Categorical Compositional Distributional Questions, Answers & Discourse Analysis

Alexis TOUMI

New College

University of Oxford

Submitted in partial completion of the

M.Sc. in Mathematics and Foundations of Computer Science

September 2018

Acknowledgements

This thesis can be seen as an extended version of the paper which I will be presenting at the CAPNS workshop the day of the deadline. Hence I shall first acknowledge how much I owe to my co-authors: my supervisers Bob and Dan, my friend Giovanni without whose constant cooperation I would not have managed.

I would also like to thank F. Genovese, S. Abramsky, M. Sadrzadeh and A. Delpeuch (in chronological order) as well as the other (present and former) members of the Quantum Group for so many helpful pieces of dialogues in the process of writing this dissertation. I will especially remember the "linguistics meetings" and how much brain storming can happen from staring hours long at something like

> A cat is on the mat.
> It wants to catch a mouse.
> It runs away.

My thoughts and prayers go to Astrid, my mother Godeleine, my father Imad, my siblings Maylis, Raphael and Xavier (in alphabetical order) — as well as every one else who had the patience to hear me try and explain general abstract nonsense to them. To my friends Adelin, Antoine and my fellow master students who had the patience to try and explain general abstract nonsense to me.

I could never be sure I had grasped a concept until I managed to make it understood to someone else, and I had no way to know if my work had any interest without getting other people interested too.

Finally, I will never thank enough my little cousin Reda for asking good questions and my grand mother Habiba for feeding my soul and my body.

Abstract

After giving a review of the literature in Categorical Compositional Distributional (*DisCoCat* in reverse functional notation) models of natural language, we extend this framework from the meaning of isolated sentences to that of *discourses*, i.e. sequences of sentences in context.

In chapter 1 we introduce the necessary categorical machinery to compute the semantics of individual sentences, fixing the notation used in the rest of the thesis. In chapter 2 we show how this same machinery can be turned into a framework for *knowledge graph querying* — we present *string diagrams* and *dagger structures* as a tool for modeling the *duality* between questions and their answers. We apply the DisCoCat framework on a fragment of natural language we call *basic*, and propose to look at the isomorphism between basic *free corpora* and *knowledge graphs* as a process for language understanding. In chapter 3 we attempt to model *ambiguity* and *reference* as a constrained optimisation problem.

Contents

0.1 Functional and Imperative 5
0.2 Algebra and Geometry 5
0.3 Mind and Body 5
0.4 World as Functor 5

1 Axioms for DisCoCat Models of Meaning 7
1.1 Syntax: Pregroup Grammars 8
1.2 Semantics: Encoding Matrices 12
1.3 Functors of Rigid Monoidal Categories 17
1.4 Alice Loves Bob and Biproducts 22

2 Diagrams of Questions and Answers 23
2.1 String Diagrams for Natural Language 24
2.2 Object and Subject Questions as Effects 30
2.3 Knowledge Graphs and Free Corpora 34

3 Ambiguity and Semantic Unification 39
3.1 Anaphora and Basic Ambiguous Discourses 39
3.2 Resolution as Constrained Optimisation 41
3.3 Towards Semantic Unification 42

Appendices

A Towards Compositional Distributional Discourse Analysis 47
A.1 Diagrams of Matrices for Knowledge Graphs 50
A.2 Diagrams for Distributional Compositional Semantics 53
A.3 From Questions and Discourse to Knowledge Graph Queries 55
A.4 Language Understanding as a Process 57

References 63

Introduction

This thesis may be read as a process: it takes a reader with whom we share a common language — that of mathematical equations and symbolic reasoning — and uses this shared language as a resource to produce some thoughts about *ambiguity*. In order to produce such a resource, we need tools to make sure that the words we use in common refer to the same thing, i.e. that the thoughts they produce are in some sense *the same*, or at least *similar*.

We leave ambiguity itself as a third uncompleted chapter. Instead, this thesis focuses on making sure that our language for reasoning about ambiguity is not itself ambiguous — which would quickly lead to circularity of thoughts. Thus we will give a particular attention to the words and symbols we use, trying to make sure that they are used in a sense defined either in this thesis, in the references at the end of it, or in the dictionnary. When the same word or symbol refers to two different things, we try to make it clear from context which meaning applies. For example, the following strings of symbols (which should look familiar to a working mathematician):

$$\begin{aligned} s &= \langle x_0, \ldots, x_n \rangle \\ t &= \sum_{x \in s} f(x) \end{aligned}$$

denotes the process of applying the function f to the n elements of the sequence s then adding them together, calling the result t. Note that we can get rid of the s altogether and denote the same process as one equation:

$$t = \sum_{i \leq n} f(x_i)$$

Our problem is that no two mathematicians will ever give you the same definition for Σ — i.e. the symbols and the words they would use will probably not look or sound similar — yet they are talking about *the same* Σ. Even though they do not have the same written shape, using these two pieces of mathematics will lead to the same result, given that the function f and the sequence $\langle x_0, \ldots, x_n \rangle$ are the same in both cases. Calling them "equal" has a name: *extensionality* — two sets are *the same* when their elements are *the same*. Hence mathematicians usually live on their daily life calling these two capital sigmas "the same" without asking too many questions.

"Foundations" — the word between "Mathematics" and "Computer Science" in the name of this M.Sc. — refers to a research program which may be summed up as: asking too many questions. Or rather: asking *how* are things the same and counting *in how many ways* they are instead of merely asking *whether* they are the same. However, asking what is the same or not the same is *not* the aim of this thesis, instead we propose to ask a weaker question:

What is similar *to what?*

In the first chapter, we present of *Categorical Compositional Distributional*[1] models of meaning as a framework for computing the similarity between words. We focus on the *axioms* for DisCoCat models, with an emphasis on the concepts of *rigidity* and *freeness*. In order to give the meaning of individual words we look at them *in context*, i.e. we compute similarities from the way words appear in a given corpus — a set of natural language documents. From right to left we can sum up

1. *distributionality* as the fact that similar words appear in similar contexts,
2. *compositionality* as the idea that the similarity of sentences should depend on the shape of the sentences and the similarity of the words,
3. *category theory* as the machinery we use in order to unify these two principles.

We will proceed starting from the middle, defining the *syntax* of a small fragment of the English language in terms of *pregroup grammars* — a mathematical framework introduced by Lambek in *From Word To Sentence* [1]. Our aim is to spell out only those axioms that are justified from a cognitive perspective, and argue that the fragment of natural language we study is *universal* in the sense of *universal grammar* introduced by Chomsky in *Syntactic Structures* [2] then further formalised by Montague in *The Proper Treatment of Quantification in Ordinary English* [3].

We then define the *semantics* of individual words in terms of *encoding matrices*. These matrices encode the data extracted from the corpus that will be used in computing word similarity. Again we aim at giving only the definitions that are necessary — those we can justify linguistically — and give a new formulation of DisCoCat models which lifts the meaning of sentences to that of documents: sequences of sentences, which we call *discourses*. We argue that this new formulation is furthermore *natural*, in a sense that is discussed in the third chapter — i.e. yet to be investigated. We conclude with a *universal example* of a *natural language* sentence:

"Alice loves Bob."

and argue that this basic example leads to the idea that we should *update* our model. We define a last piece of structure which is required in order to do so: *biproducts*.

The second chapter may be read as an attempt to answer the following question:

Can we build a machine that answers human questions?

which we will reformulate as

Can we build a machine that, given a question
gives an answer similar *to what humans would give?*

(Given that they are given the same *question, obviously.)*

[1] shortened as *DisCoCat*, for a reason we will attempt to clarify

We begin this middle chapter by a philosophical interlude investigating *graphical ontologies* — which physicists would call *relational interpretations* — descriptions of *what is* in terms of *graphs* describing *what is related to what*. We show how in our context this leads to the idea that there should exist a *duality* between knowledge graphs in the *Resource Description Framework* (RDF) and natural language discourses which we call *basic*. We argue that this duality should be seen as a *process*: namely the update which concludes the previous chapter.

We then give an introduction to the mathematics underlying this duality in terms of *dagger structures* and *string diagrams*. We focus on giving the intuition behind the use of diagrammatic reasoning in our framework for natural language and cognition: that of matrices as *processes* which transform some input data into some output. Then duality can be seen as the process of *reversing processes* — i.e. taking the output to be the input, and vice-versa.

The middle of the middle chapter is a partial attempt at formalising the idea that an answer should be the reverse process of its question — for reference, we include the publication that lead to this idea as an appendix. The main personal motivation for this dissertation was to bridge the gap between it and three other pieces of work which the author has contributed to.

1. *Generalised Relations in Linguistics and Cognition*, Coecke et al. [4] where the DisCoCat framework is formulated in an abstract fashion in terms of *hypergraph categories*. Reasoning about family trees is taken as an example for introducing *internal monads* as a tool for modeling natural language.

2. *A Tool for the Automated Verification of Nash Equilibria in Concurrent Games*, Toumi et al. [5] where *temporal logic* formulae represent the goals of *agents* behaving strategically in a system represented as a *theoretical game* between abstract machines. We compute *Nash equilibria* as the outcomes for the game that are in some sense *reasonable*.

3. *From Model Checking to Equilibrium Checking*, Wooldridge et al. [6] where the above is seen as a prototype for *rational verification*. Given a description of what agents can do and of what they want, we want to make sure that some global property holds in what is called a *multi-agent system*.

However in order to remain self-contained, this thesis will *not* mention either monads, games or temporal logic. We will not be able to avoid *Boolean logic*: conjunction, disjunction and implication[2] depicted by the symbols $\wedge$, $\vee$ and $\Longrightarrow$.

[2] but *not* negation

Context

A spectre is haunting Artificial Intelligence — the spectre of duality.

We leave only the traces this spectre has left, in the form of a bibliography:

0.1 Functional and Imperative

- M. Minsky and S. Papert

  Perceptrons: An Introduction to Computational Geometry [7]

0.2 Algebra and Geometry

Functors were first introduced to model the duality between algebra and geometry.

- S. M. Lane

  Categories for the Working Mathematician [8]

This duality can be traced back to Descartes and his system of coordinates: points in the plane are pairs of numbers on a line. The same Descartes is also known for his dualism in metaphysics.

0.3 Mind and Body

- Spinoza

  Ethics [9]

  > "The order and connection of ideas is *the same*[3]
  > as the order and connection of things." — II.VII

0.4 World as Functor

- Wittgenstein

  Tractacus Logico Philosophicus [10]

  > "*The limits of my language*[4] mean the limit of my world" — 5.6

[3] here we add the emphasis

[4] here we leave the emphasis

And what is the thought that lies behind the words
"This is a very pleasant pineapple"?
This is a very pleasant pineapple.

Listen to me.

We imagine the meaning of what we say as
something queer, mysterious, hidden from view.
But nothing is hidden! Everything is open to view!
It's just... it's just philosophers who muddy the waters.

— Derek Jarman, *Wittgenstein (1993)*

1

Axioms for DisCoCat Models of Meaning

In this chapter we give a self-contained presentation of DisCoCat models of meaning which aims at *interdisciplinarity*. Thus, we will formulate our model in the lingua franca which is shared today by scientific disciplines from computer science to theoretical linguistics through physics, economics and engineering: basic set theory. Cutting through mathematical jargon with Occam's razor, we attempt to introduce only the notions that are necessary, and to give precise definitions of every term we use. We only assume the reader to understand the words "sequence", "set" and "function".

Previous dissertations have already given excellent introductions that we would qualify as *top-down* — as opposed to the *bottom-up* approach we try to follow here — defining DisCoCat models in terms of *dagger symmetrical monoidal functor*, for example see [11]. However enlightening this abstract approach can be — the bridge between syntax and semantics at the core of DisCoCat would not have been formulated without it — we believe that it can both hide the simplicity of the model and put off scientists with less mathematical backgrounds.

Whenever a concrete definition was possible, we have prefered it over its abstract counterpart: for example we only talk about concrete matrices, never about abstract vector spaces and linear maps. Moreover all the sets we work with are countable, with the notable exception of the set of real numbers $\mathbb{R}$ — which we use in one example as a convenient workaround to what should really be floating-point arithmetic. As a warning to the pure category theorist who might read this thesis: we will make heavy use of equalities and you will not hear about natural transformations.

The only concept which we have not been able to either completely avoid or give a precise and general definition of is that of *freeness*. We apologise to the reader: making concepts sound hard is easy, making them sound easy is a lot of hard work.

1.1 Syntax: Pregroup Grammars

We begin with a few mathematical definitions, fixing the notation used in this thesis. Given a natural number $a \in \mathbb{N}$, we will write $\underline{a}$ for the set $\{\, 0, \ldots, a-1 \,\}$.

Definition 1.1 (Monoid)**.** *A* monoid *is a tuple* $(M, \times, 1)$ *where*

- M *is a set*
- $\times : M \times M \to M$ *is a binary function called* multiplication
- $1 \in M$ *is an element of* M *called the* unit

such that for all $a, b, c \in M$ *we have*

- $a \times 1 = a = 1 \times a$ *(unitality)*
- $a \times (b \times c) = (a \times b) \times c$ *(associativity)*

Thanks to associativity we can forget parenthesis and write $a \times b \times c$ *with no ambiguity. When the multiplication is clear from context we omit the symbol* $\times$ *and simply write* $abc \in M$*. For monoids as well as for the other algebraic structures we consider, we will overload the notation and write* M *for both the monoid and its set of elements.*

Given any set X, the set X^* of all finite sequences $\langle x_0, \ldots, x_n \rangle$ of elements in X is a monoid with concatenation as multiplication and the empty sequence as unit, this is called the *free monoid* generated by X. The free monoid X^* has the special property that for any monoid $(Y, \times, 1)$ and function $f : X \to Y$ there is a unique function $f^* : X^* \to Y$ such that for all $x \in X$ and $\langle a_0, \ldots, a_m \rangle, \langle b_0, \ldots, b_n \rangle \in X^*$

- $f^*\langle\rangle = 1$ and $f^*\langle x \rangle = f(x)$
- $f^*\langle a_0, \ldots, a_m, b_0, \ldots b_n \rangle = f^*\langle a_0, \ldots, a_m \rangle \times f^*\langle b_0, \ldots b_n \rangle$

Intuitively, $f^* : X^* \to Y$ takes a sequence of elements in X and applies the function f element-wise to get a sequence of elements in Y, then multiplies these elements together to get a single element in Y. We will make heavy use of this property, as well as of its extension to other algebraic structures.

Remark 1.1. *For simplicity we will allow ourselves to consider the set of all finite sets as a monoid, with the Cartesian product* $\times$ *as multiplication and the singleton* $\underline{1}$ *as unit. That is, we treat tuples as lists and consider the sets* $X \times \underline{1}$*,* X *and* $\underline{1} \times X$ *as equal — not only isomorphic. However we will need some set* $\{\, l, r \,\}$ *such that* $X \times \{\, l \,\} \neq X \times \{\, r \,\}$*.*

Definition 1.2 (Poset)**.** *A* preorder *is a tuple* $(P, \leq)$ *where* P *is a set and* $\leq$ *is a subset of* $P \times P$ *such that for all* $p, q, r \in P$ *we have*

- $p \leq p$ *(reflexivity)*
- $p \leq q \ \wedge \ q \leq r \implies p \leq r$ *(transitivity)*

For readability we use infix notation for membership in $\leq$ — i.e. $p \leq q$ iff $(p, q) \in \leq$.

Definition 1.3 (Poset)**.** *A* poset *(for partially-ordered set) is a preorder $(P, \leq)$ with the following extra axiom for all $p, q \in P$*

- $p \leq q \ \wedge \ q \leq p \implies p = q$ *(antisymmetry)*

Example 1.1. *A free monoid X^* has the structure of a poset with* infix ordering*:*

$$\langle x_0, \ldots, x_n \rangle \leq \langle y_0, \ldots, y_m \rangle \iff \langle x_0, \ldots, x_n \rangle = \langle y_i, \ldots, y_j \rangle$$

ŵhere $i, j \leq n \leq m$, for example $\langle loves, Bob \rangle \leq \langle Alice, loves, Bob \rangle$. If we forget sequence order and multiplicity of elements, we get a poset on the powerset $\mathcal{P}(X)$ with subset inclusion as ordering.

Definition 1.4 (Graph)**.** *A graph is a tuple $G = (O, A, dom, cod)$ where O is a set of* objects*, A is a set of* arrows*, and the functions $dom, cod : A \to O$ are called the* domain *and* codomain *respectively. We write $f : a \to b$ for $f \in A$ when $dom(f) = a \wedge cod(f) = b$.*

Definition 1.5 (Category)**.** *A (small)* category *is a tuple $(G, \circ, id)$ where $G = (O, A, dom, cod)$ is called the* underlying *graph, $id : O \to A$ called the* identity *and*

$$\circ : \{ \, (f, g) \in A \times A \mid cod(f) = dom(g) \, \} \to A$$

is called the composition*. We write $id(a) = 1_a$ and $\circ(f, g) = gf$ and require that for all $f : a \to b, \ g : b \to c, \ h : c \to d$*

- $f1_a = f = 1_b f$ *(unitality)*
- $h(gf) = (hg)f$ *(associativity)*

As for monoids, associativity allows us to write $hgf \in A$ with no ambiguity. We will also write $Ob(\mathcal{C})$ for O, $Ar(\mathcal{C})$ for A and $\mathcal{C}(a, b)$ for the set of arrows $f : a \to b$.

A category is *discrete* when the only arrows are identities, every set X can equivalently be seen as a discrete category with itself as the set objects. Hence we overload the notation and write X for the corresponding discrete category. Categories are also generalisation of both monoids and posets: indeed a monoid is a category with one object (arrows are the elements, composition is the multiplication), a poset is a category with at most one arrow between any two objects (objects are the elements, existence of arrows is the subset membership) together with the antisymmetry axiom. In the context of categories, antisymmetry can be generalised to the axiom of *skeletality* which ensures that there are no two distinct isomorphic objects: for all $f : a \to b, \ g : b \to a$ we have

$$gf = 1_a \ \wedge \ fg = 1_b \implies a = b$$

Definition 1.6 (Functor)**.** *Given two categories $\mathcal{C}$ and $\mathcal{D}$, a* functor *$F : \mathcal{C} \to \mathcal{D}$ is a pair of functions $F_{Ob} : Ob(\mathcal{C}) \to Ob(\mathcal{D})$ and $F_{Ar} : Ar(\mathcal{C}) \to Ar(\mathcal{D})$ such that*

- *for all* $a \in Ob(\mathcal{C})$ *we have* $F_{Ar}(1_a) = 1_{F_{Ob}(a)}$
- *for all* $f : a \to b$ *we have* $F_{Ar}(f) : F_{Ob}(a) \to F_{Ob}(b)$
- $F_{Ar}(gf) = F_{Ar}(g)F_{Ar}(f)$ *when the composition* gf *is well-defined.*

When it is clear from context we will omit the underscript and overload the notation to write F *for both the object and arrow functions, as well as the functor itself.*

Remark 1.2. *In the special case of monoids, functors are called* homomorphisms*, in the case of partial orders they are better known as* monotone functions*.*

For every category $\mathcal{C}$ there exists a unique functor $!_{\mathcal{C}} : \mathcal{C} \to \underline{1}$: it sends all the objects and arrows of $\mathcal{C}$ to the unique object and arrow of $\underline{1}$. Given any object $c \in \mathcal{C}$ we can define a functor from $\underline{1}$ to $\mathcal{C}$ which maps 0 to c and 1_0 to 1_c. We overload the notation and denote this functor by $c : \underline{1} \to \mathcal{C}$

Remark 1.3. *We can define a category* **Cat** *with functors as arrows and small categories as objects: composition of functors is given by the composition of their object and arrow functions, the identity arrow for* $\mathcal{C}$ *is given by the* identity functor $1_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ *where both the object and arrow parts are taken to be identity functions. Associativity and unitality of functor composition follow from that of functions. However Russell's paradox prevents this category to be small itself.*

In order to reason formally about syntax, we will make use of an algebraic formalism introduced by Lambek in his seminal work *From Word to Sentence* [1]. The definition proceeds in three steps: 1) partially ordered monoids, 2) pregroups and 3) pregroup grammars. Finally we will define the basic grammar that will serve as example throughout this thesis.

Definition 1.7 (Pomonoid)**.** *A* pomonoid *(for partially-ordered monoid) is a tuple* $(G, \leq, \times, 1)$ *where* $(G, \leq)$ *is a poset,* $(G, \times, 1)$ *is a monoid and both are compatible in the sense that for all* $a, b, c, d \in G$ *we have:*

$$a \leq b \ \wedge \ c \leq d \implies a \times c \leq b \times d$$

Definition 1.8 (Pregroup)**.** *A* pregroup *is a pomonoid* G *equipped with a pair of functions* $(-)^l, (-)^r : G \to G$ *called the left and right adjoints respectively, such that for all* $a \in G$

$$\begin{array}{rcccl} a^l & \times & a & \leq 1 \leq & a \times a^l \\ a & \times & a^r & \leq 1 \leq & a^r \times a \end{array} \tag{1.1}$$

We call the elements of G grammatical types*, the elements of* $\leq$ reductions*.*

Remark 1.4. *Our definition differs slightly from the standard one in that we ask for adjoints to be functions instead of merely requiring existence of* $a^l, a^r \in G$ *for every* $a \in G$*. However the standard definition implies that adjoints are unique hence these two definitions are equivalent.*

We will need the following lemma from [1].

Lemma 1.1 (Contravariance)**.** *For all* $a, b \in G$, $a \leq b \implies b^l \leq a^l \ \wedge \ b^r \leq a^r$.

