Effect-Driven Parsing

Formal studies on a categorical approach to semantic parsing

Matthieu Boyer

9th September 2025

École Normale Supérieure | Yale University

Plan

Context

- Given meanings for words, what is the meaning of a sentence?

Context

- Given meanings for words, what is the meaning of a sentence?
- We provide a categorical type and effects system for natural-language semantics parsing (following [BC25]).
- We 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:

$$\textbf{a cat} = \{c \mid \textbf{cat}\, c\} \tag{Set}$$

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

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-threaded** computations.

Theoretically, they are the duals of diagrams in a 2-category.

String Diagrams

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

Theoretically, they are the duals of diagrams in a 2-category.

Used as a **tool for parsing** in [CSC10].

Other Categorical Theories

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

[MZ25] use operads to prove results on CFGs.

Notations

- We use a language $\mathcal{L}$ of denotationally composed words.
- Start from a base typing CCC $\mathcal{C}$ and effects (functors) $\mathcal{F}(\mathcal{L})$.

Notations

- We use a language $\mathcal{L}$ of denotationally composed words.
- Start from a base typing CCC $\mathcal{C}$ and effects (functors) $\mathcal{F}(\mathcal{L})$.
- Our typing CCC $\bar{\mathcal{C}}$ is the closure of $\mathcal{C}$ under $\mathcal{F}(\mathcal{L})^*$, products and exponentials.

Our types are all (effect-sequence, base type) combos, with functions/products.

Intuitionistic-style Typing Judgements

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

$$\frac{\Gamma \vdash x : F\tau_1 \qquad \Gamma \vdash \varphi : \tau_1 \to \tau_2}{\Gamma \vdash \varphi x : F\tau_2}\,\texttt{fmap}$$

Intuitionistic-style Typing Judgements

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

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

Intuitionistic-style Typing Judgements

For natural transformations:

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

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

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

Presentation

To present a language in our formalism, we need:

- A syntax;
- A typed dictionary using effects;
- A typed lexicon of non-verbal constructs.

Introducing Higher-Order Constructs

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

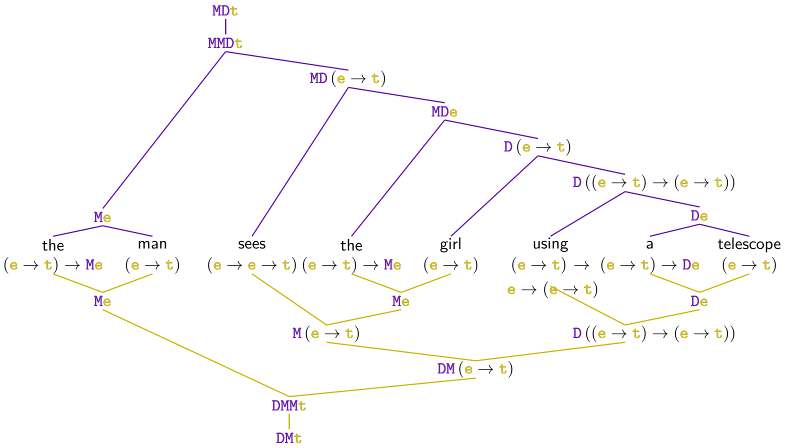
Introducing Higher-Order Constructs

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

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

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

Ambiguity


MDt
MMDt
MD (e → t)
MDe
D (e → t)
D ((e → t) → (e → t))
Me
De
the
man
sees
the
girl
using
a
telescope
(e → t) → Me
(e → t)
(e → e → t)
(e → t) → Me
(e → t)
(e → t) →
(e → t) → De
(e → t)
e → (e → t)
Me
Me
De
M (e → t)
D ((e → t) → (e → t))
DM (e → t)
DMMt
DMt

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.

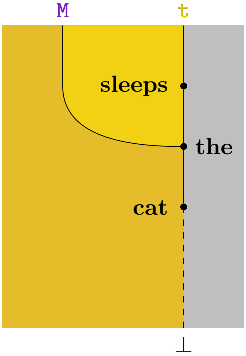
Handlers

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.



