Anaphora and Ellipsis in Lambek Calculus with a Relevant Modality: Syntax and Semantics

Lachlan McPheat[1], Gijs Wijnholds[2], Mehrnoosh Sadrzadeh[1], Adriana Correia[2], and Alexis Toumi[3]

[1]*University College London, UK*
[2]*Utrecht University, NL*
[3]*Oxford University, UK*

Abstract

Lambek calculus with a relevant modality $!\mathbf{L}^*$ of (Kanovich et al., 2016) syntactically resolves parasitic gaps in natural language. It resembles the Lambek calculus with anaphora **LA** of (Jäger, 1998) and the Lambek calculus with controlled contraction $\mathbf{L}_\diamond$ of (Wijnholds and Sadrzadeh, 2019b) which deal with anaphora and ellipsis. What all these calculi add to Lambek calculus is a copying and moving behaviour. Distributional semantics is a subfield of Natural Language Processing that uses vector space semantics for words via co-occurrence statistics in large corpora of data. Compositional vector space semantics for Lambek Calculi are obtained via the *DisCoCat* models (Coecke et al., 2010). **LA** does not have a vector space semantics and the semantics of $\mathbf{L}_\diamond$ is not compositional. Previously, we developed a DisCoCat semantics for $!\mathbf{L}^*$ and focused on the parasitic gap applications. In this paper, we use the vector space instance of that general semantics and show how one can also interpret anaphora, ellipsis, and for the first time derive the sloppy vs strict vector readings of ambiguous anaphora with ellipsis cases. The base of our semantics is tensor algebras and their finite dimensional variants: the Fermionic Fock spaces of Quantum Mechanics. We implement our model and experiment with the ellipsis disambiguation task of (Wijnholds and Sadrzadeh, 2019a).

Keywords: *Lambek Calculus Relevant Modality Controlled Contraction Anaphora Ellipsis Strict vs Sloppy Vector Space Semantics Tensor Algebras Fock Spaces.*

1. Introduction

Inspired by Linear Logic's !, extensions of Lambek calculi with exponentials were introduced in the 90's in (Morrill et al., 1990; Barry et al., 1995) for the purpose of deriving phenomena with medial extraction, as in relative clauses such as "*book which fell*", "*book which John read*", and "*book which John read yesterday*", iterated coordination, as in "*John Bill Mary and Suzy*", parasitic gaps, as in "*the paper which Suzy filed without reading*", and cases were there is a lexical item whose arguments optionally contains a gap, as in the lexical item '*too*' in clauses such as "*too boring for one to follow*". For a modern exposition and review of these systems, see (Moot and Retoré, 2012). Similar extensions, with limited copying, and combinations of permutation and associativity, rather than just permutation, were used to model coreference phenomena such as anaphora and ellipsis in (Jäger, 1998; Morrill and Merenciano Saladrigas, 1996). These systems were recently revisited in (Morrill and Valentín, 2016, 2015) and also independently in (Wijnholds and Sadrzadeh, 2018, 2019b). It was the former set that inspired the logic $!\mathbf{L}^*$: Lambek calculus with a Relevant Modality, developed in (Kanovich et al., 2016) and used in this paper.

In previous work (McPheat et al., 2021), we developed a categorical semantics for $!\mathbf{L}^*$ via coalgebra modalities of Differential Categories (Blute et al., 2006) and showed how at least two different interpretations of ! in the category of finite dimensional vector spaces and linear maps provide vector space instantiations for the abstract categorical semantics. Our focus in that paper, was to develop a vector space semantics for parasitic gaps, the motivating example of (Kanovich et al., 2016). In this paper, we briefly re-introduce the vector space semantics of $!\mathbf{L}^*$, making use of the notion of tensor algebras and their finite variants: Fermionic Fock spaces of Quantum Mechanics. Then, following previous work (Wijnholds and Sadrzadeh, 2018, 2019b) and