Proof. Assume $a \leq b$ then

$$b^l \ = \ b^l 1 \ \leq \ b^l a a^l \ \leq \ b^l b a^l \ \leq \ 1 a^l \ = \ a^l$$

and symmetrically for $b^r \leq a^r$. □

Definition 1.9 (Pregroup Grammar)**.** *A* pregroup grammar *is a tuple* (G, s, V, T) *where* G *is a pregroup with a designated element* $s \in G$ *called the* sentence type*,* V *is a set of natural language words called the* vocabulary *and* $T : V \to G$ *is a function assigning a grammatical type* $T(w) \in G$ *to every word* $w \in V$*.*

Definition 1.10 (Discourse and Corpus)**.** *We say that an* utterance*, i.e. a sequence of words,* $\langle w_0, \ldots, w_n \rangle \in V^*$ *is a* grammatical sentence *when* $T^* \langle w_0, \ldots, w_n \rangle \leq s$*. We call a sequence of sentences a* discourse *and a set of discourses a* corpus*, thus the set of corpora is* $\mathcal{P}(V^*)$*.*

Remark 1.5. *We will assume that words have a unique grammatical type, even though this is not necessarily true of natural languages: for example "type" is both a noun and a verb.*

We conclude this section by defining the basic pregroup grammar that will be used for the examples throughout this thesis. We start with a poset of *basic types* $B = (\{ s, n, obj, sub \}, \leq)$ — the latter three types standing for nouns, objects and subjects respectively — where

$$\leq \ = \{ (n, obj), (n, sub) \} \ \cup \ \{ (t, t) \mid t \in B \}$$

Then we define G as the *free pregroup* generated by the poset of *basic types* B. Defining freeness formally would lead us astray and is not essential to our discussion, for a construction of free pregroups see [1, p. 9]. We take our vocabulary to be $V = \{ Alice, \ Bob, \ loves, \ hates, \ and, \ who, \ whom, \ \ldots \}$ with

$$\begin{array}{rcll} T(Alice) & = & T(Bob) & = n \\ T(songs) & = & T(movies) & = obj \\ T(loves) & = & T(hates) & = sub^r \times s \times obj^l \\ & & T(and) & = n^r \times n \times n^l \\ & & T(who) & = n^r \times n \times sub^{ll} \times s^l \\ & & T(whom) & = n^r \times n \times obj^{ll} \times s^l \\ & & & \ldots \end{array}$$

Example 1.2. *The utterance "Alice loves Bob" is a grammatical sentence, indeed*

$$\begin{array}{rcccccccc} T^* \langle Alice, \ loves, \ Bob \rangle & = & n & sub^r & s & obj^l & n & & \\ & \leq & n & n^r & s & obj^l & n & (lemma \ 1.1) \\ & \leq & & & s & obj^l & n & (equation \ 1.1) \\ & \leq & & & s & n^l & n & (lemma \ 1.1) \\ & \leq & & & s & & & (equation \ 1.1) \end{array}$$

The basic example of grammar we use here covers only a small fragment of the English language: it does not deal with adjectives, intransitive verbs, grammatical gender, plural and singular; to cite but a few concepts that linguists have used to describe the utterances that natural language speakers would consider as grammatical. The analaysis of Lambek covers a much wider fragment of the English language, and it has been successfully applied to a number of other languages:

> "Since 1998, pregroup grammars have been applied to fragments of a number of languages: English, French, German, Italian, Turkish, Arabic, Polish, Latin and Japanese, and there is work in progress on Mandarin and Persian." — Lambek, *From Word To Sentence* [1, p. 74]

In this, he follows a long tradition which can be traced from Chomsky's seminal work *Syntactic Structures* [2] through Montague semantics and the idea of *universal grammar*. Indeed pregroup grammars have the same expressive power as *context-free grammars* [12], which are more widely known to be a foundational concept in both theoretical linguistics and computer science. Unraveling the universal structures of natural language is an ambitious research program which brings together philosophy, anthropology and cognitive science.

In this thesis we aim at describing syntax, semantics and the link between them from such a universal perspective. Hence we do not focus on any particular of the English language, taking examples that we believe are applicable to any other natural language. We make the hypothesis that an example of such a universal is the fact that the verb "love" needs a subject and an object to form a meaningful sentence.

On the other hand, two examples of phenomenon that are *not* universal in this sense — and which will not take care of in what follows — are the fact that the English "love" needs an "s" when we talk about the present of somebody else, and the fact that it is written and read from left to right.

1.2 Semantics: Encoding Matrices

Now in order to reason about semantics, we will use one of the most ubiquitous tools in mathematics and engineering: *matrices*. We define categories where the arrows are matrices and the objects are their dimensions, i.e. their number of rows and columns. These categories are parametrised on the *scalars* used for the matrix entries: intuitevely a set of numbers with a notion of addition and multiplication. Composition of arrows is given by matrix multiplication, in order for it to be well-behaved we require our sets of scalars to carry the structure of a *rig*.

Definition 1.11 (Rig)**.** *A* rig *(for "ri*N*g" without* N*egatives, also known as a* semiring*) is a pair of a monoid* $(S, \times, 1)$ *and a* commutative monoid $(S, +, 0)$ *such that for all* $a, b, c \in S$

- $a + b = b + a$ *(commutativity)*
- $a \times 0 = 0 = 0 \times a$ *(absorption)*

- $a \times (b + c) = a \times b + a \times c$ *(l-distributivity)*
- $(a + b) \times c = a \times c + b \times c$ *(r-distributivity)*

We do not *ask for* $(S, \times, 1)$ *to be commutative* yet.

Example 1.3. *The most basic example of a rig is the set of natural numbers* $\mathbb{N}$ *with addition and multiplication defined in the standard way, indeed it is the free rig generated by one element. Another example is the set of Booleans* $\mathbb{B} = \{0, 1\}$ *with* $\times = \wedge$ *and* $+ = \vee$*, which can be seen as the rig generated by one element and the equation* $1 + 1 = 1$*. Also any field is a rig hence the rationals* $\mathbb{Q}$*, the reals* $\mathbb{R}$ *and the complex numbers* $\mathbb{C}$ *are all examples of rigs.*

Definition 1.12 (Rig-valued Matrix)**.** *Given a rig* S *and a number of rows and columns* $a, b \in \mathbb{N}$*, an* $a \times b$ matrix *is a function* $\mathtt{M} : \underline{a} \times \underline{b} \to S$*. Although we will not need it in this thesis, matrices are usually represented as rectangular tables.*

Definition 1.13 (Element-Wise Addition)**.** *Given a pair of* $a \times b$ *matrices* $\mathtt{M}, \mathtt{N}$*, we overload the notation and define the* $a \times b$ element-wise addition *as* $\mathtt{M} + \mathtt{N} = (+)\ (\mathtt{M} \times \mathtt{N})\ \mu_{(\underline{a} \times \underline{b})} : \underline{a} \times \underline{b} \to Y$ *where for any set* X*,* $\mu_X : X \to X \times X$ *is given by the* copy*:*

$$\mu_X(x) \quad = \quad (x, x)$$

Definition 1.14 (Matrix Multiplication)**.** *Given a pair of matrices* $\mathtt{M}_0$ *and* $\mathtt{M}_1$ *of dimensions* $a \times b$ *and* $b \times c$ *respectively, their* multiplication $\mathtt{M}_1\mathtt{M}_0$ *is the* $a \times c$ *matrix*

$$(\mathtt{M}_1\mathtt{M}_0)(i, j) \quad = \quad \sum_{k<b} \mathtt{M}_0(i, k) \times \mathtt{M}_1(k, j)$$

Note that for consistency we use the same reversed notation as for composition.

Definition 1.15 (Transpose)**.** *Given an* $a \times b$ *matrix* $\mathtt{M}$*, we define its* transpose *as the matrix* $M^T = M\sigma_{\underline{b},\underline{a}}$ *where for any pair of sets* X *and* Y*,* $\sigma_{X,Y} : X \times Y \to Y \times X$ *is given by the* swap*:*

$$\sigma_{X,Y}(x, y) \quad = \quad (y, x)$$

Definition 1.16 (Scalar Product)**.** *Given a pair of states* $\mathtt{u}, \mathtt{v} : 1 \to a$*, their* scalar product *is the* 1×1 *matrix* $\mathtt{v}^T\mathtt{u}$*, which is the same as the scalar* $\mathtt{v}^T\mathtt{u}(0, 0) \in S$*.*

Definition 1.17 (Category of matrices)**.** *Given a rig* S*, we define* $\mathtt{Mat}_S$ *as the category with the natural numbers as objects and* $a \times b$ *matrices with values in* S *as arrows. Composition is given by multiplication and associativity follows from the rig axioms. The identity arrow for* $a \in \mathbb{N}$ *is given by* $\mathtt{1}_a = \delta_{\underline{a}}$ *where for any set* X $\delta_X : X \times X \to X$ *is given by the* Kronecker delta

$$\delta_X(x_0, x_1) \quad = \quad \begin{cases} 1 & \texttt{if } x_0 = x_1 \\ 0 & \texttt{otherwise} \end{cases}$$

Hence from now on we will use the arrow notation $\mathtt{M} : a \to b$ for $a \times b$ matrices. We will also call matrices $\mathtt{s} : 1 \to a$ *row vectors* or *states*, and matrices $\mathtt{t} : a \to 1$ *column vectors* or *effects*. We will denote particular examples of row vectors as sequences.

We conclude this section by giving the main example of how we use matrices for semantics: given some corpus $C \subseteq V^*$ (see definition 1.10), we encode the distributional semantics of $w \in V$ as a state $\mathcal{E}(C, w) : 1 \to n$ called the *word vector*, where $n \in \mathbb{N}$ will serve as a hyper-parameter of model. In our context, it will prove more convenient to reformulate this data using two processes known as *currying* and *indexing*.

Intuitively, currying is the observation that a function of pairs is the composition of a pair of functions — where in the process we go into *higher-order functions*, i.e. functions of functions. More elegantly put, we have an isomorphism

$$A \times B \to C \quad \simeq \quad A \to (B \to C)$$

where $(B \to C)$ denotes the set of functions from B to C. This isomorphism is furthermore *natural* in a sense that will be discussed in section 3.3. Once applied to our setup, we get the following natural isomorphism:

$$\begin{aligned} \mathcal{P}(V^*) \times V \to \mathtt{Mat}_S(1, n) &\simeq \mathcal{P}(V^*) \to (V \to \mathtt{Mat}_S(1, n)) \\ &= \mathcal{P}(V^*) \to (V \to (\underline{n} \to S)) \\ &\simeq \mathcal{P}(V^*) \to ((V \times \underline{n}) \to S)) \end{aligned}$$

where the first and third lines are called *currying* and *uncurrying* respectively, and in the second line we use the equality of remark 1.1.

Definition 1.18 (Indexing)**.** *An indexed set is a pair (X, i) where the bijection $i : X \to \underline{|X|}$ is called the* indexing*, the product of two indexed sets (X, i) and (Y, j) is given by $(X \times Y, i \star j)$ where*

$$(i \star j)(x, y) \quad = \quad j(y) + |Y| \times i(x)$$

their sum $(X + Y, i + j)$ is given by

$$X + Y \quad = \quad X \times \{\, l \,\} \;\cup\; Y \times \{\, r \,\}$$

$$(i + j)(x, l) \quad = \quad i(x) \qquad (i + j)(y, r) \quad = \quad |X| + j(y)$$

Definition 1.19 (One-Hot Vector)**.** *Given an indexed set (V, i), the* one-hot vectors *$|w\rangle : 1 \to |V|$ and $\langle w| : |V| \to 1$ for $w \in V$ are given by*

$$\langle w| \, (j, 0) \quad = \quad |w\rangle \, (0, j) \quad = \quad \delta_V(w, i^{-1}(j))$$

Note that, departing from its traditional use in physics, we only write set elements inside Dirac bra-ket notation to distinguish one-hot vectors from other vectors.

From this indexing, the naturality property mentioned above allows us to derive the following type for what we call the *encoding process*:

$$\mathcal{E} \quad : \quad \mathcal{P}(V^*) \to \mathtt{Mat}_S(|V|, n)$$

Although this can be seen as only a change in notation, we will argue that this formulation allows to give a high-level picture of *learning*: given a corpus $C \subseteq V^*$ we want to compute an *encoding matrix* $\mathtt{E} : |V| \to n$. From this isomorphism, we can also derive the following two statements:

$$\mathcal{E} \;:\; V \to (\mathcal{P}(V^*) \times \underline{n} \to S)$$

is an equivalent type of the encoding process and given an indexed set V we have

$$\mathtt{E} \;=\; \mathtt{E}' \;:\; |V| \to n \qquad \Longleftrightarrow \qquad \forall w \in V \;\cdot\; \mathtt{E}\,|w\rangle \;=\; \mathtt{E}'\,|w\rangle$$

While the latter is a standard fact of linear algebra, in our setup the former can be formulated in the following way. The process which takes a corpus and assigns a vector to each word is the same as the process which takes a word and assigns a vector to each corpus — i.e. computing the meaning of a corpus given the meaning of its words is the same as computing the meaning of the words given the meaning of the corpus.

If we can say anything at all — i.e. $V \neq \emptyset$ — then $\mathcal{P}(V^*)$ is an infinite set, i.e. there is no longest utterance, thus there can be no indexing and hence no encoding matrix for corpora, the vector for a corpus is *abstract*. However, even if we restricted ourselves to corpora less than a certain size — say that of the set of Wikipedia articles — this function is only a notational device and it would not make computational sense to store corpus vectors in memory. Instead, we formulate how our model gives the semantics of particular examples of discourses.

The purpose of the encoding matrix is to model a notion of *word similarity*: whereas we have $\langle w_i|w_j\rangle = 0$ whenever w_i and w_j are distinct words, we want the scalar product of $\mathtt{E}\,|cat\rangle$ an $\mathtt{E}\,|dog\rangle$ to tell us how similar "cat" is to "dog". In particular, we would like to say that "cat" is *more similar* to "dog" than "dog" is to "philosophy" — a statement which should appear as obvious to anyone who knows the meaning of these three words — and write

$$\langle philosophy|\,\mathtt{E}^T\mathtt{E}\,|dog\rangle \;\leq\; \langle dog|\,\mathtt{E}^T\mathtt{E}\,|cat\rangle$$

In order to talk about such degrees of similarity, we need to equip our set of scalars with some extra structure: namely, a poset. We want this comparison to always be possible, hence we will require the axioms of a *linear order*. Moreover, we require this order to be well-behaved with respect to the rig structure, hence we require that S forms an *ordered rig*.

Definition 1.20 (Linear order)**.** *A* linear order *is a poset $(S, \leq)$ such that for all $a, b \in S$ we have*

- $a \leq b \;\vee\; b \leq a$ *(linearity)*

Note that when $|S| = n$, a linear order on S is equivalent to an indexing $i : \underline{n} \to S$. Computing i from $\leq$ is known as sorting*.*

Definition 1.21 (Ordered rig)**.** *An* ordered rig *is a tuple $(S, \leq, \times, 1, +, 0)$ where $(S, \times, 1, +, 0)$ is a rig and $(S, \leq)$ is a linear order such that for all $a, b, c \in S$*

- $a \leq b \implies a + c \leq b + c$ *(+-compatibility)*
- $0 \leq a \ \wedge \ 0 \leq b \implies 0 \leq a \times b$ *(×-compatibility)*

In the examples that follow, the hyper-parameter n is taken to be the cardinality of a set of *context words*: some subset $\{ w_0, \ldots, w_n \} \subseteq V$ that we consider as relevant for computing word similarity. Usually this involves ignoring so-called *stop words* — i.e. words that appear too often to provide meaningful information such as "the", "of", etc. — as well as words that do not appear enough (e.g. "anticonstitutionally").

Example 1.4 ($S = \mathbb{N}$)**.** *The most basic method for computing the encoding matrix* $\mathtt{E} : |V| \to n$ *is counting:* $\mathtt{E}(Alice, w_i)$ *is the number of times "Alice" appears in the same sentence as the context word* w_i*, i.e.*

$$\mathtt{E}_{\mathbb{N}}(w_i, w_j) \ = \ \sum_{d \in C} \sum_{s \in d} \mathbf{1}_s(w_i) \times \mathbf{1}_s(w_j)$$

where $\mathbf{1}_s : V \to \mathbb{B}$ *is the indicator function for membership in the set of words in* s*. Note that we have made the arbitrary choice of considering appearance in the same sentence: we could compute a more coarse-grained meaning by taking discourse instead, or a finer-grained one by considering so-called* context windows*, i.e. whether the two words appear closer than some distance* k *of each other (which becomes another hyper-parameter of the model).*

Example 1.5 ($S = \mathbb{B}$)**.** *An example which is of perhaps less importance for semantics, but which we give nonetheless in order to illustrate the use of the Boolean rig, is to compute not* how many times *words appear together but only* whether *they appear together:*

$$\mathtt{E}_{\mathbb{B}}(w_i, w_j) \ = \ \begin{cases} 0 & \texttt{if } E_{\mathbb{N}}(w_i, w_j) = 0 \\ 1 & \texttt{otherwise} \end{cases}$$

Then $\langle w_i | \, \mathtt{E}_{\mathbb{B}}^T \mathtt{E}_{\mathbb{B}} \, | w_j \rangle = 1$ *iff* w_i *and* w_j *appeared next to the same context word.*

Example 1.6 ($S = \mathbb{R}$)**.** *Counting is usually not sufficient to give a satisfactory measure of similarity, hence count statistics are usually scaled so that less frequent words are given more weight: the fact that a word appears next to "house" gives less information than appearing next to "troglodyte" would. One traditional weighting scheme in information retrieval and text mining is* tf-idf *(for* term frequency - inverse document frequency*) of which there are numerous variations. Here we give the most basic version:*

$$\mathtt{E}_{\mathbb{Q}}(w_i, w_j) \ = \ \mathtt{E}_{\mathbb{N}}(w_i, w_j) \log\left(\frac{|C|}{\sum_{d \in C} \mathbf{1}_d(w_j)} \right)$$

We are now in a position to define the motivation for this thesis: given an encoding matrix $\mathtt{E} : |V| \to n$, we want to compute the semantics for the utterances $d \in V^*$ with $T^*(d) \leq s \times \cdots \times s$. Our first step is to remove the need for the star, as well as that for the dots: we do this by defining exponentials.

Definition 1.22 (Exponential)**.** *Given a monoid* $(X, \times, 1)$*, the* exponential *is an* $X \times \mathbb{N} \to X$ *function denoted* $(x, n) \mapsto x^{(\times n)}$ *and defined by*

$$x^{(\times n)} \quad = \quad \begin{cases} 1 & \texttt{if } n = 0 \\ x \times x^{(\times(n-1))} & \texttt{otherwise} \end{cases}$$

When x is a natural number itself we denote the exponential by x^n, similarly when it is a set with $\times$ denoting the Cartesian product, see remark 1.1. Thus V^k is the set of utterances of length k and we can reformulate our problem by fixing its dimension: given a discourse $d \in V^k$ with $T^*(d) \leq s^{(\times t)}$ — i.e. t grammatical sentences with k words in total — we encode the computation for the meaning of d as the scalar

$$\mathcal{D}(d) \quad = \quad \mathtt{r}\ \mathtt{E}^{(\otimes k)}\, |d\rangle$$

where $|d\rangle : 1 \to |V|^k$ is the one-hot state for the utterance. We construct $\mathtt{r} : n^k \to 1$ based on the grammatical derivation of the utterance while $\mathtt{E}^{(\otimes k)} : |V|^k \to n^k$ denotes the exponential for the Kronecker product:

Definition 1.23 (Kronecker Product)**.** *The* Kronecker product $\mathtt{u} \otimes \mathtt{v} : 1 \to a \times b$ *is defined by*

$$(\mathtt{u} \otimes \mathtt{v}) \quad = \quad (\times)\ (\mathtt{u} \times \mathtt{v})\ (1_a \star 1_b)^{-1} \quad : \quad \underline{1} \times \underline{a \times b} \to S$$

on pair of states $\mathtt{u} : 1 \to a$*,* $\mathtt{v} : 1 \to b$*, and* $\mathtt{M} \otimes \mathtt{N} : a \times b \to c \times d$ *is defined by*

$$(\mathtt{M} \otimes \mathtt{N})\, |i, j\rangle \quad = \quad \mathtt{M}\, |i\rangle \otimes \mathtt{N}\, |j\rangle$$

on pairs of matrices $\mathtt{M} : a \to c$*,* $\mathtt{N} : b \to d$*.*

Lemma 1.2 ($Ar(\mathtt{Mat}_S), \otimes, 1$ is a monoid)**.**

Proof. Note that we have overloaded $\otimes$: the right hand side of the bottom line is defined by the top line, which prevents circularity. The bijection $\star : \underline{a} \times \underline{b} \to \underline{a \times b}$ of definition 1.18 together with the equivalence of the previous remark makes this is a well-defined function. Then the proof follows from the definition of the product of encodings and the fact that $\mathbb{N}$ is a rig:

$$\begin{aligned} (k + |Z| \times j) + |Z \times Y| \times i \quad &= \quad k + |Z| \times j + |Z| \times |Y| \times i \\ &= \quad k + |Z| \times (j + |Y| \times i) \end{aligned}$$

for $(x, y, z) \in X \times Y \times Z$, writing i for $i(x)$ and similarly for j and k. □

1.3 Functors of Rigid Monoidal Categories

We can now get into the core of the DisCoCat model: the bridge between syntax and semantics which was first introduced in [13] then given mathematical foundations in [14]. The construction this bridge begins with the observation that pregroups

and categories of matrices share a common structure: they are both examples of *rigid monoidal categories*.

In this section, we introduce the categorical machinery required to formalise the structure-preserving mapping from syntax to semantics which drives our model. As a technical side-remark for the mathematically-minded reader, note that the categories we consider in this thesis are all skeletal hence what we actually define are *strict monoidal* categories. We will not need to mention this fact any further in what follows, indeed thanks to MacLane's coherence theorem we know that every monoidal category is equivalent to a strict one, see [8, p. 165].

A monoidal category can be seen as a category where the set of objects and arrows have the structure of two monoids that interact in a well-behaved way. Then intuitively rigid categories are to monoidal categories what pregroups are to monoids: they are equipped with left and right adjoints satisfying axioms which generalise those of definition 1.1. In order to define monoidal categories, we first define the notion of *product* of two categories and of two functors.

Definition 1.24 (Product Categories)**.** *Given a pair of categories $\mathcal{C}$ and $\mathcal{D}$, their* product $\mathcal{C} \times \mathcal{D}$ *is a category defined by*

- $Ob(\mathcal{C} \times \mathcal{D}) = Ob(\mathcal{C}) \times Ob(\mathcal{D})$
- $Ar(\mathcal{C} \times \mathcal{D}) = Ar(\mathcal{C}) \times Ar(\mathcal{D})$
- $1_{(c,d)} = (1_c, 1_d)$ *for* $c \in Ob(\mathcal{C})$ *and* $d \in Ob(\mathcal{D})$
- $(g_\mathcal{C}, g_\mathcal{D})(f_\mathcal{C}, f_\mathcal{D}) = (g_\mathcal{C} f_\mathcal{C}, g_\mathcal{D} f_\mathcal{D})$ *for* $f_\mathcal{C}, g_\mathcal{C} \in Ar(\mathcal{C})$ *and* $f_\mathcal{D}, g_\mathcal{D} \in Ar(\mathcal{D})$*, whenever the compositions of the right-hand side are well-defined.*

Associativity and unitality of composition in $\mathcal{C} \times \mathcal{D}$ follow from that of $\mathcal{C}$ and $\mathcal{D}$.