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.
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 to model our typing system: each typing judgement is associated with a **combinator** describing the way to combine two phrases when parsing.

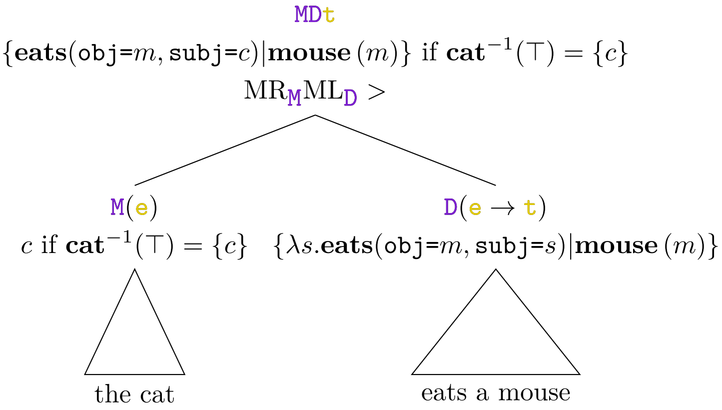
Parsing Algorithm

Typing syntax trees is exponentially too long.

We use a Context-Free Grammar to model our typing system: each typing judgement is associated with a **combinator** describing the way to combine two phrases when parsing.

This gives a complexity in $\mathcal{O}(|\mathcal{F}(\mathcal{L})|\,|\mathcal{S}|\,n^3)$.

Semantic Parse Trees I



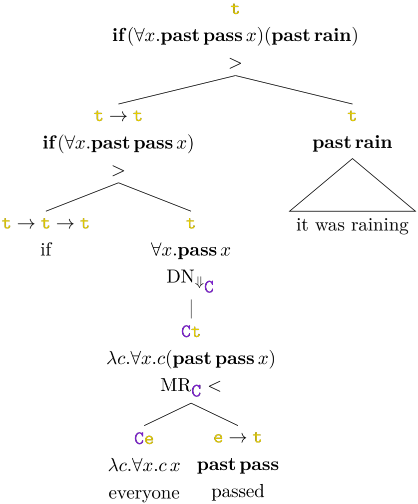
Semantic Parse Trees II

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

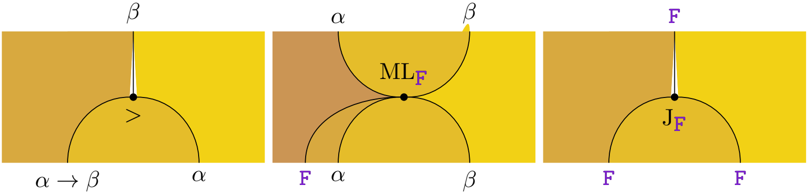
$$(\mathtt{e} \to \mathtt{t}) \to \mathtt{De} \qquad\qquad \mathtt{D}(\mathtt{e} \to \mathtt{t})$$
$$\mathrm{a} \qquad\qquad \lambda x.\mathbf{cat}\, x \wedge \mathbf{in\ a\ box}\, x$$

