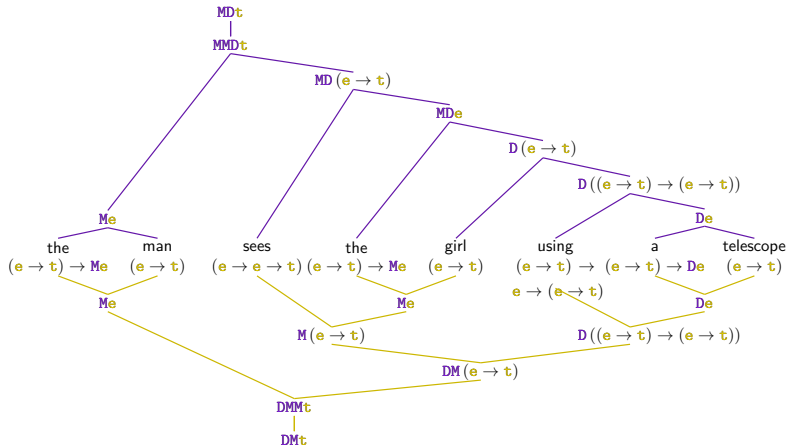 └ Introducing a language

Yale · ENS | PSL★

Introducing Higher-Order Constructs

We implement higher-order semantics (e.g. the future and plural) via functors.

We also enforce the notion of scope islands as in [BC25]:

$$\mathbf{if} \; : (\mathtt{t} \setminus \mathcal{F}(\mathcal{L})^* \, \mathtt{C}\mathtt{t}) \to \mathtt{t} \to \mathtt{t}$$

Effect-Driven Parsing
Category-theoretical type system
Introducing a language

Yale | ENS | PSL★

Ambiguity

Handlers

Handlers for an effect F are natural transformations $F \Rightarrow \mathrm{Id}$ ([WSH14]) which invert units.

Handlers

Handlers for an effect F are natural transformations $F \Rightarrow \mathrm{Id}$ ([WSH14]) which invert units.

1. Language-Defined Handlers arise from fundamental properties of the considered effects.

Handlers

Handlers for an effect F are natural transformations $F \Rightarrow \mathrm{Id}$ ([WSH14]) which invert units.

1. Language-Defined Handlers arise from fundamental properties of the considered effects.

2. Speaker-dependant handlers which are dependent on the speaker.

String Diagrams Representation of Sentences

String diagram are a representation of the side-effects and types of a sentence across its computation.

Yale ENS | PSL★
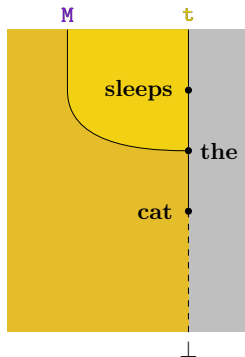
Deformation of String Diagrams

Theorem 3.1 — **Theorem 3.1 [Sel10], Theorem 1.2 [JS91]** A well-formed equation between morphism terms in the language of monoidal categories follows from the axioms of monoidal categories if and only if it holds, up to planar isotopy, in the graphical language.

Equations on String Diagrams

Properties of monads, natural transformations, adjunctions and more can be explained in terms of commutative diagrams, but also as string diagram equations.

Moreover, Theorem 3.1 can be implemented as string diagram equations.

Confluence of Reductions

Theorem 3.2 — Confluence The reduction system defined by the specified equations is confluent and therefore defines normal forms.

Theorem 3.3 — Normalization Complexity Normalization is quadratic in the number of natural transformations.

This is accomplished using a formalism based on [DV22].
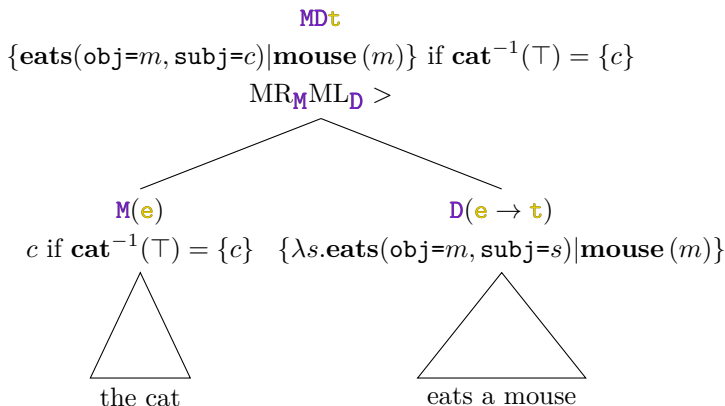
Parsing Algorithm