the work of Jäger (Jäger, 1998, 2006), we apply the setting to obtain tensor algebraic interpretations of examples with anaphora, ellipsis, and combinations of the two. By so doing, we overcome two different weaknesses of the above mentioned previous approaches: (1) developing a compositional vector space semantics for a Lambek calculus that can deal with anaphora and ellipsis, whereas the work of (Jäger, 1998, 2006) did not (it only had a lambda calculus semantics), and (2) being able to obtain two different vector semantics for the sloppy vs strict readings of ambiguous anaphora with ellipsis cases, whereas the work in (Wijnholds and Sadrzadeh, 2018, 2019b) could not. The contributions of this and previous work (McPheat et al., 2021) are thus two-fold: first we have developed a direct vector space semantics for an extension of Lambek calculus with limited contraction. The work done in (Wijnholds and Sadrzadeh, 2018, 2019b) does develop a vector space semantics for Jäger's multi modal calculus (Jäger, 1998), but via a two-step semantics and by going through a dynamic logic of vectors (Muskens and Sadrzadeh, 2019). Turning this semantics into a single-step one faced the challenge of discovering a linear copying map. We have done this for $!\mathbf{L}^*$ through the use of Fermionic tensor algebras and in effect infinitely many linear copying maps. Second, the semantics proposed in (Wijnholds and Sadrzadeh, 2018, 2019b), could not distinguish between the sloppy and strict readings of anaphora with ellipsis, and here we are able to do so. We implemented our vector space semantics on distributional data and experimented with the ellipsis disambiguation dataset of (Wijnholds and Sadrzadeh, 2019a). The results show that our linear copying operations do as well as the full non linear copying and also outperform non compositional and additive baselines.

Finally, the use of exponentials may not be necessary when dealing with the phenomena of parasitic gaps, anaphora and ellipsis; milder modal Lambek calculi with structural control, as defined in (Moortgat, 1996), or displacement calculi (Morrill et al., 2011), can also be used as a base as well, e.g. see the recent proposal of (Sadrzadeh et al., 2020), which uses modal Lambek calculi with a Frobenius algebraic semantics for parasitic gaps. Finding formal connections with these is a possible future direction.

2. Background

We introduce the relevant background and context for this paper, along with the definitions and examples of the linguistic phenomena we are considering (i.e. anaphora and ellipsis), the logic we employ, and previous examples of similar logics modelling anaphora and ellipsis.

2.1 Anaphora and Ellipsis

Anaphora and Ellipsis are instances of **coreference** in natural language, which is a phenomenon where two distinct linguistic expressions refer to the same entity (Chomsky, 1981; Bach, 2008). Resolving coreference is the task of identifying what entity is being referred to.

Anaphora is a phenomenon where expressions called *anaphors* receive their meaning from a previously mentioned word or phrase, for instance pronouns like '*He*' in '*John sleeps. He snores.*', where clearly, '*John*' and '*He*' have the same referent[1].

Ellipsis is a more general phenomenon, where broadly speaking, the meanings of entire phrases are referred to by combinations of other words, sometimes called *ellipsis markers*. In some cases ellipsis markers are dropped and a phrase is referred to by the empty expression. For instance, in "*John plays guitar. Mary does too.*", the combination of words '*does too*' is an ellipsis marker that refers to the phrase '*plays guitar*'. A complete list of types of ellipsis is not yet agreed upon in the linguistics community and different theories have different lists. The above example is sometimes called *verb-phrase ellipsis* (VP-ellipsis) and is the focus of this paper.

Anaphora with ellipsis is an utterance containing *both* anaphora and ellipsis. The example we will use throughout this paper is "*John likes his code. Bill does too*". Here the pronoun '*his*' is an anaphor, and the verb phrase '*likes his code*' is being elliptically referred to by the marker '*does too*'. Note how we can read this example in two different ways by choosing which order to resolve the anaphora or the ellipsis. The simpler way is called

[1] One may also consider **cataphora**, which are phenomena where expressions called *cataphors* receive their meanings from a word or phrase following them.

the **strict** reading, and has the meaning "*John likes John's code. Bill likes John's code*", which is achieved by first resolving the anaphor with *John*, and then the ellipsis with the verb-phrase '*likes John's code*'. The second reading is called the **sloppy** reading, and is achieved by first resolving the ellipsis with '*likes his code*', and then resolve the two distinct anaphors '*his*' with '*John*' and '*Bill*' separately, giving us the meaning "*John likes John's code. Bill likes Bill's code*". In section 3 we show how to distinguish between the two readings syntactically, and in section 5 we derive the vector semantics of the two readings.

2.2 Type Logical Grammars and Anaphora

Earlier works have modelled the syntactic structures that arise from anaphora and ellipsis as derivations in type-logical grammars (TLGs) in efforts to model these phenomena in the same formalism as TLGs model other phenomena of language. An early instance of this work comes in a paper by Jäger (Jäger, 1998), where the author defines a multimodal TLG which syntactically resolves anaphora and ellipsis. Jäger follows up this idea in his book (Jäger, 2006), where he suggests the more concise idea of using TLGs with controlled contraction and permutation. However, neither in this book nor in (Jäger, 1998) can you find any mention of a vector space semantics, as this was not the goal of the author. A standard lambda calculus semantics was however developed.