cat in a box

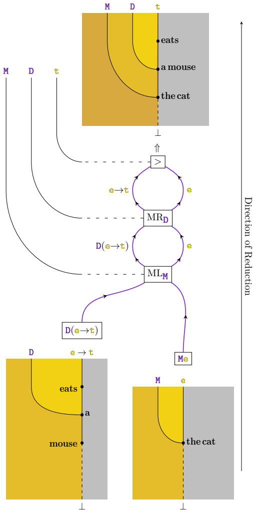
Semantic Parse Trees III



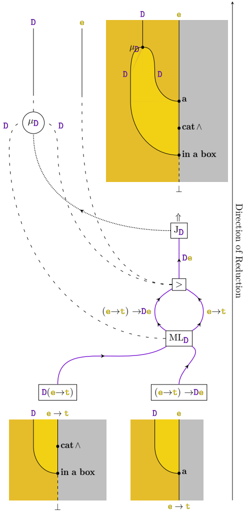
Combinators as String Diagrams



A Parsing Diagram Step

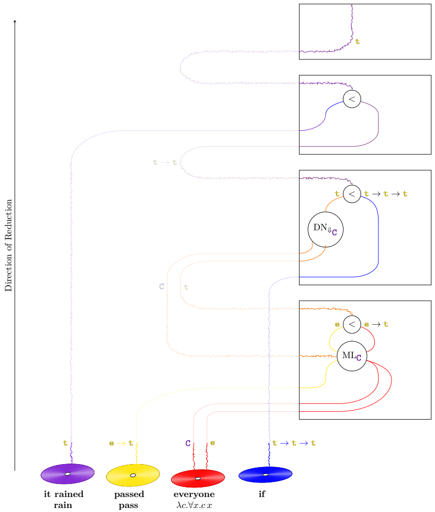


Another Parsing Diagram Step



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).

Thank you for your attention.

Do you have any questions?

Lexicon

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

Functor Denotations

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

Plural Functor

CN(P)	$\Gamma \vdash p : (\mathtt{e} \to \mathtt{t})$	$\Pi(p) = \lambda x.\,(px \wedge \lvert x \rvert \geq 2)$
ADJ(P)	$\Gamma \vdash p : (\mathtt{e} \to \mathtt{t})$	$\Pi(p) = \lambda x.\,(px \wedge \lvert x \rvert \geq 2)$
	$\Gamma \vdash p : (\mathtt{e} \to \mathtt{t}) \to (\mathtt{e} \to \mathtt{t})$	$\Pi(p) = \lambda \nu.\lambda x.\,(p\,(\nu)\,(x) \wedge \lvert x \rvert \geq 2)$
NP	$\Gamma \vdash p : \mathtt{e}$	$\Pi(p) = p$
	$\Gamma \vdash p : (\mathtt{e} \to \mathtt{t}) \to \mathtt{t}$	$\Pi(p) = \lambda \nu.p\,(\Pi \nu)$
IV(P)/VP	$\Gamma \vdash p : \mathtt{e} \to \mathtt{t}$	$\Pi(p) = \lambda o.\,(po \wedge \lvert x \rvert \geq 2)$
TV(P)	$\Gamma \vdash p : \mathtt{e} \to \mathtt{e} \to \mathtt{t}$	$\Pi(p) = \lambda s.\lambda o.\,(p\,(s)\,(o) \wedge \lvert s \rvert \geq 2)$
REL(P)	$\Gamma \vdash p : \mathtt{e} \to \mathtt{t}$	$\Pi(p) = \lambda x.\,(px \wedge \lvert x \rvert \geq 2)$
DET	$\Gamma \vdash p : (\mathtt{e} \to \mathtt{t}) \to ((\mathtt{e} \to \mathtt{t}) \to \mathtt{t})$	$\Pi(p) = \lambda \nu.p\,(\Pi \nu)$
	$\Gamma \vdash p : (\mathtt{e} \to \mathtt{t}) \to \mathtt{e}$	$\Pi(p) = \lambda \nu.p\,(\Pi \nu)$

CFG of English

$$\begin{array}{lcl}
\texttt{CP} & ::= & \texttt{DP, VP} \\
 & \mid & \texttt{Cmp, CP} \\
 & \mid & \texttt{CP, CBar} \\
\\
\texttt{CBar} & ::= & \texttt{Cor, CP} \\
\\
\texttt{Dbar} & ::= & \texttt{Cor, DP} \\
\\
\texttt{DP} & ::= & \texttt{DP, Dbar} \\
 & \mid & \texttt{Dmp, DP} \\
 & \mid & \texttt{Det, NP} \\
 & \mid & \texttt{Gen, TN} \\
\\
\texttt{Gen} & ::= & \texttt{DP, GenD}
\end{array}$$

$$\begin{array}{lcl}
\texttt{NP} & ::= & \texttt{AdjP, NP} \\
 & \mid & \texttt{NP, AdjP} \\
\\
\texttt{AdjP} & ::= & \texttt{TAdj, DP} \\
 & \mid & \texttt{Deg, AdjP} \\
\\
\texttt{VP} & ::= & \texttt{TV, DP} \\
 & \mid & \texttt{AV, CP} \\
 & \mid & \texttt{VP, AdvP} \\
\\
\texttt{TV} & ::= & \texttt{DV, DP} \\
\\
\texttt{AdvP} & ::= & \texttt{TAdv, DP}
\end{array}$$