Typing syntax trees is exponentially too long.
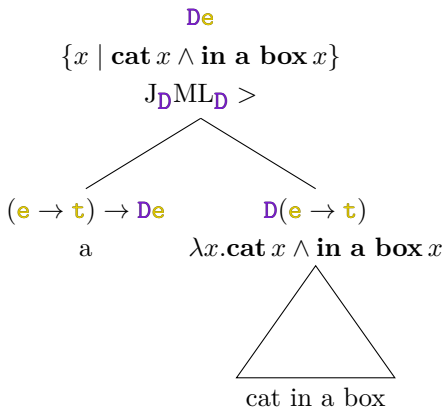
Parsing Algorithm

Typing syntax trees is exponentially too long.

We use a Context-Free Grammar in five parts to model our typing system. This gives a complexity in $\mathcal{O}(|\mathcal{F}(\mathcal{L})|\,|\mathcal{S}|\,n^3)$.
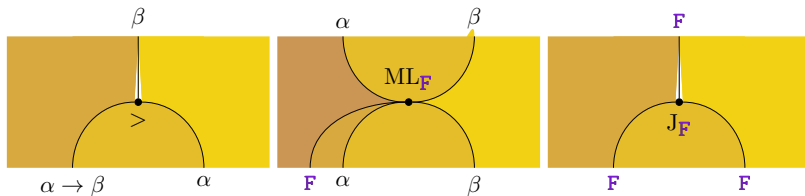
Semantic Parse Trees I

$$\text{MD}\mathtt{t}$$

$$\{\mathbf{eats}(\mathtt{obj}{=}m, \mathtt{subj}{=}c) | \mathbf{mouse}\,(m)\} \text{ if } \mathbf{cat}^{-1}(\top) = \{c\}$$

$$\text{MR}_{\mathbf{M}}\text{ML}_{\mathbf{D}} >$$

$$\text{M}(\mathtt{e})$$

$$c \text{ if } \mathbf{cat}^{-1}(\top) = \{c\}$$

$$\text{D}(\mathtt{e} \to \mathtt{t})$$

$$\{\lambda s.\mathbf{eats}(\mathtt{obj}{=}m, \mathtt{subj}{=}s) | \mathbf{mouse}\,(m)\}$$

the cat

eats a mouse

Semantic Parse Trees II

$$\texttt{D}\,\texttt{e}$$
$$\{x \mid \mathbf{cat}\,x \wedge \mathbf{in\ a\ box}\,x\}$$
$$\mathrm{J}_{\texttt{D}}\mathrm{ML}_{\texttt{D}} >$$

$$(\texttt{e} \to \texttt{t}) \to \texttt{D}\,\texttt{e} \qquad \texttt{D}(\texttt{e} \to \texttt{t})$$
$$\mathrm{a} \qquad \lambda x.\mathbf{cat}\,x \wedge \mathbf{in\ a\ box}\,x$$

cat in a box

Semantic Parse Trees III
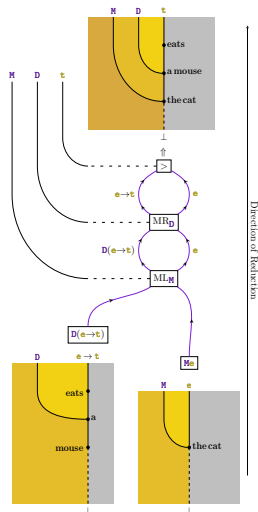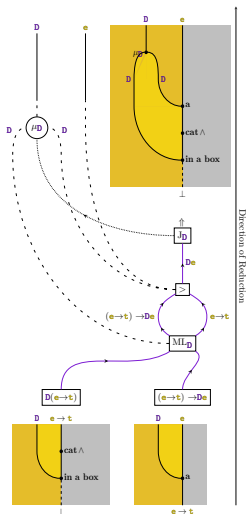
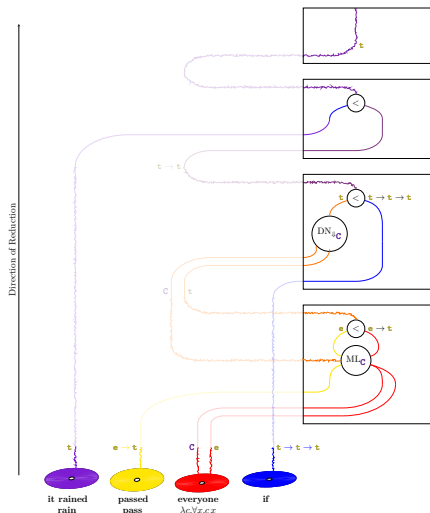# Combinators as String Diagrams

A Parsing Diagram Step

Another Parsing Diagram Step

Building an Intuition

A Full Parsing Diagram

Reducing the grammar

We translate equalities on actual denotations (from combinators or from the denotational system) into the reduction system on string diagrams.