A vector space semantics for VP-ellipsis is developed in (Wijnholds and Sadrzadeh, 2018), which employs a TLG with controlled contraction and permutation, inspired by (Jäger, 2006), and defines vector space interpretations for anaphora with ellipsis examples. This semantics, however, uses a non-linear copying operation, and fails to distinguish between strict and sloppy readings. Both of the issues with this semantics are addressed in this paper.

2.3 Lambek Calculus with a Relevant Modality

Lambek calculus with a Relevant Modality, denoted by $!\mathbf{L}^*$, is a variation of Lambek calculus, $\mathbf{L}$ where we may use empty sequents (denoted by $*$),

$$\frac{}{A \to A} \qquad \frac{\Gamma \to A \quad \Delta_1, B, \Delta_2 \to C}{\Delta_1, B/A, \Gamma, \Delta_2 \to C}\,(/L) \qquad \frac{\Gamma, A \to B}{\Gamma \to B/A}\,(/R)$$

$$\frac{\Gamma \to A \quad \Delta_1, B, \Delta_2 \to C}{\Delta_1, \Gamma, A\backslash B, \Delta_2 \to C}\,(\backslash L) \qquad \frac{A, \Gamma \to B}{\Gamma \to A\backslash B}\,(\backslash R)$$

$$\frac{\Gamma_1, A, \Gamma_2 \to C}{\Gamma_1, !A, \Gamma_2 \to C}\,(!L) \qquad \frac{!A_1, \ldots, !A_n \to B}{!A_1, \ldots, !A_n \to !B}\,(!R)$$

$$\frac{\Delta_1, !A, \Gamma, \Delta_2 \to C}{\Delta_1, \Gamma, !A, \Delta_2 \to C}\,(perm_1) \qquad \frac{\Delta_1, \Gamma, !A, \Delta_2 \to C}{\Delta_1, !A, \Gamma, \Delta_2 \to C}\,(perm_2) \qquad \frac{\Delta_1, !A, !A, \Delta_2 \to C}{\Delta_1, !A, \Delta_2 \to C}\,(contr)$$

Table 1. Rules of the $!\mathbf{L}^*$ calculus.

and which has a (relevant) modality, !, which denotes when a formula is not resource sensitive. We consider the calculus $!\mathbf{L}^*$ over the set of primitive types $\{N, S\}$ and logical connectives $\{/, \backslash, !\}$. The primitive types represent the basic grammatical types of "noun" and "sentence". With these types and the connectives we generate the full set of types[2] according to the following BNF:

$$\mathrm{Typ}_{!\mathbf{L}^*} ::= * \mid N \mid S \mid A, B \mid A/B \mid A\backslash B \mid !A.$$

The rules of $!\mathbf{L}^*$ are presented in the sequent calculus system of table 1. Sequents are written as $\Gamma \to A$, where Γ is a finite (possibly empty) list of formulas $\{A_1, \ldots, A_n\}$, and A is a single formula. We refer to the set of formulas of $!\mathbf{L}^*$ as $\mathrm{Typ}_{!\mathbf{L}^*}$.

3. Resolving Anaphora and Ellipsis in $!\mathbf{L}^*$

We follow (Wijnholds and Sadrzadeh, 2018; Jäger, 1998, 2006) in how we resolve anaphora and ellipsis syntactically using type logical grammars. Given a word or compound w with a $!\mathbf{L}^*$-type C whose meaning comes from a reference w' (necessarily of the same type) e.g. a pronoun (anaphor) or an ellipsis marker (eg. '*does-too*'), we first change the assignments of types of w to $(!C\backslash C)$. We then assign the reference word, w' the type $!C$, apply contraction

[2] We use the terms 'formula' and 'type' interchangeably.

to this type, obtain the formula $!C, !C$, and then move one of the copies to the reference type $(!C\backslash C)$, to which we then apply the $(\backslash L)$-rule. This procedure provides us with a proof tree as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{!C \longrightarrow !C} \quad \dfrac{\vdots}{\Gamma_1, !C, \Gamma_2, C, \Gamma_3 \longrightarrow B}}{\Gamma_1, !C, \Gamma_2, !C, !C\backslash C, \Gamma_3 \longrightarrow B}\,(\backslash L)}{\Gamma_1, !C, !C, \Gamma_2, !C\backslash C, \Gamma_3 \longrightarrow B}\,(perm_2)}{\Gamma_1, !C, \Gamma_2, !C\backslash C, \Gamma_3 \longrightarrow B}\,(contr)$$

where $\Gamma_1, \Gamma_2, \Gamma_3$ are (possibly empty) contexts surrounding the coreference.