Definition 1.25 (Product Functors)**.** *Given a pair of functors $F : \mathcal{A} \to \mathcal{B}$ and $G : \mathcal{C} \to \mathcal{D}$, their* product *is a functor $F \times G : \mathcal{A} \times \mathcal{C} \to \mathcal{B} \times \mathcal{D}$ defined by*

- $(F \times G)_{Ob} = F_{Ob} \times G_{Ob}$
- $(F \times G)_{Ar} = F_{Ar} \times G_{Ar}$

where in the righ-hand sides the operation $\times$ is Cartesian product of functions. That $F \times G$ is indeed a functor follows from the fact F and G are functors.

Definition 1.26 (Monoidal Category)**.** *A* monoidal category *is a tuple $(\mathcal{C}, \otimes, I)$ where $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is called the* monoidal tensor *and $I \in Ob(\mathcal{C})$ is called the* monoidal unit*, such that*

- $\otimes(1_\mathcal{C} \times I) = 1_\mathcal{C} = \otimes(I \times 1_\mathcal{C})$ *(unitality)*
- $\otimes(\otimes \times 1_\mathcal{C}) = \otimes(1_\mathcal{C} \times \otimes)$ *(associativity)*

This formulation, which we borrow from [8, p. 162], allows to define at once the two monoids of objects and arrows, as well as their interaction.

Pregroups are monoidal categories where the objects are the elements and the arrows are the grammatical reductions. The object part of the tensor is the pregroup multiplication, the arrow part follows from the pomonoid axiom.

Definition 1.27 (Rigid Category)**.** *A monoidal category $\mathcal{C}$ is* rigid *if there exists an $Ob(\mathcal{C}) \times \{ l, r \} \to Ob(\mathcal{C})$ function denoted $(a, x) \mapsto a^x$, and a pair of functions $cup, cap : Ob(\mathcal{C}) \times \{ l, r \} \to Ar(\mathcal{C})$ such that*

$$cup(a, l) : a^l \otimes a \to I \quad cup(a, r) : a \otimes a^r \to I$$

$$cap(a, l) : I \to a \otimes a^l \quad cap(a, r) : I \to a^r \otimes a$$

for all $a \in Ob(\mathcal{C})$. We call a^l and a^r the left and right duals of a and require the following four axioms

$$(1_a \otimes cup(a, l))(cap(a, l) \otimes 1_a) = 1_a = (cup(a, r) \otimes 1_a)(1_a \otimes cap(a, r))$$

$$1_{a^l} = (cup(a, l) \otimes 1_{a^l})(1_{a^l} \otimes cap(a, l)) \qquad (1_{a^r} \otimes cup(a, r))(cap(a, r) \otimes 1_{a^r}) = 1_{a^r}$$

Pregroups are special cases of rigid categories: indeed the four functions required in the definition correspond precisely to the axioms defining a pregroup, and the four equations are trivially satisfied because pregroups are posets.

We now want to look at $\mathtt{Mat}_S$ as a rigid category, however in order to recover the original formulation of DisCoCat as a functor to a *compact closed* category, we need to add an extra axiom on our set of scalars. For any *commutative* rig S — i.e. $s \times t = t \times s$ for all $s, t \in S$ — $\mathtt{Mat}_S$ is a rigid category with the extra property that for all $a \in Ob(\mathtt{Mat}_S) = \mathbb{N}$ we have $a^l = a^r = a$.

Definition 1.28 (Maximally-Entangled Vectors)**.** *Given a dimension $a \in \mathbb{N}$, the* maximally-entangled vectors $\mathtt{cup}_a : a \otimes a \to 1$ *and* $\mathtt{cap}_a : 1 \to a \otimes a$ *are given by*

$$\mathtt{cup}_a \, |i, j\rangle = \delta_{\underline{a}}(i, j) \qquad \mathtt{cap}_a = \mathtt{cup}_a^T$$

Lemma 1.3 ($(\mathtt{Mat}_S, \otimes, 1)$ is rigid)**.**

Proof. Take $cup(a, l) = cup(a, r) = \mathtt{cup}_a$ and $cap(a, l) = cap(a, r) = \mathtt{cap}_a$. The functoriality axiom on objects follows from the fact that $(\mathbb{N}, \times, 1)$ is a monoid, that on arrows can be spelled out as the *interchange rule*:

$$(\mathtt{M}_1 \mathtt{M}_0) \otimes (\mathtt{N}_1 \mathtt{N}_0) = (\mathtt{M}_1 \otimes \mathtt{N}_1)(\mathtt{M}_0 \otimes \mathtt{N}_0)$$

Both this statement and the axioms for rigidity can be proved directly by extensionality, i.e. proving that the matrices have equal entries. However the proof would be rather cumbersome and uninformative so we will allow ourselves to appeal to general abstract nonsense: $\mathtt{Mat}_S$ is equivalent to the category of *free finite-dimensional S-semimodules*. Thus it is compact closed — which implies rigidity, see [15]. □

We can now define our functor from syntax to semantics: the object part sends every grammatical type to some natural number, the arrow part sends every grammatical reduction to a matrix. Moreover we require that it behaves well with respect to the rigidity of the two categories: it is a *rigid functor*.

Definition 1.29 (Monoidal Functor)**.** *A* monoidal functor *between two monoidal categories* $(\mathcal{C}, \otimes, I)$ *and* $(\mathcal{D}, \boxtimes, J)$ *is a functor* $F : \mathcal{C} \to \mathcal{D}$ *such that*

$$FI = J \ : \ 1 \to \mathcal{D} \qquad F\otimes = \boxtimes(F \times F) \ : \ \mathcal{C} \times \mathcal{C} \to \mathcal{D}$$

As for the definition of monoidal category, this formulation allows to give the axioms for both the object and arrow part of F *at once.*

Definition 1.30 (Rigid Functor)**.** *A monoidal functor* $F : \mathcal{C} \to \mathcal{D}$ *is* rigid *if the two categories are rigid and furthermore for* $x \in \{ l, r \}$*:*

$$F(-)^x = (-)^x F \qquad Fcup(x) = cup(x)F \qquad Fcap(x) = cap(x)F$$

where we have overloaded the object and arrow functions of F *and curried the functions* $cup, cap : Ob(\mathcal{X}) \times \{ l, r \} \to Ar(\mathcal{X}) \simeq \{ l, r \} \to (Ob(\mathcal{X}) \to Ar(\mathcal{X}))$ *for* $\mathcal{X} \in \{ \mathcal{C}, \mathcal{D} \}$*. Note that this differs from the standard definition, which only requires the existence of adjoints, not explicit functions.*

Let us begin by defining the image $F_B : B \to \mathtt{Mat}_S$ of our semantic functor F on the poset of basic types B as defined in 1.1:

$$F(s) = 1 \qquad F(n) = F(sub) = F(obj) = n$$

where n is overloaded and stands for both the noun type and the hyper-parameter of the model as described in section 1.2. Defining F_B on objects uniquely defines the object function $F_{Ob} : G \to \mathbb{N}$ of the semantic functor, indeed the images on the other types follows from monoidality and the freeness of the pregroup. With this definition on objects, we can define the arrow function of $F_B : B \to \mathtt{Mat}_S$ as

$$F(n \leq obj) = F(n \leq sub) = 1_n \ : \ n \to n$$

Then image on the reductions given by the pomonoid axioms follow from the axioms of monoidal functors, it now remains to define the image on the reductions defined by the pregroup axioms. We would like to simply assert

$$F(a^l \times a \leq 1) = F(a \times a^r \leq 1) = \mathtt{cup}_{F(a)} \ : \ F(a) \otimes F(a) \to 1$$
$$F(1 \leq a \times a^l) = F(1 \leq a^r \times a) = \mathtt{cap}_{F(a)} \ : \ 1 \to F(a) \otimes F(a)$$

but this becomes problematic in the case when the same utterance has two non-isomorphic derivations. Indeed whereas pregroups are posets and therefore have at most one arrow between any two types, this definition would map the same arrow in G to two distinct arrows in $\mathtt{Mat}_S$: it cannot define a functor.

Example 1.7. *The utterance "Bob hates songs and movies that Alice loves" is a grammatical sentence in two ways. The full derivations would fill up the page, but intuitively they correspond to the two bracketings "Bob hates (songs and movies) that Alice loves" and "Bob hates songs and (movies that Alice loves)", see [16].*

In order to define the desired functor, we need to enrich our structure for syntax so that it takes into account not only *whether* a sentence is grammatical but *how* it reduces to the sentence type. We do this by extending the notion of free pregroup to that of *free rigid category* introduced by Preller and Lambek as special cases of *free compact 2-categories*.

The free rigid category generated by a poset B has the same objects as that of the free pregroup, and its arrows intuitively correspond to pregroup derivations with concatenation of proofs as composition. As for freeness of pregroups, a formal definition of free rigidity would lead us astray: we only need to know that a rigid functor from such a free category is uniquely defined by its image on the poset B.

We conclude this section by defining what we call *rigid grammars*: intuitively they are to free rigid categories what pregroup grammars are to pregroups. Finally, we define an S-valued *semantic functor* as a rigid functor from some rigid grammar to the category $\mathtt{Mat}_S$.

Definition 1.31 (Free Rigid Grammar)**.** *Given a pregroup grammar (G, s, V, T) freely generated by a poset B, we define the* rigid grammar *$(\mathcal{G}, \otimes, I)$ as the free rigid category generated by B and* word arrows *$w : I \to T(w)$ for $w \in V$.*

We overload the notation and also write $u : I \to T^*(u)$ for the *utterance arrow* generated by $u = \langle w_0, \ldots, w_n \rangle$. By construction, an utterance $u \in V^*$ is a grammatical sentence when there exists a reduction arrow $r : T^*(u) \to s$, which holds if and only if $T^*(u) \leq s$ in G.

Definition 1.32 (Semantic Functor)**.** *Given a rigid grammar $\mathcal{G}$ generated by (G, s, V, T), a* semantic functor *is a rigid functor $F : \mathcal{G} \to \mathtt{Mat}_S$ for some rig S. Given a reduction $r : a \to b$, the semantics of an utterance $u \in V^*$ with $T^*(u) = a$ is given by $F(ru) : 1 \to b$.*

From functoriality, we have that the semantics of $u = w_0 \otimes \cdots \otimes w_k : 1 \to a$ under $r : a \to b$ is equal to the composition of the images. From monoidality, we get that the image of a semantic functor on utterances is the Kronecker product of the images on its constituent words. Finally, the image on the reduction arrows follows from rigidity and the fact that $\mathcal{G}$ is free, thus we recover the formulation which concluded the previous section:

$$\begin{aligned} F(ru) &= F(r)\ F(u) \\ &= F(r)\ (\mathtt{E}\,|w_0\rangle \otimes \cdots \otimes \mathtt{E}\,|w_k\rangle) \\ &= F(r)\ \mathtt{E}^{(\otimes k)}\ (|w_0\rangle \otimes \cdots \otimes |w_k\rangle) \\ &= F(r)\ \mathtt{E}^{(\otimes k)}\ |u\rangle \end{aligned}$$

The last line follows from k applications of the definition for one-hot vectors and that for Kronecker product, the one before last follows from k applications of the interchange rule of lemma 1.3. However, in passing from the first to the second line we made the assumption that there is a single encoding matrix $\mathtt{E} : |V| \to n$, e.g. we have $\mathtt{E}\,|loves\rangle = F(loves) : 1 \to F(n^r \times s \times n^l)$ which by monoidality implies that $n = F(n^r) \times F(s) \times F(n^l) = n \times n$ — i.e. $n \in \{\, 0, 1 \,\}$ our model is trivial!

1.4 Alice Loves Bob and Biproducts

This leads to a crucial observation which distinguishes DisCoCat from previous distributional models: the word vectors for nouns and for transitive verbs cannot have the same dimension anymore: we need to define distinct encoding matrices for every grammatical type. With our definition of $F : \mathcal{G} \to \mathtt{Mat}_S$ for the basic example of grammar described in section 1.1, the meaning of a sentence is given by a scalar in S which we will interpret as a *generalised truth value*, the meaning of a noun is given by a state $1 \to n$, that of a transitive verb by a state $1 \to n \otimes n$.

Example 1.8. *Take $S = \mathbb{B}$ and $n = 2$, we partition our vocabulary $V = E \cup R$ into a set of nouns $E = \{ Alice, Bob \}$ and a set of verbs $R = \{ loves \}$. We define two encoding matrices $\mathtt{E} : |E| \to n$ and $\mathtt{R} : |R| \to n \otimes n$ as follows.*

$$\mathtt{E}\,|Alice\rangle = (01) \qquad \mathtt{E}\,|Bob\rangle = (10) \qquad \mathtt{R}\,|loves\rangle = (0101)$$

Then we compute the meaning of the sentence "Alice loves Bob" under the reduction arrow $r : n \otimes sub^r \otimes s \otimes obj^l \otimes n \to s$ corresponding to example 1.2 as the scalar

$$\begin{aligned} F(r)F(Alice \otimes loves \otimes Bob) &= (\mathtt{cup}_2 \otimes \mathtt{cup}_2)(\mathtt{E}\,|Alice\rangle \otimes \mathtt{R}\,|loves\rangle \otimes \mathtt{E}\,|Bob\rangle) \\ &= (\mathtt{E}\,|Alice\rangle \otimes \mathtt{E}\,|Bob\rangle)^T \mathtt{R}\,|loves\rangle \\ &= ((01) \otimes (10))^T (0101) = (0100)^T (0101) = 1 \end{aligned}$$

In the end, we can say that Alice really does love Bob! Note that in order to get from the first to the second line, we have used some magic which we introduce in the next chapter: string diagrams.

One way of getting around the issue mentioned above would be to make the assumption that the set of grammatical types for the words is finite, which is a more reasonable hypothesis than that discussed in section 1.2. Then we can make use of a construction which we have not yet introduced in this thesis: biproducts.

Definition 1.33 (Biproduct)**.** *Given two matrices $\mathtt{M} : a \to c$ and $\mathtt{N} : b \to d$, their biproduct $\mathtt{M} \oplus \mathtt{N} : a + b \to c + d$ is given by*

$$\mathtt{M} \oplus \mathtt{N} \;=\; (\mathtt{M} \diamond \mathtt{N}) \;\; ((1_a + 1_b)^{-1} \times (1_c + 1_d)^{-1}) \;\; : \;\; \underline{a + b} \times \underline{c + d} \to S$$

where $(+)$ is the sum of indexing, and the function $\mathtt{M} \diamond \mathtt{N} : \underline{a} + \underline{b} \to \underline{c} + \underline{d}$ is given by:

$$(\mathtt{M} \diamond \mathtt{N})((i, x), (j, y)) \;=\; \begin{cases} \mathtt{M}(i, j) & \mathtt{if} \;\; x = y = l \\ \mathtt{N}(i, j) & \mathtt{if} \;\; x = y = r \\ 0 & \mathtt{otherwise} \end{cases}$$

Thus we can update our model, i.e. take $n \mapsto n + n \times n$ and $E \mapsto E \oplus R$ so that:

$$F(ru) = F(r) \;\; \mathtt{E}^{(\otimes k)} \;\; |\tau^*(u)\rangle \qquad \tau(w) = \begin{cases} (w, l) & \mathtt{if} \;\; w \in E \\ (w, r) & \mathtt{if} \;\; w \in R \end{cases}$$

This reformulation comes at the cost of: 1) a quadratic increase in the size of our encoding matrix, 2) a loss of the separation of concerns between syntax and semantics. In the next chapter, we attempt to justify this cost.

LE PETIT PRINCE: *"Even flowers that have thorns?"*

ANTOINE: *"Yes, even flowers that have thorns."*

LE PETIT PRINCE: *"So, what's the use of thorns?"*

I didn't know. At that point I was very busy trying to unscrew a bolt in my engine that was too tight. I was very worried because the breakdown started to look very serious, and the drinking water running low made me expect the worse.

LE PETIT PRINCE: *"So, what's the use of thorns?"*

The little prince never let go of a question, once he had asked it.

2

Diagrams of Questions and Answers

In this chapter, our working hypothesis is that something similar to the dialogue:

BOB: Does Alice love Charlie?
DAN: Alice does love him.

has happened in every natural language. Hence we take this question and its answer as a universal example, in the sense discussed at the end of section 1.1. In the late eighties, Hamblin [17] formalised natural language questions in the framework of Montague semantics: he defined a homomorphism which translates questions into programs that compute the set of the answers. We will show how adapting this approach to the DisCoCat model leads to the idea that the semantics for an answer should be in some intuitive sense the *dual*[1] of its question.

We define the fragment of natural language which will be our object of study for the rest of this thesis: the discourses built from *basic sentences* of the form "subject verb object". We then show how a corpus in this fragment, once given a semantics in the DisCoCat framework, can be translated into machine language in a structure-preserving way: it forms a *knowledge graph* — also called an *ontology*.

[1] This is a footnote to Plato, whose dialogues have a similar structure to *game semantics* — a framework which could be used to formalise the logic behind our models of meaning: multiplicative linear logic [18]. To make a political analogy, one player tries to make a claim while the other answers: "fake news!" Then the claim is true if and only if the first player has a winning strategy, i.e. he answers every question his opponent asks and wins the debate.

The idea that truth is a process has notably been formalised in Hegel's *Science of Logic*, which has long been accused by scientists and logicians of being neither scientific nor logical. With his Hegelian taco [19], Lawvere offers some food (a monoid) for thoughts on how modern mathematics may end up going back to Hegel — in a way, he might be considered as the founding father of "general abstract nonsense" in philosophy.

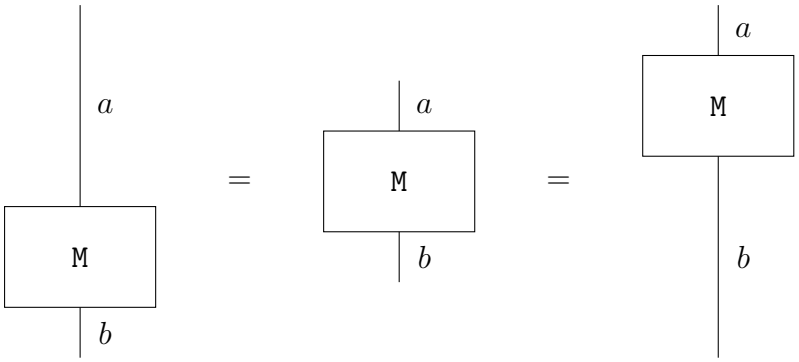
Borrowing the intuition from Wittgenstein's *Tractacus*, dialogues of questions and answers between a master and a student can be seen as "ladders", the student "eventualy recognises them as nonsensical, when he has used them — as steps — to climb beyond them. (He must, so to speak throw the ladder after he has climbed up it)" [10, p. 6.54].

2.1 String Diagrams for Natural Language

One of the strengths of DisCoCat models for natural language is that they allow us to reason about *information flow* through a graphical language known as *string diagrams*. In this section we introduce equations of diagrams as a tool to reason about compact closed categories, we refer the reader to [20] for a survey of diagrammatic reasoning for monoidal categories in general.

Before formalising the notion of duality, we start by spelling out the main intuition behind how we use diagrams to reason about natural language. The graphical language allows to literally *see* how matrices can encode the computation for the meaning of a discourse: they *transform* and connect together the meanings of the individual words. The algebraic formulations of chapter 1 allowed us to express semantic computations in a generic and concise way. On the other hand writing string diagrams takes more space, however it is well-suited to the study of particular examples.

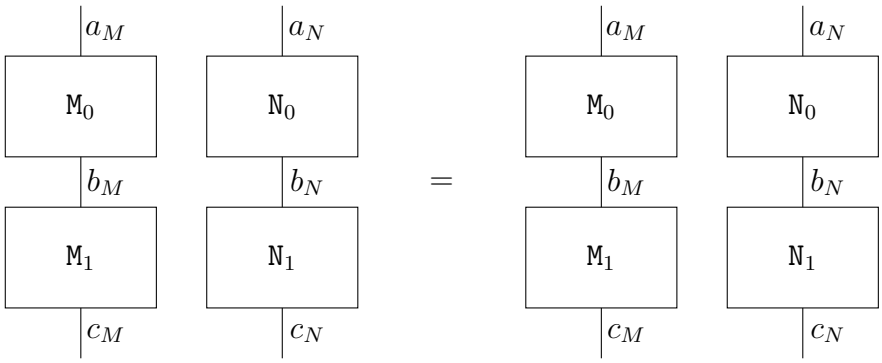
We represent a matrix $\mathtt{M} : a \to b$ as a box with an input wire labeled with a on top, and an output wire labeled with b at the bottom. This is the so-called *pessimistic* convention where information flows downwards, i.e. *future is hell*. The main intuition behind the graphical notation is that of *matrices as processes*: a-dimensional information flows into the matrix $\mathtt{M}$ which transforms it into b-dimensional information. The identity matrices leave this information untouched, thus we represent them as plain wires instead of boxes. From this intuitive notation, the unit axiom of categories becomes a topological move: we are allowed to slide boxes up and down wires. Indeed, the equalities $\mathtt{M}\mathtt{1}_a = \mathtt{M} = \mathtt{1}_b\mathtt{M}$ become graphically



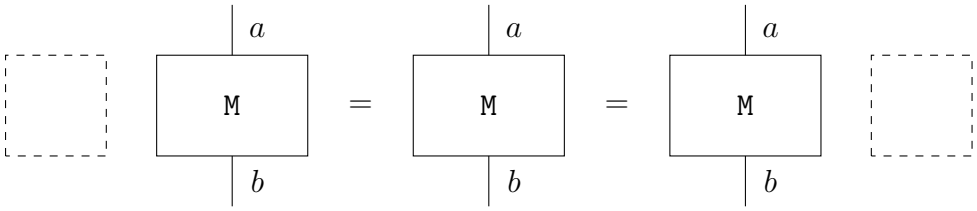
where composition of matrices $\mathtt{M}_0 : a \to b, \mathtt{M}_1 : b \to c$ is depicted by the vertical concatenation of their boxes, connecting the output wire of the top box with the input of the bottom one. This notation allows to get the associativity axiom of categories for free: we never need to draw parenthesis. This was was already true for the associativity of monoids written as one-dimensional equations — at least when the monoid multiplication was clear from context. Now whereas connecting wires of boxes vertically represents composition, horizontal juxtaposition of boxes will represent the Kronecker product.