Commutation of effects, Theorem 3.1 and more, allow a reduction of the constant in the algorithmic complexity.

Conclusion

- Theoretical enhancement of a type system for natural language semantics;

Conclusion

- Theoretical enhancement of a type system for natural language semantics;
- No load added neither on the user (comprehension) nor the parser (efficiency).

Conclusion

- Theoretical enhancement of a type system for natural language semantics;
- No load added neither on the user (comprehension) nor the parser (efficiency).

I would like to thank Simon Charlow for his advice and guidance, my mother for the knitting, Antoine Groudiev for the rotation of the snakes in equation labels, Bella Senturia, Bob Frank and Paul-André Melliès for their suggestions of papers to read about categories and linguistics.

Thank you for your attention.

Do you have any questions?

Lexicon

Expression	Type	λ-Term
planet	$\mathtt{e} \to \mathtt{t}$	$\lambda x.\mathbf{planet}\,x$
	Generalizes to **common nouns**	
carnivorous	$(\mathtt{e} \to \mathtt{t})$	$\lambda x.\mathbf{carnivorous}\,x$
	Generalizes to **predicative adjectives**	
skillful	$(\mathtt{e} \to \mathtt{t}) \to (\mathtt{e} \to \mathtt{t})$	$\lambda p.\lambda x.px \wedge \mathbf{skillful}\,x$
	Generalizes to **predicate modifier adjectives**	
Jupiter	$\mathtt{e}$	$\mathbf{j} \in \mathrm{Var}$
	Generalizes to **proper nouns**	
sleep	$\mathtt{e} \to \mathtt{t}$	$\lambda x.\mathbf{sleep}\,x$
	Generalizes to **intranitive verbs**	
chase	$\mathtt{e} \to \mathtt{e} \to \mathtt{t}$	$\lambda o.\lambda s.\mathbf{chase}\,(o)\,(s)$
	Generalizes to **transitive verbs**	
be	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{e} \to \mathtt{t}$	$\lambda p.\lambda x.px$
she	$r \to \mathtt{e}$	$\lambda g.g_0$
it	$\mathtt{G}\mathtt{e}$	$\lambda g.g_0$
which	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{S}\mathtt{e}$	$\lambda p. \{x \mid px\}$
the	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{M}\mathtt{e}$	$\lambda p.x$ if $p^{-1}(\top) = \{x\}$ else $\#$
a	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{D}\mathtt{e}$	$\lambda p.\lambda s. \{\langle x, x + s \rangle \mid px\}$
no	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{C}\mathtt{e}$	$\lambda p.\lambda c.\neg \exists x.px \wedge c\,x$
every	$(\mathtt{e} \to \mathtt{t}) \to \mathtt{C}\mathtt{e}$	$\lambda p.\lambda c.\forall x, px \Rightarrow c\,x$
$\cdot, \mathbf{a}\,\cdot$	$\mathtt{e} \to (\mathtt{e} \to \mathtt{t}) \to \mathtt{W}\mathtt{e}$	$\lambda x.\lambda p.\, \langle x, px \rangle$
as for	$\mathtt{e} \to \mathtt{T}\mathtt{e}$	$\lambda x.\lambda s.\, \langle x, x + s \rangle$
$\cdot_{\mathbf{F}}$	$\mathtt{e} \to \mathtt{F}\mathtt{e}$	$\lambda v.\, \langle v, \{x \mid x \in D_e\} \rangle$