For a concrete example of this procedure, consider: "*John sleeps. He snores*". Clearly, '*He*' refers to '*John*', but since'*John*' is already being used as the argument of '*sleeps*', it is not possible for '*He*' to use it as well in classical Lambek calculus. Thus we instead wish to "copy" the meaning of '*John*' using the $(contr)$-rule of $!\mathbf{L}^*$, and move one copy to where '*He*' can use it, using the $(perm_2)$-rule of $!\mathbf{L}^*$.

In what follows, we present examples of anaphora, ellipsis, and the combination of the two. These examples involve some seemingly intimidating derivations, of which we want the reader to be wary. We will verbally describe what is happening in each of these derivations, and later in section 5 we will be able to view the derivations as string diagrams, which are far more intuitive.

3.1 Anaphora

Following (Wijnholds and Sadrzadeh, 2018), we work with the example "*John sleeps. He snores*" and the following type assignment [3] :

$$\{(\text{John}, !N), (\text{sleeps}, N\backslash S), (\text{snores}, N\backslash S), (\text{he}, !N\backslash N)\}$$

which provides us with the derivation proof tree in figure 1. We note that the

[3] Note that one can always "ignore" the ! of a type by applying the $(!L)$-rule. This means that in principle one may wish to consider every noun and every verb phrase as having type !, since every noun/VP may be referred to at some point. We choose to exclude this typing, as it adds more steps to the derivations.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{N \to N} \quad \dfrac{\dfrac{}{!N \to !N} \quad \dfrac{\dfrac{}{N \to N} \quad \dfrac{}{S, S \to S, S}\,(\backslash L)}{S, N, N\backslash S \to S, S}\,(\backslash L)}{S, !N, !N\backslash N, N\backslash S \to S, S}\,(\backslash L)}{N, N\backslash S, !N, !N\backslash N, N\backslash S \to S, S}\,(!L)}{!N, N\backslash S, !N, !N\backslash N, N\backslash S \to S, S}\,(perm_2)}{!N, !N, N\backslash S, !N\backslash N, N\backslash S \to S, S}\,(contr)}{!N, N\backslash S, !N\backslash N, N\backslash S \to S, S}$$

Figure 1. Anaphora Derivation

very first rule applied (reading the proof from bottom to top) is the contraction which copies the type $!N$ of '*John*' providing us with $!N, !N$. We then permute the rightmost copy to the immediate left hand side of the type of '*He*', i.e. the formula $!N\backslash N$, and then unify the types of '*John*' and '*He*' using the $(\backslash L)$-rule.

3.2 Ellipsis

For ellipsis, consider the simple example "*John plays guitar. Mary does too*" and the following type assignment

$$\{(\text{John}, N), (\text{plays}, !(N\backslash S)/N), (\text{guitar}, N), (\text{Mary}, N), (\text{does-too}, (!(N\backslash S))\backslash(N\backslash S))\}\ .$$

In order to resolve the ellipsis in this example, all we need to do is to show that the sequent

$$N, !(N\backslash S)/N, N, N, (!(N\backslash S))\backslash(N\backslash S) \longrightarrow S, S$$

is derivable in $!\mathbf{L}^*$. This is done in figure 2. The first application of a rule is the $(/L)$-rule, which applies '*plays*' to its subject '*guitar*'. After that application we proceed as we do in the anaphora example, the next rule being contraction, but this time to the formula of the entire verb phrase '*plays guitar*', that is, the formula $!(N\backslash S)$. One of the copies is then moved to the ellipsis site "*does too*", and identified using the $(\backslash L)$-rule.

3.3 Anaphora with Ellipsis

We now combine anaphora and ellipsis and work with the example "*John likes his code, Bill does too.*", where we have an anaphor '*his*' and ellipsis

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{N \longrightarrow N} \quad \dfrac{\dfrac{}{N \longrightarrow N} \quad \dfrac{}{S, S \longrightarrow S, S}\,(\backslash L)}{S, N, N\backslash S \longrightarrow S, S}\,(\backslash L)}{N, N\backslash S, N, N\backslash S \longrightarrow S, S} \quad \dfrac{}{!(N\backslash S) \longrightarrow !(N\backslash S)}}{N, !(N\backslash S), N, N\backslash S \longrightarrow S, S}\,(!L)}{N, !(N\backslash S), N, !(N\backslash S), (!(N\backslash S))\backslash(N\backslash S) \longrightarrow S, S}\,(\backslash L)}{N, !(N\backslash S), !(N\backslash S), N, (!(N\backslash S))\backslash(N\backslash S) \longrightarrow S, S}\,(perm_2)}{N, !(N\backslash S), N, (!(N\backslash S))\backslash(N\backslash S) \longrightarrow S, S}\,(contr) \quad \dfrac{}{N \longrightarrow N}}{N, !(N\backslash S)/N, N, N, (!(N\backslash S))\backslash(N\backslash S) \longrightarrow S, S}\,(/L)$$