The intuition is that composition of matrices encodes processes in sequence while the Kronecker product encodes processes in parallel. As for vertical concatenation, horizontal juxtaposition gives us the associativity axiom of monoidal categories

for free. This is perhaps key advantage of two-dimensional diagrams over their corresponding one-dimensional bracketed expressions, as can be seen with the interchange rule spelled out in lemma 1.3. Spotting the applications of this rule inside complex expressions would require some heavy mental gymnastics, however once we represent it graphically as the following equation of diagrams:



... it becomes trivial! The final piece of notation we will need for reasoning about $\mathtt{Mat}_S$ as a monoidal category is that for its unit 1. Intuitively whereas we represent the identity matrix $\mathtt{1}_a$ as a labeled wire carrying a-dimensional information, the identity matrix $\mathtt{1}_1 : 1 \to 1$ cannot carry any information: we do not bother drawing it at all or rather, we draw it as the empty diagram. Again, this notation gives us the unit axiom for free: juxtaposition with the empty diagram does nothing. It may also be seen as allowing us to leave some space on the left and right of a diagram.



We will distinguish states and effects from other matrices by drawing them as triangles — with no input and no output respectively — instead of boxes. This is consistent with the representation of $\mathtt{1}_1 : 1 \to 1$ as an empty diagram: concatenating it on top of a state or below an effect does nothing. Finally, composing a state with an effect of the same dimension yields a diagram with no input or output: a scalar in S.

We conclude this section by defining the mathematical devices we will use to reason about duality, from *opposite categories* to *symmetric dagger structures*: intuitively, we flip all the arrows upside-down. Note that in this chapter we will also forget about left and right, that is we ask for the multiplication in rig of scalars S to be *commutative*: for all $s, t \in S$

$$s \times t \;=\; t \times s$$

This extra assumption has the following linguistic consequence: scalar product becomes commutative which makes word similarity a symmetric relation — e.g. "France" is as similar to "Belgium" as "Belgium" is to "France" — which violates experimental results in psychology, see [21]. Non-commutative cases and their applications to cognition have been investigated in some previous work, see [4].

Definition 2.1 (Opposite Category)**.** *The* opposite *of the graph* $G = (O, A, dom, cod)$ *is* $G^\dagger = (O, A, cod, dom)$*. The opposite of* $\mathcal{C} = (G, \circ, id)$ *is* $\mathcal{C}^\dagger = (G^\dagger, \circ\ \sigma_{A\times A}, id)$*.*

Note that the *opposite functor* $F : \mathcal{C}^\dagger \to \mathcal{D}^\dagger$ is the same as $F : \mathcal{C} \to \mathcal{D}$ — i.e. they have the same object and arrow functions — and $\mathcal{C}^{\dagger\dagger}$ is the same as $\mathcal{C}$, which follows from the fact that $\sigma_{A\times A} : A \times A \to A \times A$ is an involution: $(\sigma_{A\times A})(\sigma_{A\times A}) = 1_{A\times A}$. Currying our notation, if

$$\mathcal{G} \;=\; (\mathcal{C}, cup(l), cap(l), cup(r), cap(r))$$

is rigid then its opposite

$$\mathcal{G}^\dagger = (\mathcal{C}^\dagger, cap(r), cup(r), cap(l), cup(l))$$

is rigid as well. However we avoid mentioning $\mathcal{C}^\dagger$ directly as it is really the same as $\mathcal{C}$ — opposite categories are only a device we use to define *daggers*.

Definition 2.2 (Dagger Structure)**.** *A* dagger structure *is a functor* $\dagger : \mathcal{C} \to \mathcal{C}^\dagger$ *such that*

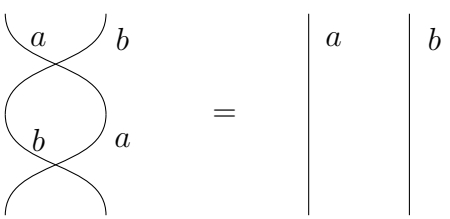
$$\dagger \;=\; 1_{Ob(\mathcal{C})} \;:\; Ob(\mathcal{C}) \to Ob(\mathcal{C}) \qquad \dagger\dagger \;=\; 1_{Ar(\mathcal{C})} \;:\; Ar(\mathcal{C}) \to Ar(\mathcal{C})$$

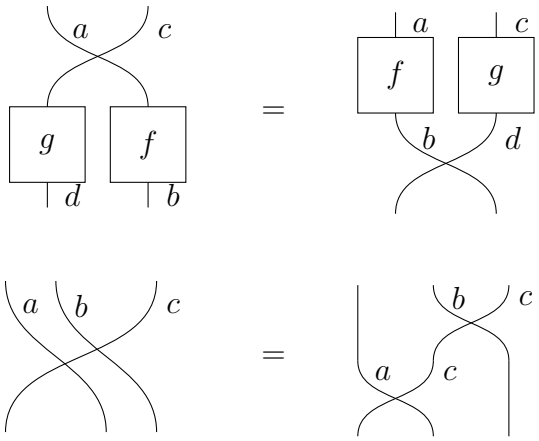
in short, an involutive identity-on-object contravariant endofunctor*. When applied to categories with a single object, a dagger structure is called an* involutive monoid*.*

Definition 2.3 (Symmetric Category)**.** *A monoidal category* $(\mathcal{C}, \otimes, 1)$ *is* symmetric *if there is a function* σ *assigning a* swap *arrow* $\sigma(a, b) : a \otimes b \to b \otimes a$ *to each pair* $(a, b) \in Ob(\mathcal{C}) \times Ob(\mathcal{C})$ *such that for all* $f : a \to b$ *and* $g : c \to d$

- $\sigma(1, a) \;=\; 1_a \;=\; \sigma(a, 1)$
- $\sigma(b, a)\ \sigma(a, b) \;=\; 1_a \otimes 1_b$
- $\sigma_{b,d}(f \otimes g) = (g \otimes f)\sigma_{a,c}$
- $\sigma(a \times b, c) \;=\; (\sigma(a, c) \otimes 1_b)\ (1_a \otimes \sigma(b, c))$

We depict the swap of a and b by the crossing of the wire depicting their identity so that we can translate these axioms as equations of string diagrams:





Thus we can interpret the four axioms for symmetry as encoding our graphical intuition for *bending* and *crossing* of wires. The first axiom — crossing with the empty diagram does nothing — allows us to bend an identity wire as long as it does not disconnect. The second axiom allows us to uncross two wires while the third is called *naturality*. Applying it to the case when f or g are the identity allows us to slide boxes through wires, in the same way as the axiom for identity allowed us to slide it up and down. Finally, the last axiom called the *pentagon* allows to decompose the crossing of a pair of wires into a pair of crossings, one wire at a time.

Definition 2.4 (Symmetric Dagger)**.** *A dagger structure* $\dagger : \mathcal{C} \to \mathcal{C}^{\dagger}$ *is* symmetric *if* $\mathcal{C}$ *is symmetric,* $\dagger$ *is monoidal and* $\dagger(\sigma(a,b)) = \sigma(b,a)$ *for all* $a, b \in Ob(\mathcal{C})$*.*

Definition 2.5 (Inner Product)**.** *When* $(S, \dagger)$ *is a commutative involutive rig, the* conjugate transpose *of the* $a \times b$ *matrix* $\mathtt{M}$ *is given by*

$$\mathtt{M}^{\dagger} \;=\; \dagger\ \mathtt{M}^{T} \;:\; \underline{b} \times \underline{a} \to S$$

The inner product *of two states* $\mathtt{u}, \mathtt{v} : 1 \to a$ *is the scalar* $\mathtt{v}^{\dagger}\mathtt{u} \in S$*.*

Lemma 2.1. *The conjugate transpose* $\dagger : \mathtt{Mat}_S \to \mathtt{Mat}_S^{\dagger}$ *defines a symmetric dagger with the swap arrow* $\sigma(a,b) : a \times b \to b \times a$ *given by the matrix:*

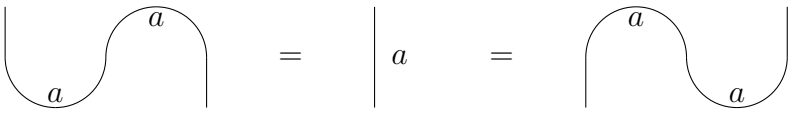
$$\sigma(a,b) \;=\; (\star)\ \sigma_{\underline{a}\times\underline{b}}\ (\star)^{-1}$$

where σ *in the right-hand side denotes the swap function of definition 1.15 and from right to left* $(\star)$ *denotes first* $\underline{a} \times \underline{b} \to \underline{a \times b}$ *then* $\underline{b} \times \underline{a} \to \underline{b \times a}$ *from definition 1.18.*

Proof. See [15] where this result is used to build *toy quantum theories*. □

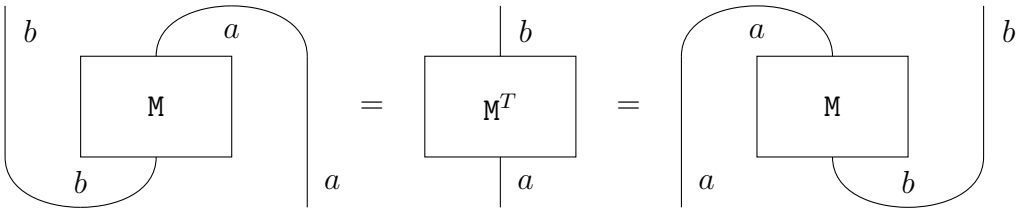
Rigid symmetric monoidal categories are called *compact closed*, rigid symmetric dagger structures are simply called *compact dagger*. Now in order to reason about matrices as a compact closed category, we introduce an extra piece of notation for the maximally-entangled states and effects which justifies their name. Indeed, while the intuition behind the use of vectors is that they serve as information-storing devices, we can interpret the special case of $\mathtt{cup}_a$ and $\mathtt{cap}_{\mathtt{a}}$ as information-passing

mechanisms. Thus we represent them as bent wires, the reason for this notation becomes clearer once we represent the axioms for rigidity — also known as the *snake equations* — graphically:

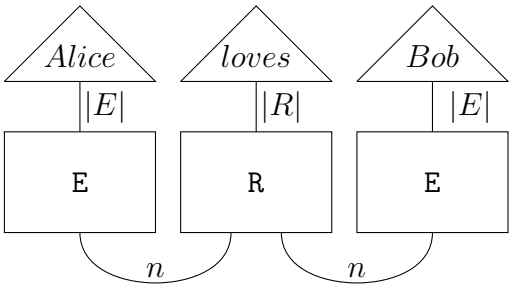


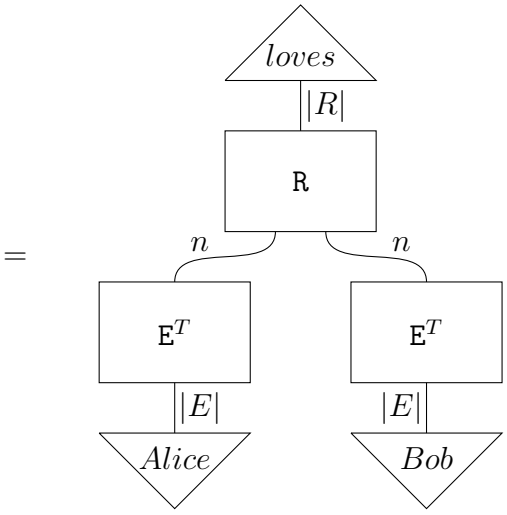
String diagrams are not only an intuitive notation: they can be treated as a formal language in a mathematical sense. Indeed, any equality which can be proved from the graphical notation — by bending and stretching wires, sliding boxes without disconnecting them — can also be proved directly from the axioms of compact closed categories. The converse is also true: any equation derived from the axioms of compact closed categories can be proved in the graphical language. A proof of these two statements — called *soundness* and *completeness* of the graphical language — would go well beyond the scope of this thesis, we refer the interested reader to [20].

The standard concepts of linear algebra can be formulated in terms of this graphical language, see Coecke and Kissinger [22] for an introduction to diagrammatic reasoning in the context quantum computing. As an example, we can redefine graphically the transpose of a matrix $\mathtt{M} : a \to b$ as follows.



Note that the equation $\mathtt{M}^T = (\mathtt{1}_a \otimes \mathtt{cup}_b)\ (\mathtt{1}_a \otimes \mathtt{M} \otimes \mathtt{1}_b)\ (\mathtt{cap}_a \otimes \mathtt{1}_b)$ would be non-trivial without the help of monoidal category theory. The graphical language makes clear that taking the transpose encodes some intuitive notion of *reversing the process*. We are now in a position where we can justify the derivation we used in the example which concluded the previous chapter. The meaning of the sentence "Alice loves Bob" given the grammatical reduction described in example 1.2 is represented by the following diagram.

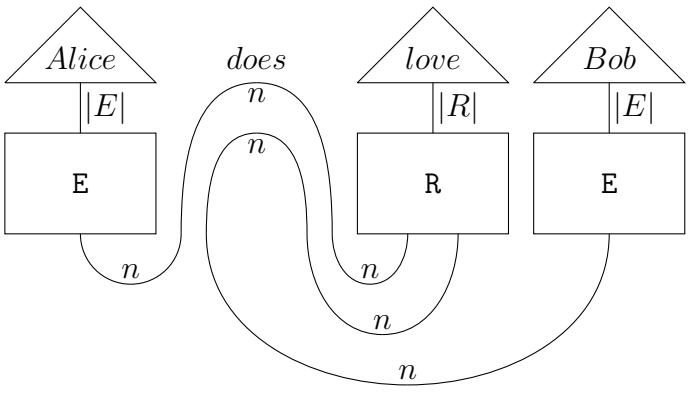




We have rotated the encoding matrices $\mathtt{E} : |E| \to n$ into their transpose $\mathtt{E}^T : n \to |E|$, the transposes of the one-hot states $|Alice\rangle, |Bob\rangle : 1 \to |E|$ being the corresponding effects $\langle Alice|, \langle Bob| : |E| \to 1$. This allows to see graphically how the maximally-entangled effect $\mathtt{cup}_n : n \otimes n \to 1$ feeds the information from the subject and the object into the verb. In essence, the scalar we compute with this diagram can be interpreted as the similarity between the meaning of "loves" and the pair of meanings for the two nouns "Alice" and "Bob", i.e. their scalar product.

In a more complex example we adapt from [23], we show how the maximally entangled state $\mathtt{cap}_n : 1 \to n \otimes n$ behaves in a similar way. It starts from the observation that the meaning of the sentence "Alice does love Bob" — in our setting where this is a scalar encoding a generalised notion of truth value — should be the same as that of "Alice loves Bob" even though the two sentences have a different syntax. The auxiliary "does" can be given the pregroup type $v_{tr} \times (v_{tr})^l$ where $v_{tr} = sub^r \times s \times obj^l$: intuitively it waits for a transitive verb on its right to yield a composite verb. The image of this type under our semantic functor is $F(v_{tr} \times (v_{tr})^l) = (n \otimes 1 \otimes n) \otimes (n \otimes 1 \otimes n) = n^4$, which is enforced by the axioms of a monoidal functor.

We choose to model the meaning of the word "does" — i.e. define the image of its word arrow under the semantic functor — as the state $\mathtt{cap}_n(\mathtt{1}_n \otimes \mathtt{cap}_n \otimes \mathtt{1}_n) : 1 \to n^4$. Then the meaning of "Alice does love Bob" is represented by the diagram



where we have composed the state $F^*\langle Alice, does, love, Bob\rangle : 1 \to n^8$ with the effect

$F(r) : n^8 \to 1$ — the image of the semantic functor on the following reduction arrow:

$$\begin{aligned} sub\ v_{tr}\ (v_{tr})^l\ v_{tr}\ obj &\to sub\ v_{tr}\ obj \\ &\to sub\ sub^r\ s\ obj^l\ obj \\ &\to 1\ s\ 1 \\ &\to s \end{aligned}$$

Note that we have made no other choice than the type for the word "does", the axioms which define our semantic functor as monoidal have enforced:

$$F(r) = \mathtt{cup}_n \otimes (1_n \otimes \mathtt{cup}_n \otimes 1_n)(\mathtt{cup}_n \otimes (1_n \otimes 1_n \otimes \mathtt{cup}_n \otimes 1_n \otimes 1_n))$$

which is easier to remember as a 4-line diagram than as a 1-dimensional formula.

Then our choice for the image of the word arrow for "does" becomes graphically intuitive: the state for "does" takes the meaning of "loves" on its right then feeds it into the meaning for the pair "Alice" and "Bob". This is consistent with the reverse intuition we gave above for the sentence "Alice loves Bob", indeed any scalar $s : 1 \to 1$ is trivially self-transpose — which is provable graphically by rotating it by a π.

2.2 Object and Subject Questions as Effects

While the dimension of word vectors and the matrices for the reduction follow freely from the axioms for rigidity, we are free to choose the image of our semantic functor on words as we will. Assigning special vectors to *structural words* — such as "does" in the previous section — has been an ongoing area of research since the first formulation of DisCoCat models.

In a series of work [24] [25] [26], Sadrzadeh et al. have applied this method to give a semantics to a class of such structural words: *relative pronouns* such as "that", "which" and "whom". Intuitively, a relative pronoun takes the information of some noun on its left — the *referent* — and updates it with some new information coming from an incomplete sentence on its right — the *relative clause*. This intuition has been modelled mathematically using the *generalised Kronecker delta*:

Definition 2.6 (Generalised Kronecker Delta)**.** *For any triple of dimensions $a, b, n \in \mathbb{N}$, the* Kronecker delta $\delta_n^{a,b} :\ n^a \to n^b$ *is given by*

$$\delta_n^{a,b} |w_0, \ldots, w_a\rangle \ = \ \begin{cases} |i\rangle^{(\otimes b)} & \texttt{if } \exists\ i < n\ \cdot \bigwedge_{j<a} \delta_{\underline{n}}(w_j, i) = 1 \\ \mathtt{0}^{(\otimes b)} & \texttt{otherwise} \end{cases}$$

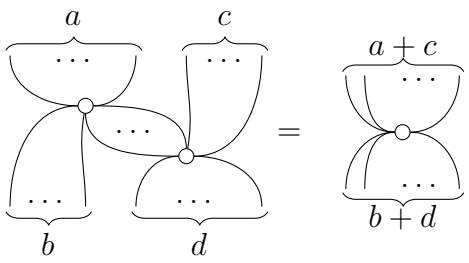
for $\langle w_0, \ldots, w_a\rangle \in \underline{n}^a$ where for any $m, n \in \mathtt{N}$ the zero matrix $\mathtt{0} : m \to n$ *denotes the constant $\mathtt{0}(i, j) = 0 \in S$ and $\bigwedge_{j<a}$ denotes the function*

$$\bigwedge_{j<a} b_j \ = \ \begin{cases} b_0 & \texttt{if } a = 1 \\ b_{(a-1)} \ \wedge \ \Big(\bigwedge_{j<(a-1)} b_j \Big) & \texttt{otherwise} \end{cases}$$

We will omit n when the context is clear.

This defines a basic algorithm for comparing a sequence of a one-hot vectors, and in the case when they are all equal to some $|i\rangle : 1 \to n$, output a sequence $|i\rangle^{(\otimes b)} : 1 \to n^b$ of b copies of it. This definition by cases — which is justified by the equivalence discussed in section 1.2 — puts emphasis on the computational steps required to compute the composition of $\delta_n^{a,b}$ with particular states. However, from this notation a basic symmetry of the generalised Kronecker delta is hidden: taking its mirror image and swapping the a for the b — i.e. $\delta_n^{a,b}\,|w_0,\ldots,w_a\rangle$ becomes $\langle w_0,\ldots,w_b|\,\delta_n^{a,b}$ and $\bigwedge_{j<a} b_j$ becomes $\bigwedge_{i<b} a_i$ — yields an equivalent definition. We will need the following graphical theorem in order to make sense of this symmetry.

Theorem 2.1 (Spider Fusion)**.** *For all $n \in \mathbb{N}$ and $(a,b,c,d) \in \mathbb{N}^4$, we have the following* spider fusion *rule where we denote $\delta_n^{a,b}$ as an a-input, b-output spider:*



Proof. $\left(\mathtt{id}_n^{(\otimes b)} \otimes \delta_n^{(k+c),d}\right)\left(\delta_n^{a,(b+k)} \otimes \mathtt{id}_n^{(\otimes c)}\right) = \delta_n^{(a+c),(b+d)}$, see [27]. Note how the string diagrams make the topological nature of the equation apparent while hiding the bureaucracy of indices and parenthesis. □

Definition 2.7 (Relative Clause)**.** *Given a free rigid grammar $\mathcal{G}$ generated by (G,V,s,T) and a designated pair of* object type *$o \in G$ and* noun type *$n \in G$ with $n \leq o$, we define an (object)* relative pronoun *as a word arrow*

$$that \;:\; 1 \to rel \;=\; n^l \times n \times (o^l)^l \times s^l$$

An (object) relative clause *is an utterance $u = \langle that, w_0, w_1, \ldots\rangle \in V^*$ with a reduction arrow $r : T^*(u) \to rel$. Symmetrically for subject relative clauses.*

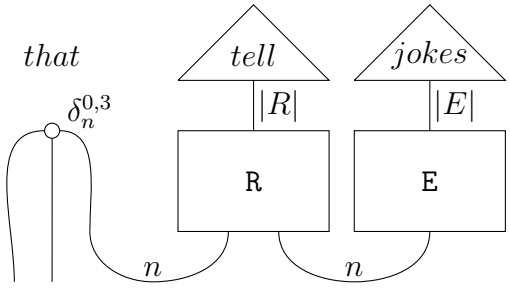
Example 2.1. *For $F : \mathcal{G} \to \mathtt{Mat}_S$ and some hyper-parameter $n \in \mathbb{N}$, we define:*

$$F(s) = 1 \qquad F(o) = F(n) = n \qquad F(n \leq o) = 1_n$$
$$F(that) = \delta^{0,3}_{F(n)} \;:\; 1 \to F(n) \otimes F(n) \otimes F(o)$$

We compute the semantics of the utterance $u = \langle that, tell, jokes\rangle$ with

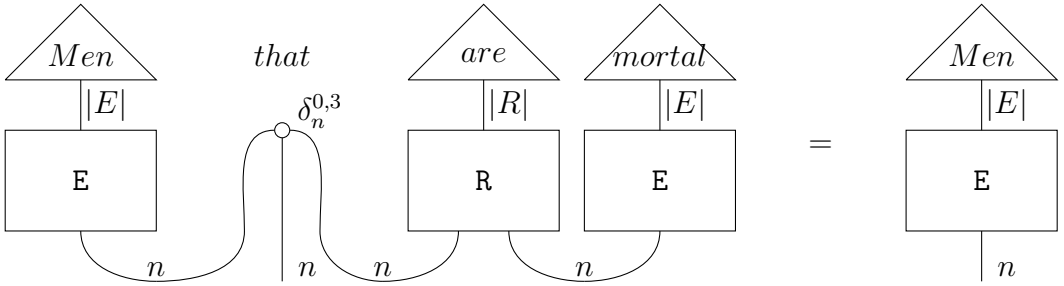
$$T(tell) = n^r \times s \times o^l \qquad T(jokes) = o$$
$$r = 1_{n^l} \otimes 1_n \otimes cup_n^r \otimes 1_s \otimes cup_o^l$$

then $F(ru) \;:\; 1 \to F(n) \otimes F(n)$ is given by the following scalar.