CFG for Parsing

$$\begin{aligned}
>, \beta &::= (\alpha \to \beta), \alpha \\
<, \beta &::= \alpha, (\alpha \to \beta) \\
\mathsf{J}_{\mathtt{F}}\ \mathtt{F}\tau &::= \mathtt{F}\mathtt{F}\tau \\
\mathsf{DN}_{\mathtt{C}}\ \tau &::= \mathtt{C}_{\tau}\tau \\
\mathsf{ML}_{\mathtt{F}}(\alpha, \beta) &::= \mathtt{F}\alpha, \beta \\
\mathsf{MR}_{\mathtt{F}}(\alpha, \beta) &::= \alpha, \mathtt{F}\beta
\end{aligned}$$

$$\begin{aligned}
\mathsf{A}_{\mathtt{F}}(\alpha, \beta) &::= \mathtt{F}\alpha, \mathtt{F}\beta \\
\mathsf{UR}_{\mathtt{F}}(\alpha \to \alpha', \beta) &::= \mathtt{F}\alpha \to \alpha', \beta \\
\mathsf{UL}_{\mathtt{F}}(\alpha, \beta \to \beta') &::= \alpha, \mathtt{F}\beta \to \beta' \\
\mathsf{C}_{\mathtt{LR}}(\mathtt{L}\alpha, \mathtt{R}\beta) &::= (\alpha, \beta) \\
\mathsf{ER}_{\mathtt{R}}(\mathtt{R}(\alpha \to \alpha'), \beta) &::= \alpha \to \mathtt{R}\alpha', \beta \\
\mathsf{EL}_{\mathtt{R}}(\alpha, \mathtt{R}(\beta \to \beta')) &::= \alpha, \beta \to \mathtt{R}\beta'
\end{aligned}$$

Combinator Denotations

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

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

$$\mathsf{ML}_{\mathtt{F}} = \lambda M.\lambda x.\lambda y.(\mathtt{fmap}_{\mathtt{F}}\lambda a.M(a,y))x$$

$$\mathsf{MR}_{\mathtt{F}} = \lambda M.\lambda x.\lambda y.(\mathtt{fmap}_{\mathtt{F}}\lambda b.M(x,b))y$$

$$\mathsf{A}_{\mathtt{F}} = \lambda M.\lambda x.\lambda y.(\mathtt{fmap}_{\mathtt{F}}\lambda a.\lambda b.M(a,b))(x)\mathtt{<*>}y$$

$$\mathsf{UL}_{\mathtt{F}} = \lambda M.\lambda x.\lambda\varphi.M(x,\lambda b.\varphi(\eta_{\mathtt{F}}b))$$

$$\mathsf{UR}_{\mathtt{F}} = \lambda M.\lambda\varphi.\lambda y.M(\lambda a.\varphi(\eta_{\mathtt{F}}a),y)$$

$$\mathsf{J}_{\mathtt{F}} = \lambda M.\lambda x.\lambda y.\mu_{\mathtt{F}}M(x,y)$$

$$\mathsf{C}_{\mathtt{LR}} = \lambda M.\lambda x.\lambda y.\varepsilon_{\mathtt{LR}}(\mathtt{fmap}_{\mathtt{L}}(\lambda l.\mathtt{fmap}_{\mathtt{R}}(\lambda r.M(l,r))(y))(x))$$

$$\mathsf{EL}_{\mathtt{R}} = \lambda M.\lambda\varphi.\lambda y.M(\Upsilon_{\mathtt{R}}\varphi,y)$$

$$\mathsf{ER}_{\mathtt{R}} = \lambda M.\lambda x.\lambda\varphi.M(x,\Upsilon_{\mathtt{R}}\varphi)$$

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

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.