Functor Denotations

Constructor	fmap	Typeclass
$\mathbf{G}(\tau) = \mathbf{r} \to \tau$	$\mathbf{G}\varphi(x) = \lambda r.\varphi(xr)$	Monad
$\mathbf{W}(\tau) = \tau \times \mathbf{t}$	$\mathbf{W}\varphi(\langle a, p \rangle) = \langle \varphi a, p \rangle$	Monad
$\mathbf{S}(\tau) = \{\tau\}$	$\mathbf{S}\varphi(\{x\}) = \{\varphi(x)\}$	Monad
$\mathbf{C}(\tau) = (\tau \to \mathbf{t}) \to \mathbf{t}$	$\mathbf{C}\varphi(x) = \lambda c.x(\lambda a.c(\varphi a))$	Monad
$\mathbf{T}(\tau) = \mathbf{s} \to (\tau \times \mathbf{s})$	$\mathbf{D}\varphi(\lambda s. \{\langle x, x + s \rangle \,|\, px\}) = \lambda s. \langle \varphi x, \varphi x + s \rangle$	Monad
$\mathbf{F}(\tau) = \tau \times \mathbf{S}\tau$	$\mathbf{F}\varphi(\langle v, \{x \,|\, x \in D_e\} \rangle) = \langle \varphi(v), \{x \,|\, x \in D_e\} \rangle$	Monad
$\mathbf{D}(\tau) = \mathbf{s} \to \mathbf{S}(\tau \times \mathbf{s})$	$\mathbf{D}\varphi(\lambda s. \{\langle x, x + s \rangle \,|\, px\}) = \lambda s. \{\langle \varphi x, \varphi x + s \rangle \,|\, px\}$	Monad
$\mathbf{M}(\tau) = \tau + \bot$	$\mathbf{M}\varphi(x) = \begin{cases} \varphi(x) & \text{if } \Gamma \vdash x : \tau \\ \# & \text{if } \Gamma \vdash x : \# \end{cases}$	Monad

Yale ENS | PSL★

CFG of English

```
CP    ::=  DP, VP
       |   Cmp, CP
       |   CP, CBar


CBar  ::=  Cor, CP


Dbar  ::=  Cor, DP


DP    ::=  DP, Dbar
       |   Dmp, DP
       |   Det, NP
       |   Gen, TN


Gen   ::=  DP, GenD
```

```
NP    ::=  AdjP, NP
       |   NP, AdjP


AdjP  ::=  TAdj, DP
       |   Deg, AdjP


VP    ::=  TV, DP
       |   AV, CP
       |   VP, AdvP


TV    ::=  DV, DP


AdvP  ::=  TAdv, DP
```

Yale   ENS | PSL★

CFG for Parsing

$$>, \beta \quad ::= \quad (\alpha \to \beta), \alpha$$

$$<, \beta \quad ::= \quad \alpha, (\alpha \to \beta) \qquad \mathsf{A}_{\mathtt{F}}(\alpha, \beta) \quad ::= \quad \mathtt{F}\alpha, \mathtt{F}\beta$$

$$\mathsf{UR}_{\mathtt{F}}(\alpha \to \alpha', \beta) \quad ::= \quad \mathtt{F}\alpha \to \alpha', \beta$$

$$\mathsf{J}_{\mathtt{F}} \; \mathtt{F}\tau \quad ::= \quad \mathtt{FF}\tau \qquad\qquad \mathsf{UL}_{\mathtt{F}}(\alpha, \beta \to \beta') \quad ::= \quad \alpha, \mathtt{F}\beta \to \beta'$$

$$\mathsf{DN}_{\mathtt{C}} \; \tau \quad ::= \quad \mathtt{C}_{\tau}\tau \qquad\qquad \mathsf{C}_{\mathtt{LR}}(\mathtt{L}\alpha, \mathtt{R}\beta) \quad ::= \quad (\alpha, \beta)$$

$$\mathsf{ER}_{\mathtt{R}}(\mathtt{R}(\alpha \to \alpha'), \beta) \quad ::= \quad \alpha \to \mathtt{R}\alpha', \beta$$

$$\mathsf{ML}_{\mathtt{F}}(\alpha, \beta) \quad ::= \quad \mathtt{F}\alpha, \beta \qquad \mathsf{EL}_{\mathtt{R}}(\alpha, \mathtt{R}(\beta \to \beta')) \quad ::= \quad \alpha, \beta \to \mathtt{R}\beta'$$

$$\mathsf{MR}_{\mathtt{F}}(\alpha, \beta) \quad ::= \quad \alpha, \mathtt{F}\beta$$