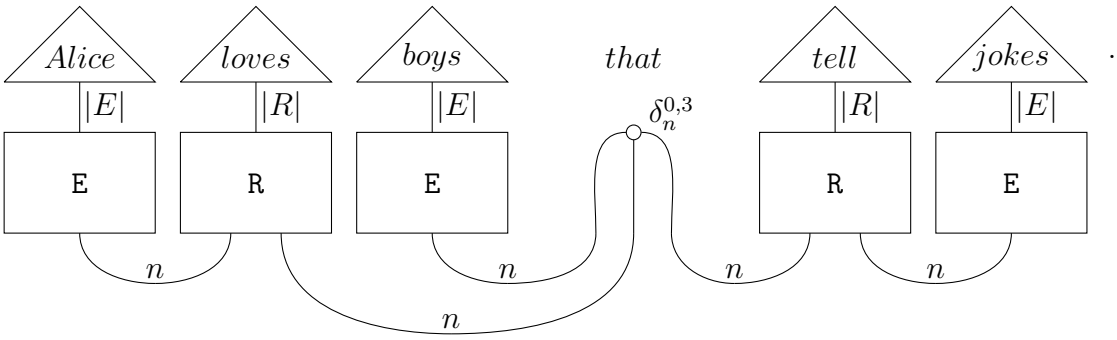


Relative clauses, as introduced in [11], allow us to encode a notion of *definition*. One of the requirements for our encoding matrices should be that a word is *similar* to its definition. As this is a hard constraint to express, we will begin by asking for equality:

Example 2.2. *If our corpus contains some philosophy, we ought to have*



We propose to apply relative clauses — together with the notion of duality discussed in the previous section — and show how in our model they generate *basic questions*. We start from the observation that computing the value of the scalar



should be *the same* as answering the question

BOB: "Does Alice love boys that tell jokes?"

Thus once we have computed some encoding matrices from a corpus, our semantic functor gives a way of computing the semantics natural language questions — from the similarity of the words in the sentence, and from its grammatical derivation. However, giving a semantics to the answers of such basic yes-no questions proved to be surprisingly hard to encode. Or rather to decode: given a scalar in a rig S, which we obtained by computing the semantics of a question, what natural language utterance is the answer?

While giving a semantics to positive answers is trivial — applying the reverse process and turning the question into a declarative sentence — negatives require a good encoding for the word "not":

DAN: "Alice does not."

Instead, we propose to adapt Lambek's analysis [1] to our framework, looking at *object* questions as *sentences with holes*. Intuitively, the word "Who'' is a process which takes an incomplete sentence "Alice loves —" on its right and transforms it into the question "Who does Alice love?". The role of the word "does" has already been discussed in the previous section: it does nothing. Or rather, nothing else than making the question sound grammatical to an English reader.

Definition 2.8 (Who and Whom)**.** *In a free rigid grammar $\mathcal{G}$ with designated types $s, q, o, n \in Ob(\mathcal{G})$ such that $n \leq o$, the (object)* question words *"Who" and "whom" are given by arrows*

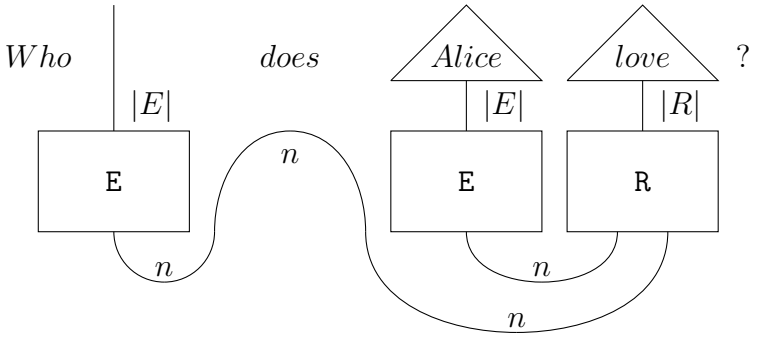
$$Who \ - \ ? \ : \ q \to (o^l)^l \times s^l$$
$$- \ whom? \ : \ q \to o \times s^r$$

where s and q are the sentence and question, o and n the object and noun types.

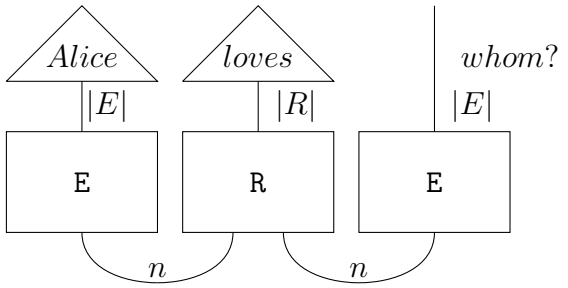
Example 2.3. *Take the set of nouns $E = \{ w \in V \mid T(w) = n \}$, we define*

$$F(s) = 1 \qquad F(q) = |E| \qquad F(o) = F(n) = n \in \mathbb{N}$$
$$F(n \leq o) = 1_n \qquad F(Who \ - \ ?) = F(\ - \ whom?) = \mathtt{E} \ : \ |E| \to n$$

Then the semantics of the question "Who does Alice love?" is given by:



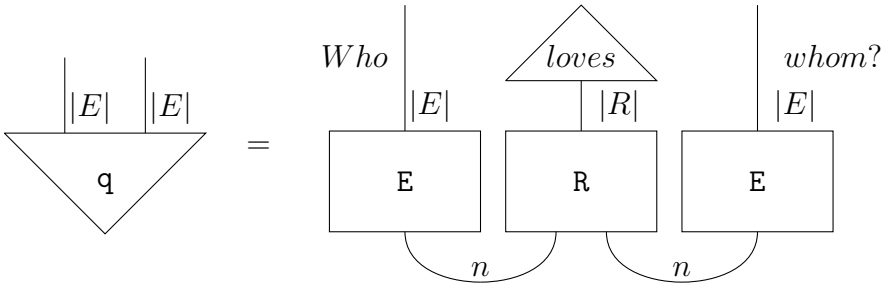
Using string diagrams, we can see graphically what Chomsky [2] calls the trace*:*



In pregroup grammars the trace is encoded by the *double adjoint type* $(o^l)^l \in G$, which we have already encountered in the definition of relative clauses:

> "Chomskyan linguists might say that the question word has been moved from the end to the beginning of the question, and I have placed a dash at the end to represent what they call a trace. Traces are not really needed for our approach, their place being taken by double adjoints." — Lambek, *From Word To Sentence* [1, p. 29]

$\mathtt{Mat}_S$ is compact closed — we have assumed the rig S to be commutative — hence by symmetry we can extend this analysis to that of subject questions, and we may also imagine giving the semantics of questions of two arguments, e.g.



In chapter 3 we propose instead to look at *anaphora* as a resource for generating a class of effects — which includes the image of basic questions as given here, and extends it to effects of type $\mathtt{q} : |E|^k \to 1$ for any number of arguments $k \in \mathbb{N}$.

2.3 Knowledge Graphs and Free Corpora

We believe that one of the main applications of our work would be in building concrete systems for interaction between human and machine — in this chapter, our aim is to model how the two could answer each other's questions. To that effect, we introduce *basic discourses* as mid-level representations between human language and machine-readable format.

In this section, we start by showing how a basic corpus — i.e. a collection of basic discourses — generates a *knowledge graph* in the *Resource Description Framework* (RDF) for the Semantic Web. We focus on the Boolean case and show that a knowledge graph generates what we call a *basic* semantic functor which in turn generates a *free corpus*.

Adapting Montague's framework for natural language semantics to our basic setup, we argue that the equivalence between these three definitions should be seen as a process which — once generalised beyond basic grammars and Boolean semantics — gives an abstract framework for *natural language understanding*. First, we fix the terms of our ontology by defining an *RDF signature*: a pair of finite sets E and R denoting *entities* and *relations* respectively. As in the first chapter we assume that the sets have been indexed.

Definition 2.9 (Knowledge Graph)**.** *Given a signature* (E, R)*, a* knowledge graph *is a subset* $K \subseteq \mathcal{T}$ *of the set of* triples $\mathcal{T} = E \times R \times E$*. The concrete implementation of a knowledge graph is also known as a* triple store*.*

Remark 2.1. *Every knowledge graph defines a graph in the sense of definition 1.4*

$$A = K \qquad O = E \qquad dom(x, r, y) = x \qquad cod(x, r, y) = y$$

In some sense, we have forgotten *the relation part of* K*.*

A knowledge graph encodes facts, we will assume that we can express those facts: i.e. that our vocabulary has been translated into RDF — every noun has been encoded as an entity, and every transitive verb as a relation.

Definition 2.10 (Basic Grammar)**.** *Given a signature* (E, R)*, the* basic grammar $\mathcal{G}_{E,R}$ *is the free rigid grammar generated by:*

- *the set of basic types* $\{ s, n \}$
- *a* noun arrow $e : 1 \to n$ *for every entity* $e \in E$
- *a* verb arrow $r : 1 \to v_{tr}$ *for every relation* $r \in R$*, where* $v_{tr} = n^r \times s \times n^l$

Lemma 2.2. *The set of* basic sentences *— i.e. utterances grammatical under the grammar* $\mathcal{G}_{E,R}$ *— is* $\mathcal{T} = E \times R \times E$*. The set of basic discourses is* $\mathcal{T}^* = (E \times R \times E)^*$*.*

Proof. Applying Lambek's switching lemma for pregroups [1] in our setup with no order on the basic types, we know that any reduction arrow $g : a \to b$ has the form $g = ec$ where $c : a \to z$ is generated by *contraction arrows* and $e : z \to b$ is generated by *expansion arrows* — which correspond to the left and right handside of definition 1.8 respectively. Hence for basic grammars, we can exclude expansions from reduction arrows while retaining the same definition of grammaticality.

Then given an utterance $u \in V^*$ and a reduction arrow $g : T^*(u) \to s$, the proof that u has to be a basic sentence of the form $\langle subject, verb, object \rangle \in E \times R \times E$, i.e. that

$$g \;=\; cup_n^r \otimes 1_s \otimes cup_n^l \;:\; n \times v_{tr} \times n \to s$$

proceeds by induction on the structure of pregroup derivations. Knowing that contraction arrows can only reduce the length of types, we are sure that the proof comes to an end, we refer the interested reader to [16].

Note that under the Curry-Howard correspondance, finding such pregroup derivations is isomorphic to parsing the syntax of sentences — investigating Lambek's side of this correspondance is one of the motivations for this thesis. □

We now show how if we're given a set of basic discourses, we can generate a knowledge graph from it — intuitively, this process amounts to collecting all the facts in a corpus.

Lemma 2.3. *Given a corpus of basic discourses* $C \subseteq \mathcal{T}^*$*, we can construct a knowledge graph:*

$$\mathcal{K}(C) \;=\; \left(\bigcup_{d \in C} \bigcup_{s \in d} s \right) \;\subseteq\; E \times R \times E$$

Moreover, any knowledge graph can be generated in that way.

Proof. The function $\mathcal{K} : \mathcal{P}(\mathcal{T}^*) \to \mathcal{P}(\mathcal{T})$ is injective: for all K we have $\mathcal{K}(K^*) = K$. We call $K^* \subseteq (E \times R \times E)^*$ the *free corpus* generated by K. □

After introducing basic discourses and knowledge graphs, we now look at the third side of the equivalence: we define a class of semantic functors which adapts Montague's framework to our basic setup.

Definition 2.11 (Basic Semantic Functor)**.** *A* basic semantic functor *is a functor* $F : \mathcal{G}_{E,R} \to \mathtt{Mat}_S$ *such that*

- $F(s) = 1$ *and* $F(n) = |E|$
- *the image of entities* $e \in E$ *is their one-hot vector* $F(e) = |e\rangle : 1 \to |E|$

Note that because $\mathcal{G}_{E,R}$ *is free, a basic semantic functor is uniquely determined by its image on the verb arrows* $F(r) : 1 \to |E| \times |E|$*, i.e. by an encoding matrix* $\mathtt{R} : |R| \to |E| \times |E|$*.*

Lemma 2.4. *Given a knowledge graph* $K \subseteq E \times R \times E$ *we can construct a basic semantic functor* $\mathcal{F}(K) : \mathcal{G}_{E,R} \to \mathtt{Mat}_S$ *such that the set of true sentences is* K*, the set of true discourses is* K^**. Moreover when* $S = \mathbb{B}$*,* $\mathcal{F}$ *is an isomorphism between basic semantic functors and knowledge graphs.*

Proof. In the case $S = \mathbb{B}$, currying gives that encoding matrices and of knowledge graphs are both functions $E \times R \times E \to \mathbb{B}$, the isomorphism being implicit in the indexing we use for one-hot vectors.

We exhibit one side of the construction which has the desired property for any S, i.e. given $K \subseteq \mathcal{T}$ we construct $\mathcal{F}(K) : \mathcal{G}_{E,R} \to \mathtt{Mat}_S$ such that for all utterance arrows $u : 1 \to a$ and reduction arrows to the sentence type $g : a \to s$

$$\mathcal{F}(K)(gu) = 1 \iff u \in K$$

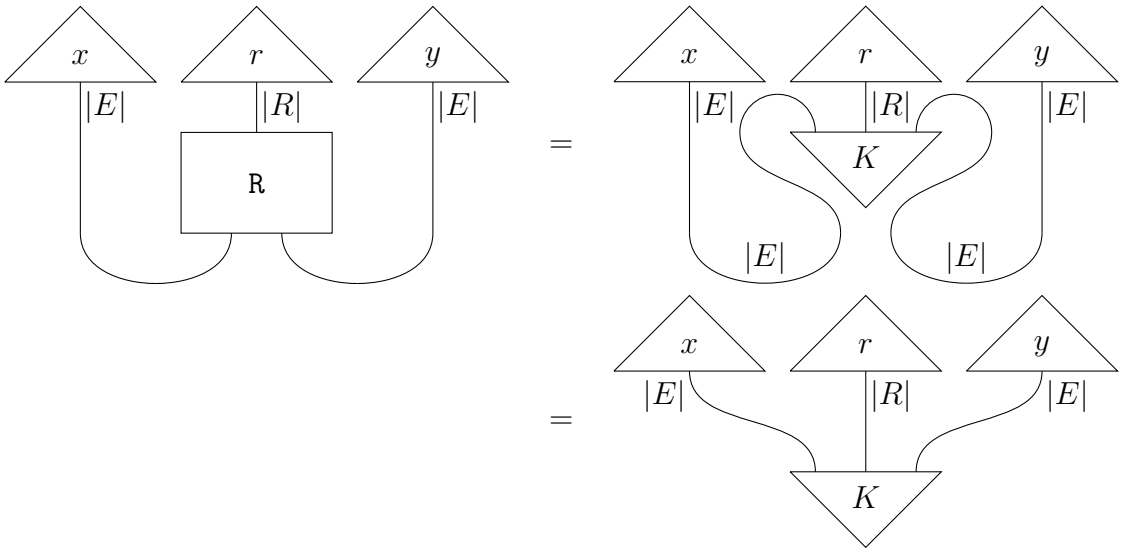
We first construct an effect $\langle K| : |\mathcal{T}| \to 1$ as follows:

$$\langle K| \;=\; \sum_{(x,r,y)\in K} \langle x| \otimes \langle r| \otimes \langle y| \;:\; |E| \times |R| \times |E| \to 1$$

where we take element-wise addition of matrices and we define:

$$\mathtt{R}_K \;=\; (\mathtt{1}_{|E|} \otimes \langle K| \otimes \mathtt{1}_{|E|})(\mathtt{cap}_{|E|} \otimes \mathtt{1}_{|R|} \otimes \mathtt{cap}_{|E|}) \;:\; |R| \to |E| \times |E|$$

From lemma 2.2 we know that a grammatical sentence has the form $u = x \otimes r \otimes y : 1 \to |\mathcal{T}|$ and we can derive $F(g)F(u) = (\mathtt{cup}_{|E|} \otimes \mathtt{cup}_{|E|})F(u) = \langle K|u\rangle$ graphically:



By construction we have that $\langle K|\,(|x\rangle \otimes |r\rangle \otimes |y\rangle) = 1$ if and only if $(x, r, y) \in K$. $\square$

These three lemmas are simply reformulations of the same object — the same function $K : E \times R \times E \to \mathbb{B}$ — as a knowledge graph, as a Boolean semantic functor, as a set of true sentences generating a corpus. Instead of focusing on any of the three formulation, we propose to look at the translation between them as an abstract process for natural language understanding.

Once extended beyond basic grammars and Boolean semantics, it suggests the following learning scheme for natural language. We borrow the terminology of machine learning — given a *training set* $C \subseteq V^*$ of true discourses:

1. parse the sentences of the corpus into basic discourses (lemma 2.2)
2. aggregate the facts of the corpus into a knowledge graph (lemma 2.3)
3. from the knowledge graph, compute an encoding matrix for verbs and generate a basic semantic functor (lemma 2.4)

Then this semantic functor can be used to predict whether some unseen *testing utterance* $u \in V^*$ is a true discourse — the set of basic semantic functor is the *hypothesis class*. Under this scheme, the construction we have given defines the most trivial learning algorithm: it remembers the examples it has seen as true and judges everything else as false. One would say that our construction *does not generalise*, however this is not entirely true: in a certain sense the algorithm learns to *close* its knowledge under concatenation — that is for all $u, v \in C$ we have $\mathcal{FK}(C)(u \otimes v) = 1$ even if $u \otimes v$ was not in the training set — as well as under the infix relation defined in example 1.1 $u \leq v \in C \implies \mathcal{FK}(C)(u) = 1$.

While this closure describes a compositional learning process, we conjecture that one can recover its distributional counterpart through *tensor factorisation* — i.e. finding low-dimensional approximations of the encoding matrix. In the conclusion we discuss directions towards integrating our framework with that of *knowledge graph completion* — a factorisation scheme for $\langle K| : |\mathcal{T}| \to 1$ through $S = \mathbb{C}$ introduced in [28], which is motivated both from the point of view of computational learning theory and that of abstract linear algebra.

Finally we believe this framework naturally leads to a perspective between that of *online learning* and *formal languages*: our semantic functor and its updating may be seen as the state and transitions of what could be called a *compositional perceptron*, extending that of Minsky and Papert [7]. Note that although the automata theory and machine learning communities seem totally separate today, they share a common founding father. Indeed, the same Kleene both formalised the star we use as a notation for free monoids, and introduced them in order to model the language of neural networks — which in the late sixties were called "nerve nets", see [29].

Automata theory and linguistics also share a common father: Chomsky's hierarchy is both an order on languages, and a dual order on the machines that generate those languages. At the bottom of the hierarchy, free monoids together with their union, intersection and complementation yield the notion of *regular languages*, generated by *finite-state* machines. Going one level higher in the hierarchy, push-down automaton implement *stacks* to store sequences of arbitrary-length in memory. These machines generate context-free grammars — which are equivalent to pregroup grammars, as discussed in section 1.1.

Does it contain any abstract reasoning concerning quantity or number? No.

Does it contain any experimental reasoning concerning matter of fact and existence? No.

Commit it then to the flames: for it can contain nothing but sophistry and illusion.

— David Hume, *An Enquiry Concerning Human Understanding*

3 Ambiguity and Semantic Unification

A lack of space and time forced us to commit most of this chapter to the flames, we apologise to the reader and leave only a few definitions and the sketch of a plan:

3.1 Anaphora and Basic Ambiguous Discourses

Definition 3.1 (Basic Anaphoric Functor)**.** *Given a signature* (E, R) *with a designated set of* anaphora $\{ us,\ them,\ they,\ \dots \} \subseteq E$*, we define the* basic anaphoric grammar $\mathcal{G}^{\dagger}_{E,R}$ *as the free rigid category generated by:*

- *the set of basic types* $\{ s, n, n^{\dagger} \}$
- *a noun arrow* $e : 1 \to n$ *for all* $e \in E$ *and a verb arrow* $r : 1 \to v_{tr}$ *for all* $r \in R$*, where* $v_{tr} = n^r \times s \times n^l$
- *an* anaphoric arrow $a : n^{\dagger} \to n$ *for all* $a \in \{ us,\ them,\ they,\ \dots \}$

Note that the symbol † *does* not *refer to the same* † *as that of the previous section* yet*.*

Definition 3.2. *A* basic anaphoric functor *is a semantic functor* $F : \mathcal{G}^{\dagger}_{E,R} \to \mathtt{Mat}_S$ *such that:*

- $F(s) = 1$ *and* $F(n) = F(n^{\dagger}) = |E|$
- $F(e) = \begin{cases} 1_{|E|} : |E| \to |E| & \texttt{if}\quad e \in \{ us,\ them,\ \dots \} \\ |e\rangle : 1 \to |E| & \texttt{otherwise} \end{cases}$

Lemma 3.1. *F is completely determined by its image on the verb arrows, hence by an encoding matrix* $\mathtt{E} : |R| \to |E| \times |E|$*.*

Proof. $\mathcal{G}^{\dagger}_{E,R}$ is free and by functoriality $F(r) : 1 \to F(v_{tr}) = F(n^l) \times F(1) \times F(n^r) = |E| \times |E|$. □

Lemma 3.2 (Basic Anaphoric Sentence)**.** *The set of* basic anaphoric sentences *is* $\mathcal{T} = E \times R \times E$*, the set of discourses is* $\mathcal{T}^*$*. Given a basic anaphoric functor* F*, the image of a basic sentence is given by an effect*

$$F(g)\ \ F\langle e_l, v, e_r\rangle \ \ :\ \ |E|^a \times |E|^b \to 1$$

where $(a, b) = \mathbf{1}^*_{\{us,them,\dots\}}(e_l, e_r)$ *is given by membership in* $\{us, them, \dots\}$ *and*

$$g \ \ = \ \ cup_n^r \otimes 1_s \otimes cup_n^l$$

Proof. This follows from the same reasoning as lemma 2.2. If $0 < l + r$, we say the sentence is *anaphoric*, when $l = 0$ we say the sentence subject is *unambiguous*, symmetrically for the object position. □

Lemma 3.3 (Basic Ambiguous Discourse)**.** *For a* basic ambiguous discourse $d \in \mathcal{T}^k$ *with* $k \in \mathbb{N}$ *sentences, the image of a basic semantic functor is given by an effect*

$$F(g^{(\times k)})F(d) \ \ :\ \ |E|^a \to 1$$

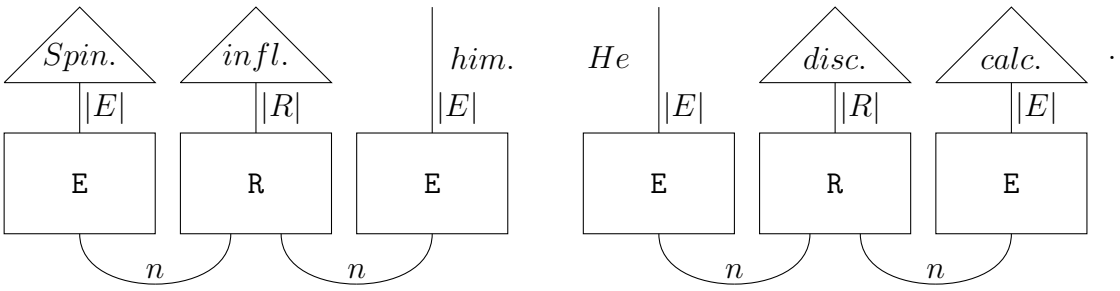
for $a \leq 2 \times k$ *the number of anaphoric expressions in the discourse.*

Proof. By example. □

Example 3.1. *The semantics of*

$$d \ \ = \ \ \langle Spinoza, influenced, him, he, discovered, calculus\rangle$$

is given by the effect



Remark 3.1. *The set* $\{\,(x, y) \mid F(g^{(\times 2)})\ F(d)\ \ |x, y\rangle \ = \ 1\,\}$ *should contain*

$$(Leibniz,\ \ Leibniz) \ \in \ E^2$$

but then it should also contain

$$(Wittgenstein,\ \ Newton) \ \in \ E^2$$

we need some constraints.

3.2 Resolution as Constrained Optimisation

Discourse representation theory is a computational framework introduced by Kamp in [30] for modeling natural language constraints. in this section, we use basic *discourse representation structures* (DRS) in the sense of Abramsky [31].

For a discourse $d \in \mathcal{T}^k$ with $F(d) : |E|^a \to 1$, the k basic sentences are called "literals" and the a anaphoras are called "variables". Given a set $\mathcal{D}(d) \subseteq \underline{a} \to E$ of *matching functions* satisfying some DRS constraints on the discourse d, we formulate *probabilistic anaphora resolution* as an *optimisation problem.*

Definition 3.3. *Given a* matching function $\mu : \underline{a} \to E$ *which assigns a* referent *in* E *to each anaphora in the effect* $F(d) : |E|^a \to 1$*, the* resolution of the ambiguity *is defined as the scalar:*

$$\mathcal{A}(d, \mu) \;=\; F(g)^{(\otimes a)} \;\; F(d) \;\; | \;\; \mu(0), \ldots, \mu(a-1) \;\; \rangle \;\; \in S$$

we define probabilistic anaphora resolution *as the following optimisation problem:*

$$\underset{\mu \in \mathcal{D}(d)}{\operatorname{argmax}} \;\; \mathcal{A}(d, \mu)$$

Note that the symbol $\mathcal{D}(d)$ *does* not *refer to the same* $\mathcal{D}$ *as that of the previous chapter* yet.

Lemma 3.4. *Given a matching function* $\mu : \underline{a} \to E$*, we construct an entity store*

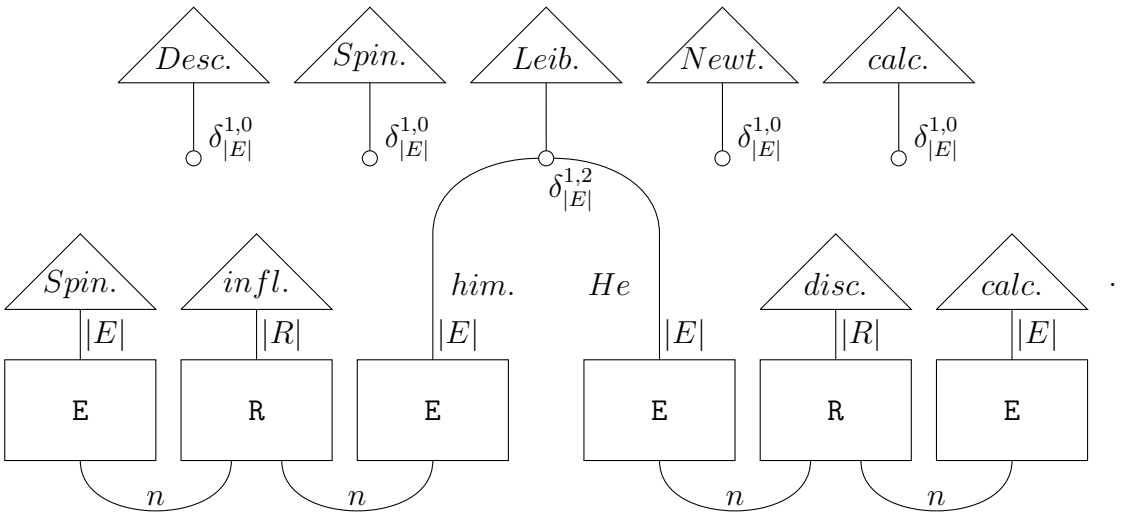
$$!E \;\; = \;\; \bigotimes_{e \in E} |e\rangle \;\; : \;\; 1 \to |E|^{|E|}$$

and a matching matrix $!\mu : |E|^{|E|} \to |E|^a$ *such that:*

$$\forall d : |E|^a \to 1 \;\; \cdot \;\; \mathcal{A}(\mathtt{d}, \mu) \;\; = \;\; \mathtt{d}(!\mu)(!E)$$

Proof. By example, intuitively we *draw the graph* of the function μ. □

Example 3.2. *We compute the resolution* $\mathcal{A}(\mathtt{d}, \mu) = d(!\mu)(!E) \in \mathbb{R}$ *as*



where $\mu = \{0, 1 \mapsto \mathit{Leibniz}\}$. *If we compute our encoding matrix* $\mathtt{E}$ *from an encyclopedia, we should expect this scalar to be close to* 1. *Choosing* $\mu = \{0, 1 \mapsto \mathit{Descartes}\}$ *instead should make it close to* 0.

Lemma 3.5. *In the* SPARQL Protocol and RDF Query Language*, a basic ambiguous discourse generates a query of the form*

$$\mathtt{SELECT} \ * \ \mathtt{WHERE} \ \{ \ \langle \ \mathtt{x}, \mathtt{r}, \mathtt{y} \ \rangle \ \in \ \mathtt{BGP} \ \}$$

where $\mathtt{x}$ *and* $\mathtt{y}$ *are taken to range over both referents and variables and* $\mathtt{BGP}$ *is the set of* basic graph patterns*, also called the* conjunctive *fragment of* SPARQL.

Proof. Future work.
See [32] for an analysis of the complexity of SPARQL conjunctive queries. □

3.3 Towards Semantic Unification

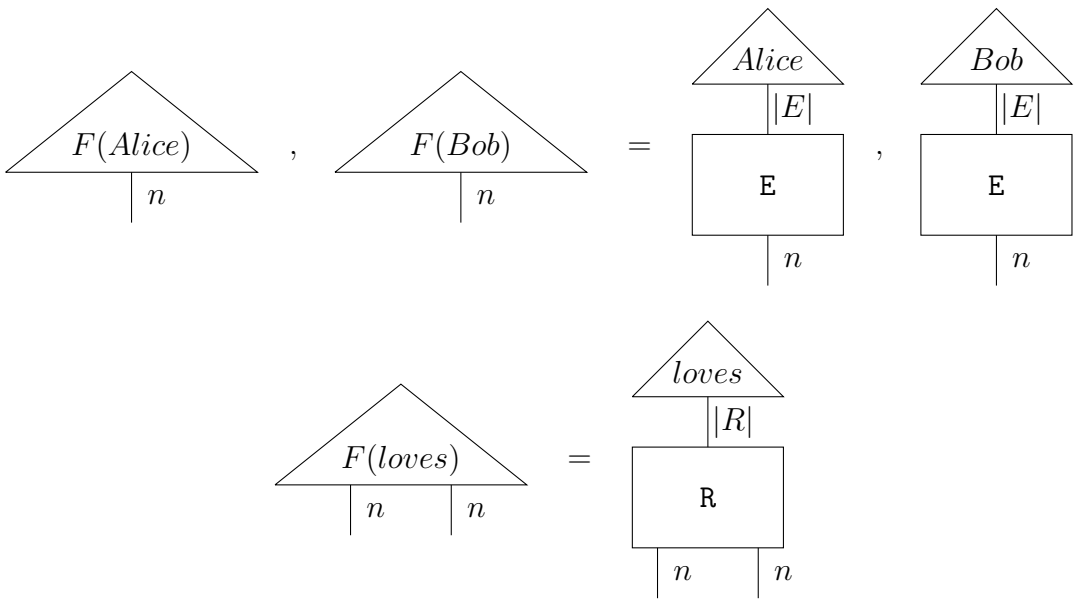
Note that throughout this thesis we have avoided to overload two technical terms, the meaning of which is ambiguous and has not yet been unified: *natural* and *universal*. We have considered the fragment defined in this thesis as a basic form of *natural language*, and we argued that the examples we took were representative of some *universal structures*: both of these terms belong to cognitive science.

We have not mentioned *natural transformations* and *universal constructions* of category theory. Although in some intuitive sense, the meaning of these two words should be *the same* in the two contexts: the motivation of thesis may be seen as attempting to *unify* their semantics. We refer the reader to [31] for a formal construction of *semantic unification*, as well as the M.Sc. thesis [33] for a development of this formalism in terms of *Kripke semantics* and *topos theory*. Making our framework fit in the *Elephant* of [34] is left to future work.

Conclusion

This thesis was the conclusion of two years of research in *Computer Science and the Foundations of Mathematics*. This is the conclusion of the conclusion, hence by duality we refer the reader back to the introduction either of this thesis, or of the copy of the publication in the following appendix.

We have attempted to unravel the duality between the same word: on one side, seen as an abstract syntactic unit for building sentences — Frege's principle of *compositionality* — on the other as the collection of all its contexts — Wittgenstein's "*meaning is use*". In terms of string diagrams, the aim of this thesis may be seen graphically as follows.



We gave a new formulation of the DisCoCat framework where the vector $F(Alice) : 1 \to n$ representing "Alice" is decomposed as the composition of its *one-hot encoding* $|Alice\rangle : 1 \to |E|$ followed by an *encoding matrix* $\mathtt{E} : |E| \to n$. Applying the same method to transitive verbs leads to an encoding matrix $\mathtt{R} : |R| \to n \times n$, which we propose to interpret as a *generalised knowledge graph*. In order to do so, we restrict ourselves to a natural language fragment we call *basic discourses*, and define basic *semantic functors* which are uniquely determined by such knowledge graphs. We used this decomposition to lift the semantics of isolated sentences to that of sets of sentences in sequence.

We proposed two directions for further research: modeling object and subject questions and their answers as knowledge graph queries of one or two arguments,

using relative clauses and anaphora as a resource for extending this to queries of arbitrary number of arguments. Other potential directions for future work using this approach include the following.

1. Integrating our approach with that of Hedges and Sadrzadeh [35], which models *generalised quantifiers* as *bialgebras*. We believe that our factorisation would allow to use these bialgebras to generate complex queries.

2. Giving a presentation of our approach in terms of *PROducts and Permutations categories* (PROPs), which have been used to study the interaction of bialgebras and *Frobenius algebras* — abstract counterparts to the generalised Kronecker delta of section 2.2 — see [36] [37].

3. Using the complex rig $\mathbb{C}$ in order to distinguish between objects and subjects by modeling them as the *conjugate* of one another, as developed by Trouillon et al. in [28] where a complex factorisation of the relation matrix $\mathtt{R}$ allows *dimensionality reduction* and *statistical learning*. A proper treatment of the object-subject distinction should also allow us to implement the switch from active to passive voice as *diagram rewriting* [38].

4. Investigating the relation of our abstract model with concrete implementations of question answering such as the *tensor product recurrent networks* of Palangi et al. [39]. We conjecture that the *hypergraph categories* introduced in [40] for modeling open and interconnected systems — together with Fong and Spivak's formulation of neural networks in terms of PROPs [41] — could lead to a *functorial implementation*: given some cognitive requirements for our model of language, we want to be *correct by construction*.

5. Extending our framework for anaphora to the more general problem of *coreference resolution* — which has applications in the automated analysis of electronic medical records, see [42]. We wish to apply our computational approach to cognition and linguistics to the study of human psychology, and we conjecture that the same notion of duality could apply — for example in modeling the "sticky switch" between the left and right hemispheres of the brain, which lies behind bipolar disorders [43].

Appendices

A

Towards Compositional Distributional Discourse Analysis

This is a copy of the publication by B. Coecke, D. Marsden, G. De Felice and the present author, which we will present at the CAPNS18 Workshop on the day of the present deadline. The title and details may be subject to minor updates, as the camera-ready version will be released before the end of the year.

Abstract

Categorical compositional distributional semantics provide a method to derive the meaning of a sentence from the meaning of its individual words: the grammatical reduction of a sentence automatically induces a linear map for composing the word vectors obtained from distributional semantics. In this paper, we extend this passage from word-to-sentence to sentence-to-discourse composition. To achieve this we introduce a notion of basic anaphoric discourses as a mid-level representation between natural language discourse formalised in terms of basic *discourse representation structures* (DRS); and knowledge base queries over the Semantic Web as described by *basic graph patterns* in the Resource Description Framework (RDF). This provides a high-level description of compositional algorithms for *question answering* and *anaphora resolution*, and allows us to give a picture of *natural language understanding* as a process involving both statistical and logical resources.

Introduction

In the last couple of decades, the traditional *symbolic* approach to AI and cognitive science — which aims at characterising human intelligence in terms of abstract logical processes — has been challenged by so-called *connectionist* AI: the study of the human brain as a complex network of basic processing units [44]. When it comes to human language, the same divide manifests itself as the opposition between two principles, which in turn induce two distinct approaches to Natural Language Processing (*NLP*). On one hand Frege's principle of *compositionality* asserts that the meaning of a complex expression is a function of its sub-expressions, and the way in which they are composed — *distributionality* on the other hand can be summed up in Firth's maxim "You shall know a word by the company it keeps". Once implemented in terms of concrete algorithms we have expert systems driven by formal logical rules on one end, artificial neural networks and machine learning on the other.

Categorical Compositional Distributional (*DisCoCat*) models, first introduced in [13], aim at getting the best of both worlds: the string diagrams notation borrowed from category theory allows to manipulate the grammatical reductions as linear maps, and compute graphically the semantics of a sentence as the composition of the vectors which we obtain from the distributional semantics of its constituent words.

In this paper, we introduce *basic anaphoric discourses* as mid-level representations between natural language discourse on one end — formalised in terms of basic *discourse representation structures* (DRS) [31]; and knowledge queries over the Semantic Web on the other — given by *basic graph patterns* in the Resource Description Framework (RDF) [32]. We construct discourses as formal diagrams of real-valued matrices and we then use these diagrams to give abstract reformulations of NLP problems: probabilistic *anaphora resolution* and *question answering*.

In the first two sections we introduce the notation used in the rest of the paper, then give a brief summary of how DisCoCat models can be used to turn declarative sentences into probabilistic `Ask`-type queries in the SPARQL protocol and RDF query language. In the third section, we extend this analysis to `Select`-type queries and show how they can be translated as "who" or "whom" questions in the simple case, as discourses making use of ambiguous pronouns such as "they" or "them" in more complex cases, i.e. when the SPARQL query contains more than two output-variables. In the last section we give a finer-grained analysis of anaphora resolution as a resource-sensitive process, copying word-vectors from memory and feeding them in the meaning computation of some anaphoric discourse.

We conclude our discussion with related work on deep neural networks and knowledge graph factorisation, as well as potential directions for modeling more involved linguistic phenomena and translating them in terms of knowledge base queries.

A.1 Diagrams of Matrices for Knowledge Graphs

First we fix an ordered set of *scalars* $\mathcal{S}$ with addition $+$, multiplication $\times$ and units $0, 1 \in \mathcal{S}$ respectively, then we write $\mathtt{M} : a \to b$ for $a \times b$ matrices with entries in $\mathcal{S}$ and $\mathtt{id}_n : a \to a$ for the $a \times a$ identity matrix. For all $a, b, c \in \mathbb{N}$ and matrices

$\mathtt{M}_1 : a \to b$, $\mathtt{M}_2 : b \to c$, we have their composition $\mathtt{M}_2\mathtt{M}_1 : a \to c$ given by matrix multiplication. We also have element-wise addition $\mathtt{M} + \mathtt{N}$ of matrices $\mathtt{M}, \mathtt{N} : a \to b$. Hence given a choice of scalars (where $+$ and $\times$ respect the axioms of a semi-ring) we have the structure of a category $\mathtt{Mat}_{\mathcal{S}}$, with the natural numbers as objects and matrices as arrows. We will focus on the Boolean case $\mathcal{S} = \mathbb{B}$ with $+ = \vee$ and $\times = \wedge$ and on the non-negative reals $\mathcal{S} = \mathbb{R}^+$ encoding probabilities. Note that a proper treatment of normalisation is beyond the scope of this paper.

We will manipulate matrices using the *string diagrams* notation for monoidal categories — for a reference guide to graphical languages see [20], for an introduction targeted at a general audience see [45] — we depict $\mathtt{id}_1 : 1 \to 1$ as the empty diagram and other identity matrices as labeled wires; we represent non-trivial matrices as boxes with labeled input and output wires and their composition as vertical concatenation. Finally, horizontal juxtaposition of boxes depicts the Kronecker product $\mathtt{M} \otimes \mathtt{N} : a \otimes c \to b \otimes d$ of matrices $\mathtt{M} : a \to b$, $\mathtt{N} : c \to d$. We overload the notation and also write $a \otimes b$ for multiplication of natural numbers, i.e. the dimension of the tensor space.

We want to translate natural language into machine-readable format through linear algebra, hence we reformulate the standard Semantic Web terminology in terms of vectors and matrices. We assume that our vocabulary has been encoded in RDF: nouns come from an ordered set of *entities* $E = \{ Alice, boys, \dots \}$, verbs come from an ordered set of *relations* $R = \{ loves, tell, \dots \}$. In Semantic Web languages, relations are always binary so we will focus on transitive verbs, for a translation of adjectives and intransitive verbs in RDF see [46]. We encode every word as a binary code with one bit up, also called a *one-hot vector* — both entities $e_i \in E$ and relations $r_j \in R$ correspond to one-hot row vectors, of dimension $|E|$ and $|R|$ respectively. We denote these one-hot vectors using Dirac's bra-ket notation:

$$|e_i\rangle \;=\; \Big(\underbrace{\overbrace{0 \;\dots\; 0}^{i}\; 1\; 0 \;\dots 0}_{|E|} \Big) \;:\; 1 \to |E|$$

$$|r_j\rangle \;=\; \Big(\underbrace{\overbrace{0 \;\dots\; 0}^{j}\; 1\; 0 \;\dots 0}_{|R|} \Big) \;:\; 1 \to |R|$$

The corresponding one-hot column vectors are denoted $\langle e_i| : |E| \to 1$ and $\langle r_j| : |R| \to 1$. Row vectors are also called *states*, in the diagrammatic language we depict them as triangles with no inputs; similarly column vectors are called *effects* which we depict as triangles with no outputs.

Composing a state with an effect of the same dimension yields a diagram with no inputs or ouputs: a scalar in $\mathcal{S}$. Thus, in the case $\mathcal{S} = \mathbb{R}^+$ any effect $\mathtt{q} : n \to 1$ induces an ordering of the states $\mathtt{a}, \mathtt{b} : 1 \to n$, take $\mathtt{a} \leq \mathtt{b}$ if and only if $(\mathtt{qa}) \leq (\mathtt{qb})$. When $\mathcal{S} = \mathbb{B}$, $\mathtt{q}$ yields a subset of the states: $\mathtt{a} : 1 \to n$ is in that subset if and only if $\mathtt{qa} = 1$. In both cases, the one-hot column vector $\langle e| : |E| \to 1$ corresponding to an entity $e \in E$ acts as a test of identity with e: for all $e' \in E$ we have $\langle e|e'\rangle = 1$ if and only if $e' = e$.

A *knowledge graph* (also called an *ontology*) is given by a set of *RDF triples* $K \subseteq E \times R \times E$ which contains all the true statements that we care about, e.g.