Combinator Denotations

$$>= \lambda\varphi.\lambda x.\varphi x$$

$$<= \lambda x.\lambda\varphi.\varphi x$$

$$\mathsf{ML_F} = \lambda M.\lambda x.\lambda y.(\mathtt{fmap_F}\lambda a.M(a,y))x$$

$$\mathsf{MR_F} = \lambda M.\lambda x.\lambda y.(\mathtt{fmap_F}\lambda b.M(x,b))y$$

$$\mathsf{A_F} = \lambda M.\lambda x.\lambda y.(\mathtt{fmap_F}\lambda a.\lambda b.M(a,b))(x)\mathtt{<*>}y$$

$$\mathsf{UL_F} = \lambda M.\lambda x.\lambda\varphi.M(x,\lambda b.\varphi(\eta_\mathbf{F}b))$$

$$\mathsf{UR_F} = \lambda M.\lambda\varphi.\lambda y.M(\lambda a.\varphi(\eta_\mathbf{F}a),y)$$

$$\mathsf{J_F} = \lambda M.\lambda x.\lambda y.\mu_\mathbf{F}M(x,y)$$

$$\mathsf{C_{LR}} = \lambda M.\lambda x.\lambda y.\varepsilon_\mathbf{LR}(\mathtt{fmap_L}(\lambda l.\mathtt{fmap_R}(\lambda r.M(l,r))(y))(x))$$

$$\mathsf{EL_R} = \lambda M.\lambda\varphi.\lambda y.M(\Upsilon_\mathbf{R}\varphi,y)$$

$$\mathsf{ER_R} = \lambda M.\lambda x.\lambda\varphi.M(x,\Upsilon_\mathbf{R}\varphi)$$

$$\mathsf{DN_\Downarrow} = \lambda M.\lambda x.\lambda y.\Downarrow M(x,y)$$

Bibliography I

[BC25]   Dylan Bumford and Simon Charlow. *Effect-Driven Interpretation: Functors for Natural Language Composition*. Mar. 2025.

[CSC10]   Bob Coecke, Mehrnoosh Sadrzadeh and Stephen Clark. *Mathematical Foundations for a Compositional Distributional Model of Meaning*. Mar. 2010. DOI: 10.48550/arXiv.1003.4394. arXiv: 1003.4394 [cs].

Bibliography II

[DV22]   Antonin Delpeuch and Jamie Vicary. *Normalization for Planar String Diagrams and a Quadratic Equivalence Algorithm*. Jan. 2022. DOI: 10.48550/arXiv.1804.07832. arXiv: 1804.07832.

[HM23]   Ralf Hinze and Dan Marsden. *Introducing String Diagrams: The Art of Category Theory*. 1st ed. Cambridge University Press, July 2023. ISBN: 978-1-00-931782-5 978-1-00-931786-3. DOI: 10.1017/9781009317825.

Bibliography III

[JS91]   André Joyal and Ross Street. "The Geometry of Tensor Calculus, I". In: *Advances in Mathematics* 88.1 (July 1991), pp. 55–112. ISSN: 00018708. DOI: 10.1016/0001-8708(91)90003-P.

[Mar]   Marcolli, Matilde et Chomsky, Noam et Berwick, Robert C. *Mathematical Structure of Syntactic Merge*.

Bibliography IV

[Mog89]   E. Moggi. "Computational Lambda-Calculus and
           Monads". In: *[1989] Proceedings. Fourth Annual
           Symposium on Logic in Computer Science*. Pacific
           Grove, CA, USA: IEEE Comput. Soc. Press, 1989,
           pp. 14–23. ISBN: 978-0-8186-1954-0. DOI:
           10.1109/LICS.1989.39155.

Bibliography V

[MZ25]   Paul-André Melliès and Noam Zeilberger. "The Categorical Contours of the Chomsky-Sch\"{u}tzenberger Representation Theorem". In: *Logical Methods in Computer Science* Volume 21, Issue 2 (May 2025), p. 13654. ISSN: 1860-5974. DOI: 10.46298/lmcs-21(2:12)2025.

[Sel10]   Peter Selinger. "A Survey of Graphical Languages for Monoidal Categories". In: vol. 813. 2010, pp. 289–355. DOI: 10.1007/978-3-642-12821-9_4. arXiv: 0908.3347 [math].

Bibliography VI

[SM25]   Isabella Senturia and Matilde Marcolli. *The Algebraic Structure of Morphosyntax*. June 2025.

[WSH14]   Nicolas Wu, Tom Schrijvers and Ralf Hinze. "Effect Handlers in Scope". In: *Proceedings of the 2014 ACM SIGPLAN Symposium on Haskell*. Haskell '14. New York, NY, USA: Association for Computing Machinery, Sept. 2014, pp. 1–12. ISBN: 978-1-4503-3041-1. DOI: 10.1145/2633357.2633358.