$(Alice, loves, Bob), (Bob, hates, Charles) \in K$. We can turn any knowledge graph into an effect by summing over the column vectors of its triples:

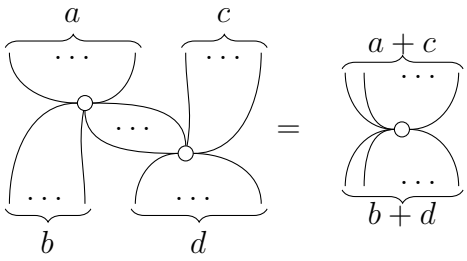
$$\langle K| \;=\; \sum_{(s,v,o)\in K} \langle s| \otimes \langle v| \otimes \langle o| \;\;:\;\; |E| \otimes |R| \otimes |E| \to 1$$

Then for any triple $t \in E \times R \times E$, if we compose the state $|t\rangle : 1 \to |E| \otimes |R| \otimes |E|$ with the effect $\langle K|$, the scalar $\langle K|t\rangle \in \mathcal{S}$ we obtain is equal to 1 if and only if $t \in K$— we consider this scalar as the outcome of a basic knowledge graph query. In order to study more complex queries, we need a way to wire knowledge graphs together: we do this using the *generalised Kronecker delta* together with the following standard results.

Definition A.1. *For $a, b, n \in \mathbb{N}$, the* (a, b) Kronecker delta *over n — which we depict as an a-input, b-output* spider *— is the matrix:*

$$\delta_n^{a,b} \;=\; \sum_{i<n} |e_i\rangle^{(\otimes b)} \langle e_i|^{(\otimes a)} \;\;:\;\; n^a \to n^b \tag{A.1}$$

Theorem A.1. *For all $n \in \mathbb{N}$, the set of matrices $\left\{\delta_n^{a,b}\right\}_{a,b\in\mathbb{N}}$ obeys the* spider fusion *equations:*



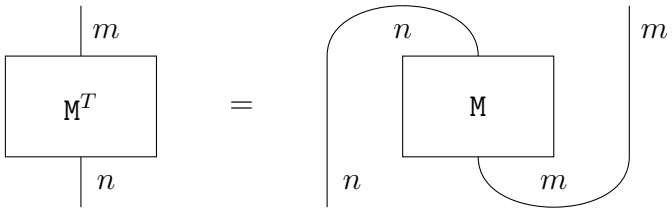
Proof. We have: $\left(\mathtt{id}_n^{(\otimes b)} \otimes \delta_n^{(k+c),d}\right)\left(\delta_n^{a,(b+k)} \otimes \mathtt{id}_n^{(\otimes c)}\right) \;=\; \delta_n^{(a+c),(b+d)}$, see [27]. Note how the string diagrams make the topological nature of the equation apparent while hiding the bureaucracy of indices. □

The $(2, 0)$ and $(0, 2)$ Kronecker deltas — which we depict as *cups* and *caps* respectively — make $\mathtt{Mat}_{\mathcal{S}}$ a *compact closed* category. We recall a couple of useful properties of cups and caps.

Corollary A.1.1. *For all $n \in \mathbb{N}$, the effect $\mathtt{cup}_n = \delta_n^{2,0} : n \otimes n \to 1$ and the state $\mathtt{cap}_n = \delta_n^{0,2} : 1 \to n \otimes n$ obey the* snake equations*:*

Proof. We have: $(\mathtt{id}_n \otimes \mathtt{cup}_n)\;(\mathtt{cap}_n \otimes \mathtt{id}_n) \;=\; \mathtt{id}_n \;=\; (\mathtt{cup}_n \otimes \mathtt{id}_n)\;(\mathtt{id}_n \otimes \mathtt{cap}_n)$ which follows from Theorem A.1 and $\delta_n^{1,1} \;=\; \sum_{i<n} |e_i\rangle\langle e_i| \;=\; \mathtt{id}_n$, see [22]. □

Corollary A.1.2. *We can define the transpose of a matrix* $\mathtt{M} : m \to n$ *as:*



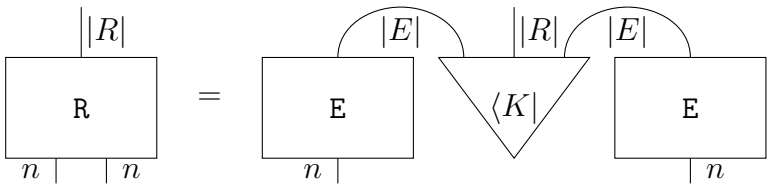
Proof. We have $\mathtt{M}^T = (\mathtt{id}_n \otimes \mathtt{cup}_m)\ (\mathtt{id}_n \otimes \mathtt{M} \otimes \mathtt{id}_m)\ (\mathtt{cap}_n \otimes \mathtt{id}_m)\ :\ n \to m$, see [22]. □

A.2 Diagrams for Distributional Compositional Semantics

Lambek introduced *pregroup grammars* as a means of encoding the grammatical structure of natural language. In [13, 14], Clark et al. observe that both pregroups and vector spaces carry the same mathematical structure: both are compact closed categories. This allows us to assign vectors to words and automatically combine them together using a linear map induced by the grammatical reduction. In this section, we give a graphical presentation of the concrete implementation described in [47] and reformulate their construction in terms of knowledge graphs.

The first step is to build a *noun space* N which we will assign to the noun type of our pregroup grammar. This can be done by collecting co-occurrence data over a large natural language corpus: the dimension n of the noun space can be seen as a hyper-parameter of the model, where in the simple case each dimension corresponds to a "context word". However, our model is agnostic of the exact method used to construct N: we will only assume that every entity e is assigned a vector $v_e \in N$, and store them as one matrix $\mathtt{E} = \sum_{e \in E} v_e \langle e|$. This encoding matrix $\mathtt{E} : |E| \to n$ gives us a measure of similarity between the entities, indeed whereas we have $\langle cat|dog\rangle = 0$ now we expect the inner product $\langle cat|\, \mathtt{E}^T\mathtt{E}\, |dog\rangle \in \mathbb{R}^+$ to be strictly greater than zero: their contexts share at least the words "pet", "food", etc.

Then for every verb $v \in R$, we compute its meaning by summing over the outer products $\mathtt{E}\,|s\rangle \otimes \mathtt{E}\,|o\rangle : 1 \to n \otimes n$ of all the encoded entities that it relates, i.e. all $s, o \in E$ such that $(s, v, o) \in K$. Given an encoding matrix $\mathtt{E}$ and a knowledge graph K we store this construction as a matrix:



encoding every relation v in the *verb space* $N \otimes N$. Finally, we can compute the semantics of "subject verb object" sentences using the following recipe: let $\mathtt{T} = \mathtt{E} \otimes \mathtt{R} \otimes \mathtt{E}$ be the encoding matrix for triples,

1. tensor the encoded states for $t = (s, v, o) \in E \times R \times E$ together

$$\mathtt{T}\,|t\rangle = \mathtt{E}\,|s\rangle \otimes \mathtt{R}\,|v\rangle \otimes \mathtt{E}\,|o\rangle : 1 \to n \otimes (n \otimes n) \otimes n$$

2. translate the grammar into a linear map in $\mathtt{Mat}^{+}_{\mathbb{R}}$, here the effect

$$\mathtt{G} = \mathtt{cup}_n \otimes \mathtt{cup}_n : (n \otimes n) \otimes (n \otimes n) \to 1$$

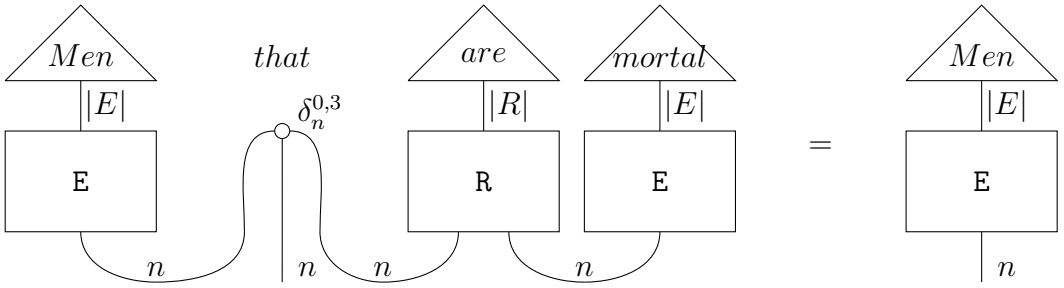
3. compose word meanings with grammar to get a scalar $\mathtt{GT}\,|t\rangle \in \mathbb{R}^{+}$.

Computing these real-valued scalars allows to generalise to unseen sentences and outperform the standard bag-of-words approach on word disambuigation tasks, where the experimental data can be seen as a small knowledge graph [47]. We can give the semantics of more complex sentences — hence more complex knowledge graph patterns — by modeling relative pronouns as three-output spiders over n. For any $\mathtt{x}, \mathtt{y} : 1 \to n$, if we encode the word "that" as the Kronecker delta $\delta^{0,3}_n : n^3 \to 1$ and apply the appropriate grammatical reduction we get:

$$\mathtt{G}'\ \Big(\ \mathtt{x} \otimes \mathit{that} \otimes \mathtt{y}\ \Big)\ =\ \delta^{2,1}_n\ \Big(\ \mathtt{x} \otimes \mathtt{y}\ \Big)\ :\ 1 \to n$$

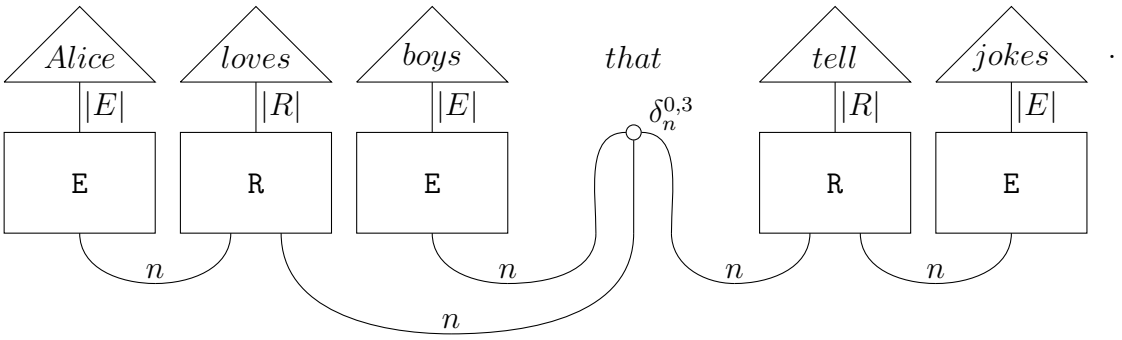
where $\mathtt{G}' = \mathtt{cup}_n \otimes \mathtt{id}_n \otimes \mathtt{cup}_n : n^5 \to 1$. This the coordinate-wise multiplication of the vectors $\mathtt{x}$ and $\mathtt{y}$, which can be seen as the intersection of entities in co-occurrence space, see [24] and [25].

Example A.1. *If we encode some philosophy into our knowledge graph K, we want that:*



i.e. all men are mortal.

Example A.2. *We compute the semantics of "Alice loves boys that tell jokes." as the following scalar:*

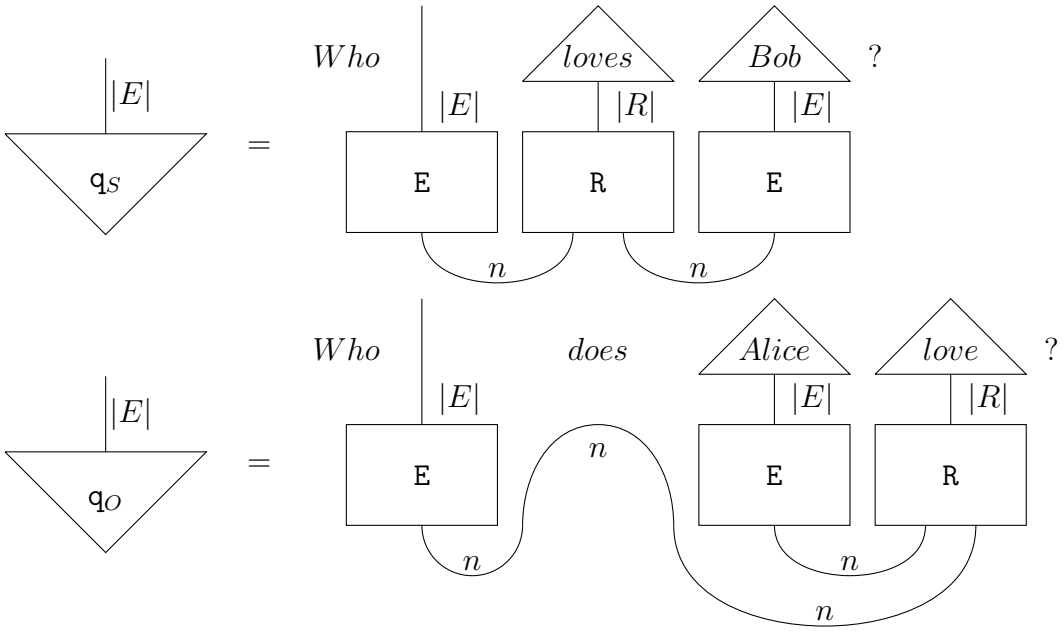


A.3 From Questions and Discourse to Knowledge Graph Queries

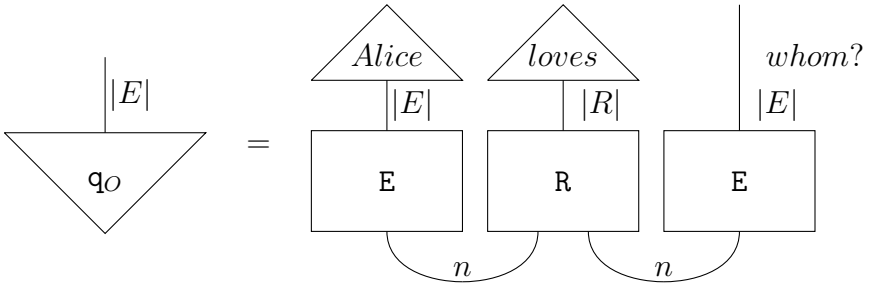
From an NLP perspective, computing the diagram for the sentence above can equivalently be seen as answering the question "Do the boys that Alice loves tell jokes?". We will exploit this question-answer duality to show how natural language can be translated into machine-readable format in a structure-preserving way. Given a concrete implementation of a knowledge graph, also called a *triplestore*, the semantics of yes-no questions can be interpreted as the `true - false` output of ASK-type queries in the SPARQL protocol and RDF query language. As a first step towards more complex query patterns, we extend this language-to-database translation to object and subject questions as SELECT-type queries.

In [1], Lambek gives a thorough analysis of the grammar of questions in terms of pregroups. Here we will leave the subtleties of the English language aside and only assume designated grammatical types Q_O and Q_S for object and subject questions respectively ; in our semantic category $\mathtt{Mat}_{\mathcal{S}}$, both correspond to effects $\mathtt{q} : |E| \to 1$. Indeed, given a potential answer $a \in E$ the inner product $\mathtt{q}\,|a\rangle \in \mathcal{S}$ gives us a measure of how well a answers the query $\mathtt{q}$; comparing these scalars allow us to translate a natural language question into a ranking of its possible answers. Note that if we interpret $\mathtt{q}$ with $\mathcal{S} = \mathbb{B}$ we get crisp answers while taking $\mathcal{S} = \mathbb{R}^+$ and the encoding $\mathtt{T} = \mathtt{E} \otimes \mathtt{R} \otimes \mathtt{E}$ yields real-valued approximations.

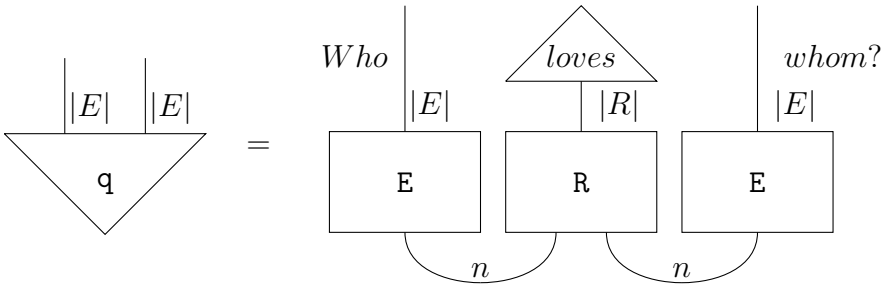
In order to form a subject question, Lambek starts by constructing incomplete sentences with a hole in the subject position, e.g. "- loves Bob". Object questions proceed dually, for example "Alice loves -". Question words like "who" and "whom" can then be seen as processes which turn these incomplete sentences into questions of type Q_S and Q_O. In our setup, the semantics of both words are given by the encoding matrix $\mathtt{E} : |E| \to n$, they take the state for the answer as input and feed it in the computation of the corresponding query. The semantics for "Who loves Bob?" and "Who does Alice love?" are given by:



Note that the semantics of "does" is given by a cap, i.e. a maximally-entangled state of two noun sytems, this is discussed further in [23]. Indeed, using the symmetry $(\langle x| \otimes \langle y|)\ \mathtt{cap}_{|E|} = (\langle y| \otimes \langle x|)\ \mathtt{cap}_{|E|}$ for all $x, y \in E$ and the snake equations we can rewrite object questions and see the cap serve as an information-passing mechanism:



We can extend this to two-variable SELECT queries such as



however, general SELECT queries can have an unbounded number of variables whereas natural language questions of three arguments or more will sound awkward to a human speaker — picture a stranger asking you: "Who did what where with whom?"... Instead, we argue that natural language speakers can express complex query patterns through ambiguous *discourse* together with *anaphora* as a resource for interaction of meanings. Concretely, we translate personal pronouns such as "us" and "them" in terms of SPARQL query variables ranging over the set of entities E, then we say a sentence is *anaphoric* when it contains a personal pronoun: its meaning depends on that of another expression in context.

We define an *atomic sentence* as one of the form "subject verb object", where both subject and object may be either entities or personal pronouns. In the anaphoric case we obtain a diagram with an open wire for the anaphoric expression, representing the input required from the context in order to compute the meaning of the sentence. Once the ambiguity has been resolved, i.e. once we have assigned an entity to the pronoun, we get a diagram with no open wires as in the unambiguous case of Section 2. Finally a *basic anaphoric discourse* is defined as a sequence of these atomic sentences, when it has k pronouns in total its semantics is an effect of dimension $|E|^k$: a knowledge graph query of k variables.

Definition A.2. *Take* $a, b \in \{0, 1\}$*, an* atomic sentence *is a matrix in*

$$\left\{ \mathtt{G} \left(\mathtt{sub} \otimes \mathtt{R}\,|v\rangle \otimes \mathtt{obj} \right) \ : \ |E|^a \otimes |E|^b \to 1 \right\}$$

where the verb v ranges over the relations R. If $a = 1$, the sentence is anaphoric *in the subject position and we take* $\texttt{sub} = \texttt{E} : |E| \to n$. *If $a = 0$, sub ranges over the encoded entities* $\texttt{E}|s\rangle : 1 \to n$ *and the sentence subject is* unambiguous. *Symmetrically for the object position:*

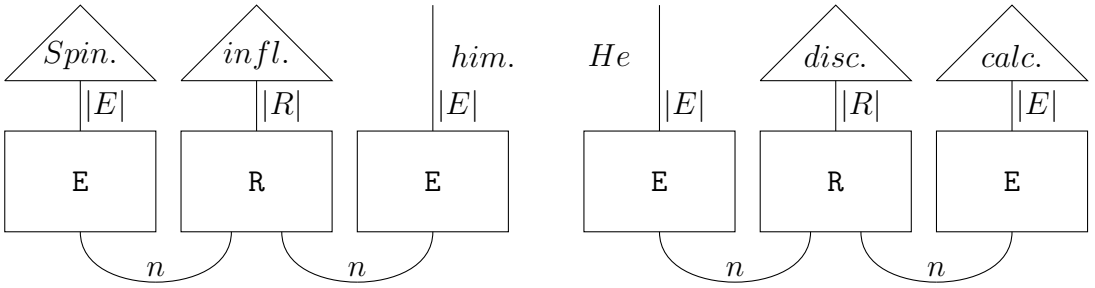
$$\texttt{obj} \;=\; \begin{cases} \texttt{E} & : |E| \to n \qquad \texttt{if} \quad b = 1 \\ \texttt{E}|o\rangle & : \;\; 1 \to n \qquad \texttt{for } o \in E \texttt{ otherwise} \end{cases}$$

We define a basic anaphoric discourse *as the matrix* $\texttt{d} = \bigotimes_{i<l} \texttt{t}_i : |E|^k \to 1$ *obtained by tensoring a list of l atomic sentences*

$$\texttt{t}_i \;:\; |E|^{a_i} \otimes |E|^{b_i} \to 1$$

where $k = \sum_{i<l} (a_i + b_i)$, *i.e. the discourse* d *contains k anaphoric expressions in total.*

Example A.3. *The semantics of "Spinoza influenced him. He discovered calculus." is given by the effect*



A.4 Language Understanding as a Process

We have seen that the semantics of *basic anaphoric discourses* involving k anaphoric expressions can be given in terms of knowledge graph queries of k variables. However, interpreting directly the effect $|E|^2 \to 1$ corresponding to the discourse in the example above as a SPARQL query of two variables will give counter-intuitive answers such as:

"Spinoza influenced Wittgenstein. Newton discovered calculus."

whereas an English speaker will have naturally assumed that the two pronouns "him" and "he" have to refer to the same entity. Without this constraint, maximising over the complex query would be equivalent to maximising the outcome for the two sentences independently.

Discourse representation theory is a computational framework first introduced in [30] for manipulating such natural language constraints. In this paper, we model basic *discourse representation structures* (DRS) in the sense of [31], where entities are called "discourse referents", atomic sentences "literals" and pronouns "variables". Using the set of constraints formalised in terms of DRS, together with our encoding $\texttt{T} = \texttt{E} \otimes \texttt{R} \otimes \texttt{E}$ allows us to formalise *probabilistic anaphora resolution* as an optimisation problem.

Definition A.3. *Given a* matching function $\mu : \{0, \ldots, k-1\} \to E$ *which assigns an entity in* E *to each pronoun in* $\mathtt{d}$*, the* resolution of the ambiguity *in the discourse* $\mathtt{d}$ *is defined as the following scalar:*

$$\mathcal{A}(\mathtt{d}, \mu) \;=\; d \bigotimes_{i<k} |\; \mu(i) \;\rangle \;\in \mathcal{S}$$

Given a set $\mathcal{D}(\mathtt{d}) \subseteq E^{\{0 \ldots k-1\}}$ *of matching functions satisfying some DRS constraints for the anaphoric discourse* $\mathtt{d}$*, we define* probabilistic anaphora resolution *as the following optimisation problem:*

$$\underset{\mu \in \mathcal{D}(\mathtt{d})}{\operatorname{argmax}} \;\; \mathcal{A}(\mathtt{d}, \mu)$$

The function μ can only give a static picture of the ambiguity resolution's result, instead we want a fine-grained diagram of the computational process involved. We give a resource-aware reformulation of anaphora resolution in terms of an *entity-store*, defined as the state:

$$!E \;=\; \bigotimes_{e \in E} |e\rangle \;\;:\;\; 1 \to |E|^{|E|}$$

From the matching function μ we construct a *matching process* $!\mu$ which takes an entity-store $!E$ and feeds the appropriate entities into the ambiguous sentences in $\mathtt{d}$. Graphically this construction amounts to drawing the graph of the function μ, labeling all the wires with $|E|$ and interpreting this as a diagram in $\mathtt{Mat}_{\mathcal{S}}$. When i edges of the graph meet at a vertex in E — i.e. when the same entity is referenced by i distinct pronouns — we witness this with a Kronecker delta: the head of a 1-input i-output spider. Whereas in section 2 the multi-input one-output spider acted as intersection of meanings, here the dual process copies an entity state and feeds it to its i outputs i.e. for all $e \in E$ we have that:

$$\begin{aligned}
(\delta^{1,i}_{|E|})\,|e\rangle \;&=\; \sum_{e' \in E} |e'\rangle^{(\otimes i)}\, \langle e'|e\rangle \\
&=\; \left(\sum_{e' \neq e} |e'\rangle^{(\otimes i)} \underbrace{\langle e'|e\rangle}_{=0} \right) \;+\; |e\rangle^{(\otimes i)} \underbrace{\langle e|e\rangle}_{=1} \;=\; |e\rangle^{(\otimes i)}
\end{aligned}$$

Theorem A.2. *Given a matching function* $\mu : \{0, \ldots, k-1\} \to E$ *we can construct a matching process* $!\mu : |E|^{|E|} \to |E|^k$ *such that:*

$$\forall\; \mathtt{d} \;:\; |E|^k \to 1 \;\cdot\; \mathcal{A}(\mathtt{d}, \mu) \;=\; \mathtt{d}(!\mu)(!E)$$

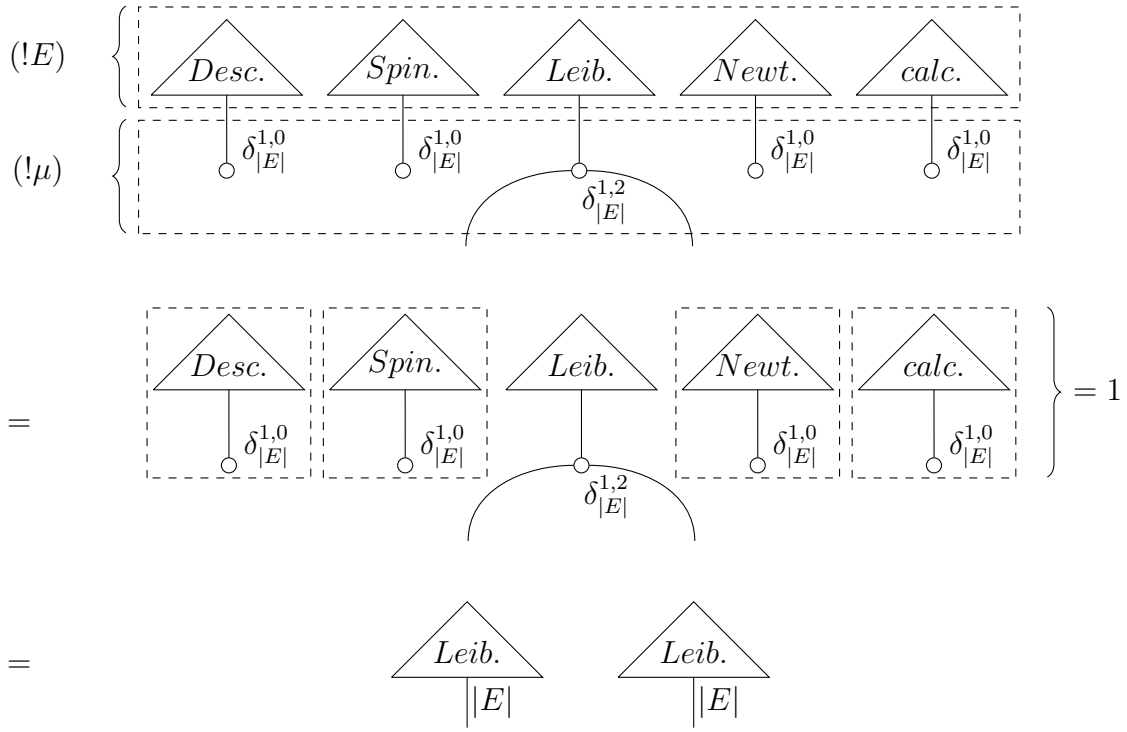
Proof. In set-theoretic terms, $!E$ is the list of all the elements $e \in E$, ordered by the indices of the corresponding vectors $|e\rangle : 1 \to |E|$. Then $!\mu$ is the incidence matrix of the graph for the function

$$f_\mu \;(\; e_0, \;\ldots,\; e_{|E|-1} \;) \;=\; (\; e_{\mu(0)}, \;\ldots,\; e_{\mu(k-1)} \;) \;\;:\;\; E^{|E|} \to E^k$$

In the case when the graph of the function f_μ is not planar, constructing $!\mu$ as a diagram in $\mathtt{Mat}_{\mathcal{S}}$ would require a formalisation of the intuitive notion of "wire-crossing", which can be obtained using the *symmetry morphisms* of $\mathtt{Mat}_{\mathcal{S}}$. This

falls outside the scope of our discussion but we refer the interested reader to any introduction to *symmetric monoidal categories* (e.g. [45]) which are the suitable framework in this context. □

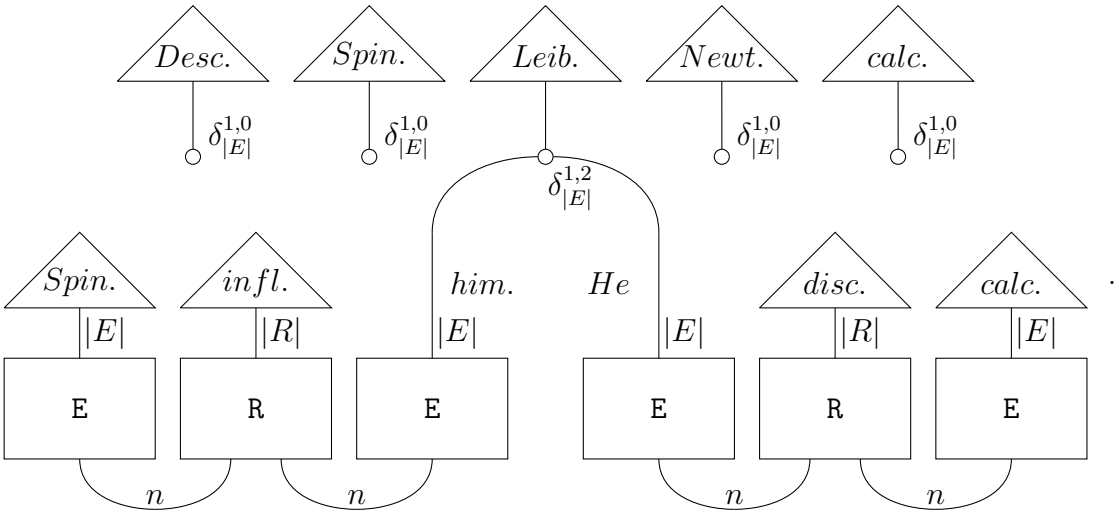
Example A.4. *Take* $E = \{\, Descartes, Spinoza, Leibniz, Newton, calculus \,\}$*, let* μ *be the constant function* $\{\, 0, 1 \mapsto Leibniz \,\}$*. We have:*



Even though our construction involves systems of exponential dimensions, the concrete computation it induces is linear in the size $n \times |E|$ of the encoding matrix: we are only retrieving and copying vectors. In effect, what this abstract formulation allows is to manipulate the knowledge graph, the entity encoding and matching explicitly, while hiding the bureaucratic process of labeling, indexing and wiring the entity-vectors required to compute complex queries.

We can now focus on what really is the core of the model: the interaction between the symbolic world of RDF and the statistical world of distributional semantics. In diagrammatic terms, this is witnessed by the interaction of distinct spiders, i.e. Kronecker deltas over two different dimensions: $\delta_{|E|}$ for copying and matching entities, δ_n for their intersection in the noun space N.

Example A.5. *We compute the resolution* $\mathcal{A}(\mathtt{d}, \mu) = d(!\mu)(!E) \in \mathbb{R}$ *as the scalar:*



If we extract our encoding matrix E *and knowledge graph* K *from an encyclopedia, we should expect this scalar to be close to* 1*. Choosing* $\mu = \{0, 1 \mapsto Descartes\}$ *instead should make it close to* 0*.*

In order to get the class of all basic DRS, we need to assume that for every $r \in R$ we have its negation $not(r) \in R$ such that $E(x, not(r), y) \in K$ if and only if $(x, r, y) \notin K$. Lifting this function $not : R \to R$ to the matrix of its graph $|not\rangle : |R| \to |R|$ and deriving a compositional semantics for natural language negations in $Mat_{\mathcal{S}}$ from this is left for future work. Nevertheless, if we restrict ourselves to the negation-free fragment, we can already give a semantics for basic anaphoric discourses in terms of the conjunctive fragment of SPARQL. Indeed, we can generate any query of the form

$$\texttt{SELECT} \ * \ \texttt{WHERE} \ \{ \ \langle \ \mathtt{x}, \mathtt{r}, \mathtt{y} \ \rangle \ \in \ \texttt{BGP} \ \}$$

where x and y are taken to range over both entities and variables. In the Semantic Web community, the set of triples BGP is also called a *basic graph pattern* [32].

Conclusion

We have developed a process-theoretic framework for natural language understanding, making use of both statistical and logical resources to translate question answering and anaphora resolution as probabilistic knowledge base queries. Using formal diagrams of matrices we give a high-level picture of NLP algorithms which are motivated both from the point of view of AI and of human cognition.

Even though we focused on $\mathcal{S} = \mathbb{B}$ and $\mathcal{S} = \mathbb{R}^+$, our discussion can be extended to any other semi-ring structure such as the unit interval $[0, 1]$ with *min* and *max* encoding fuzzy truth values. It is also possible to incorporate further structure such as the convexity property used in conceptual space models of cognition, or taking proof relevance into account and record *why* are entities related [4].

In the RDF language, variables can also serve as *blank nodes* also called *anonymous resources*: entities that are used in the computation but do not appear in the output of the query, e.g. `y` in

$$\texttt{SELECT x WHERE } \{ \ \langle \ \texttt{x}, \texttt{loves}, \texttt{y} \ \rangle, \ \langle \ \texttt{y}, \texttt{loves}, \texttt{x} \ \rangle \ \}$$

In $\texttt{Mat}_S$, we conjecture that these would translate in terms of an *entity-discarding* process $\delta^{0,1}_{|E|} : 1 \to |E|$.

The translation from pregroup grammars to RDF triples is developed further in [46], in order to model count nouns (e.g. "a cat") and relative pronouns (e.g. "whose tail"), Delpeuch and Preller introduce a notion of categories with *side effects* — i.e. creation and update of blank nodes. It would be interesting to investigate the connections with our current work.

We would like to investigate how our model relates to concrete architectures for NLP, such as the Tensor Product Recurrent Networks (TPRNs) of [39]. Another machine learning model which would fit our approach is that of [28]: this method uses gradient descent over a landscape of complex-valued encoding matrices `E` and `R` to predict missing values in a knowledge graph K. Linguistically, the passage to the alternative semi-ring of complex numbers would allow to give two distinct encodings for entities: in our model the object-pronoun "them" would be modelled as the complex conjugate of the subject "they". Modelling the active-to-passive switch with diagrams of complex matrices, as well as exploring potential implementations making use of quantum resources is left to later work.

In this paper we looked only at one discourse d at a time, assuming a global matching process $!\mu$ and entity-store $!E$. Using methods from sheaf theory would allow to study the passage from local to global ambiguity and investigate the notion of *contextuality* common to natural language, quantum mechanics and database theory [48]. This finer-grained analysis of the resolution process, as well as the implementation of other DRS operations, such as disjunction, implication and quantification, in the category $\texttt{Mat}_{\mathcal{S}}$ is left for future work.

References

[1] Jim Lambek. *From Word to Sentence: A Computational Algebraic Approach to Grammar*. Open access publications. Polimetrica, 2008.

[2] Noam Chomsky. *Syntactic Structures*. The Hague: Mouton and Co., 1957.

[3] Richard Montague. "The Proper Treatment of Quantification in Ordinary English". In: *Approaches to Natural Language*. Ed. by K. J. J. Hintikka, J. Moravcsic, and P. Suppes. Dordrecht: Reidel, 1973, pp. 221–242.

[4] Bob Coecke et al. "Generalized relations in linguistics & cognition". In: *Theoretical Computer Science* (2018). URL: http://www.sciencedirect.com/science/article/pii/S0304397518301476.

[5] Alexis Toumi, Julian Gutierrez, and Michael Wooldridge. "A Tool for the Automated Verification of Nash Equilibria in Concurrent Games". In: *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings*. Ed. by Martin Leucker, Camilo Rueda, and Frank D. Valencia. Vol. 9399. Lecture Notes in Computer Science. Springer, 2015, pp. 583–594. URL: https://doi.org/10.1007/978-3-319-25150-9_34.

[6] Michael Wooldridge et al. "Rational Verification: From Model Checking to Equilibrium Checking". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 4184–4191. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12268.

[7] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[8] S.M. Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1998. URL: https://books.google.fr/books?id=eBvhyc4z8HQC.

[9] Matthew Kisner, ed. *Spinoza: Ethics*. Cambridge University Press, 2018. URL: https://doi.org/10.1017/9781107706972.

[10] L. Wittgenstein. "Tractatus Logico-Philosophicus". In: *London: Routledge, 1981* (1922). Ed. by D.F.Pears. URL: http://scholar.google.de/scholar.bib?q=info:1G2GoIkyCZIJ:scholar.google.com/&output=citation&hl=de&ct=citation&cd=0.

[11] Robin Piedeleu. "Ambiguity in Categorical Models of Meaning". MA thesis. University of Oxford, 2014.

[12] Wojciech Buszkowski. "Lambek grammars based on pregroups". In: *Logical Aspects of Computational Linguistics* LNAI.2099 (2001), pp. 95–109.

[13] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. "A Compositional Distributional Model of Meaning". In: *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)* (2008), pp. 133–140.

[14] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. "Mathematical foundations for a compositional distributional model of meaning". In: *A Festschrift for Jim Lambek*. Ed. by J. van Benthem, M. Moortgat, and W. Buszkowski. Vol. 36. Linguistic Analysis. arxiv:1003.4394. 2010, pp. 345–384.

[15] Stefano Gogioso. *Fantastic Quantum Theories and Where to Find Them*. 2017. eprint: `arXiv:1703.10576`.

[16] Anne Preller and Joachim Lambek. "Free compact 2-categories". In: *Mathematical Structures in Computer Science* 17.2 (2007), pp. 309–340. URL: `https://doi.org/10.1017/S0960129506005901`.

[17] C. L. Hamblin. "Questions in Montague English". In: *Foundations of Language* 10.1 (1973), pp. 41–53. URL: `http://www.jstor.org/stable/25000703`.

[18] Samson Abramsky and Radha Jagadeesan. "Games and Full Completeness for Multiplicative Linear Logic". In: *CoRR* abs/1311.6057 (2013). arXiv: `1311.6057`. URL: `http://arxiv.org/abs/1311.6057`.

[19] F. W. Lawvere. "Display of graphics and their applications, as exemplified by 2-categories and the Hegelian "taco"". In: *Proceedings of the first international conference on algebraic methodology and software technology University of Iowa, May 22-24 1989, Iowa City, pp.* (1989), pp. 51–74.

[20] P. Selinger. "A Survey of Graphical Languages for Monoidal Categories". In: *New Structures for Physics*. Springer Berlin Heidelberg, 2010, pp. 289–355. URL: `https://doi.org/10.1007/978-3-642-12821-9_4`.

[21] Amos Tversky. "Features of similarity". In: *Psychological Review* 84.4 (1977), pp. 327–352.

[22] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017. URL: `http://dx.doi.org/10.1017/9781316219317`.

[23] Stephen Clark et al. "A quantum teleportation inspired algorithm produces sentence meaning from word meaning and grammatical structure". In: *Malaysian Journal of Mathematical Sciences* 8 (2014). arXiv:1305.0556, pp. 15–25.

[24] Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. "The Frobenius anatomy of word meanings I: subject and object relative pronouns". In: *Journal of Logic and Computation* 23 (2013). arXiv:1404.5278, pp. 1293–1317.

[25] Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. "The Frobenius anatomy of word meanings II: possessive relative pronouns". In: *Journal of Logic and Computation* abs/1406.4690 (2014), exu027. URL: `http://arxiv.org/abs/1406.4690`.

[26] Dimitri Kartsaklis et al. "Reasoning about Meaning in Natural Language with Compact Closed Categories and Frobenius Algebras". In: *CoRR* abs/1401.5980 (2014). URL: `http://arxiv.org/abs/1401.5980`.

[27] Bob Coecke and Ross Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". In: *New Journal of Physics* 13 (2011). arXiv:quant-ph/09064725, p. 043016.

[28] Théo Trouillon et al. "Knowledge graph completion via complex tensor factorization". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4735–4772. eprint: `arXiv:1702.06879`.

[29] S. C. Kleene. "Representation of events in nerve nets and finite automata". In: *Automata Studies*. Ed. by Claude Shannon and John McCarthy. Princeton, NJ: Princeton University Press, 1956, pp. 3–41.

[30] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Vol. 42. Studies in Linguistics and Philosophy. Dordrecht: Springer, 1993.

[31] Samson Abramsky and Mehrnoosh Sadrzadeh. "Semantic unification". In: *Categories and Types in Logic, Language, and Physics*. Springer, 2014, pp. 1–13. URL: `http://arxiv.org/abs/1403.3351`.

[32] Giorgio Stefanoni, Boris Motik, and Egor V. Kostylev. "Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation". In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. Ed. by Pierre-Antoine Champin et al. ACM, 2018, pp. 1043–1052. URL: `http://doi.acm.org/10.1145/3178876.3186003`.

[33] Adelin Travers. "Sheaf Theoretic Semantics for Vector Space Models of Natural Language". MA thesis. University of Oxford, 2015.

[34] Peter T Johnstone. *Sketches of an elephant: a Topos theory compendium*. Oxford logic guides. New York, NY: Oxford Univ. Press, 2002. URL: `https://cds.cern.ch/record/592033`.

[35] Jules Hedges and Mehrnoosh Sadrzadeh. *A Generalised Quantifier Theory of Natural Language in Categorical Compositional Distributional Semantics with Bialgebras*. 2016. eprint: `arXiv:1602.01635`.

[36] Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. "Interacting Bialgebras Are Frobenius". In: 2014. URL: `http://dx.doi.org/10.1007/978-3-642-54830-7_23;http://dblp.uni-trier.de/rec/bib/conf/fossacs/BonchiSZ14`.

[37] Ross Duncan and Kevin Dunne. "Interacting Frobenius Algebras are Hopf". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*. 2016, pp. 535–544. URL: `http://doi.acm.org/10.1145/2933575.2934550`.

[38] Filippo Bonchi et al. "Rewriting modulo symmetric monoidal structure". In: (2016). eprint: `arXiv:1602.06771`.

[39] Hamid Palangi et al. "Question-Answering with Grammatically-Interpretable Representations". In: *arXiv preprint arXiv:1705.08432* (2017). eprint: `arXiv:1705.08432`.

[40] Brendan Fong, Paweł Sobociński, and Paolo Rapisarda. "A categorical approach to open and interconnected dynamical systems". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*. 2016, pp. 495–504. URL: http://doi.acm.org/10.1145/2933575.2934556.

[41] Brendan Fong, David I. Spivak, and Rémy Tuyéras. *Backprop as Functor: A compositional perspective on supervised learning*. 2017. eprint: arXiv:1711.10455.

[42] Özlem Uzuner et al. "Evaluating the state of the art in coreference resolution for electronic medical records". In: *Journal of the American Medical Informatics Association : JAMIA* 19 5 (2012), pp. 786–91.

[43] J. D. Pettigrew and S. M. Miller. "A 'sticky' interhemispheric switch in bipolar disorder?" In: *Proceedings of the Royal Society B: Biological Sciences* 265.1411 (1998), pp. 2141–2148. URL: https://doi.org/10.1098/rspb.1998.0551.

[44] Paul Smolensky. "Connectionist AI, symbolic AI, and the brain". In: *Artif. Intell. Rev.* 1.2 (1987), pp. 95–109. URL: https://doi.org/10.1007/BF00130011.

[45] Bob Coecke and Eric Oliver Paquette. "Categories for the practicing physicist". In: *New Structures for Physics*. Ed. by B. Coecke. Lecture Notes in Physics. arXiv:0905.3010. Springer, 2011, pp. 167–271.

[46] Antonin Delpeuch and Anne Preller. "From Natural Language to RDF Graphs with Pregroups". In: *EACL'2014: 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden: EACL, Apr. 2014, pp. 55–62. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-00992381.

[47] Edward Grefenstette and Mehrnoosh Sadrzadeh. "Experimental Support for a Categorical Compositional Distributional Model of Meaning". In: *The 2014 Conference on Empirical Methods on Natural Language Processing.* arXiv:1106.4058. 2011, pp. 1394–1404.

[48] Samson Abramsky. "Contextual semantics: From quantum mechanics to logic, databases, constraints, and complexity". In: *Contextuality from Quantum Physics to Psychology*. World Scientific, 2016, pp. 23–50.