Figure 2. Ellipsis Derivation

marker '*does too*'. Clearly, the derivation of the sloppy reading will involve more resolutions than the strict reading, so one would expect this derivation to be more complex than a strict one, which is indeed the case.

Below, we present the derivations of the strict and sloppy readings in full, using the following type-dictionary for both examples:

$$\{(\text{John} :!N), (\text{likes} :!((!(!N\backslash S))/N), (\text{his} : (!(!N\backslash N))/N), (\text{code} : N),$$
$$(\text{Bill} :!N), (\text{does too} : (!(!N\backslash S))\backslash(!N\backslash S))\}$$

Strict Reading.

The derivation of the strict reading is presented in figure 3, where we can see only two uses of the (*contr*)-rule. We expect only two uses of (*contr*) since we are first resolving the anaphor '*his*' with '*John*', and then the verb-phrase '*likes John's code*' with the ellipsis site '*does-too*'.

Sloppy Reading.

The derivation of the sloppy reading is presented in figure 4, and is clearly more complex than the strict reading in figure 3. Looking at the right hand side of the sloppy derivation we see four instances of the (*contr*)-rule, which correspond to the four different coreferences we are resolving. The Gentzen presentation really obfuscates the coreference resolution visually, and one is forced to read these derivations very closely to understand what is happening.

$$\dfrac{\overline{!N \to !N} \qquad \overline{S, S \to S, S}}{S, !N, !N\backslash S \to S, S}\ (\backslash L)$$

$$\dfrac{\overline{!(!N\backslash S) \to !(!N\backslash S)} \qquad S, !N, !N\backslash S \to S, S}{S, !N, !(!N\backslash S), (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (\backslash L)$$

$$\dfrac{\overline{!N \to !N} \qquad S, !N, !(!N\backslash S), (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, !N\backslash S, !N, !(!N\backslash S), (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (\backslash L)$$

$$\dfrac{!N, !N\backslash S, !N, !(!N\backslash S), (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, !(!N\backslash S), !N, !(!N\backslash S), (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (!L)$$

$$\dfrac{!N, !(!N\backslash S), !N, !(!N\backslash S), (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, !(!N\backslash S), !(!N\backslash S), !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (perm_2)$$

$$\dfrac{!N, !(!N\backslash S), !(!N\backslash S), !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, !(!N\backslash S), !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (contr)$$

$$\dfrac{\overline{N \to N} \qquad !N, !(!N\backslash S), !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, (!(!N\backslash S))/N, N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (/L)$$

$$\dfrac{\overline{!N \to !N} \qquad !N, (!(!N\backslash S))/N, N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, (!(!N\backslash S))/N, !N, !N\backslash N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (\backslash L)$$

$$\dfrac{!N, (!(!N\backslash S))/N, !N, !N\backslash N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, !N, (!(!N\backslash S))/N, !N\backslash N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (perm_2)$$

$$\dfrac{!N, !N, (!(!N\backslash S))/N, !N\backslash N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, (!(!N\backslash S))/N, !N\backslash N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (contr)$$

$$\dfrac{!N, (!(!N\backslash S))/N, !N\backslash N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, (!(!N\backslash S))/N, !(!N\backslash N), !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (!L)$$

$$\dfrac{\overline{N \to N} \qquad !N, (!(!N\backslash S))/N, !(!N\backslash N), !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, (!(!N\backslash S))/N, (!(!N\backslash N))/N, N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (/L)$$

$$\dfrac{!N, (!(!N\backslash S))/N, (!(!N\backslash N))/N, N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}{!N, !((!(!N\backslash S))/N), (!(!N\backslash N))/N, N, !N, (!(!N\backslash S))\backslash(!N\backslash S) \to S, S}\ (!L)$$

Figure 3. The derivation of the strict reading: 'John likes John's code, Bill likes John's code', in $!\mathbf{L}^*$

It is far easier to read the string-diagrams attached to these proofs which we demonstrate later in section 5

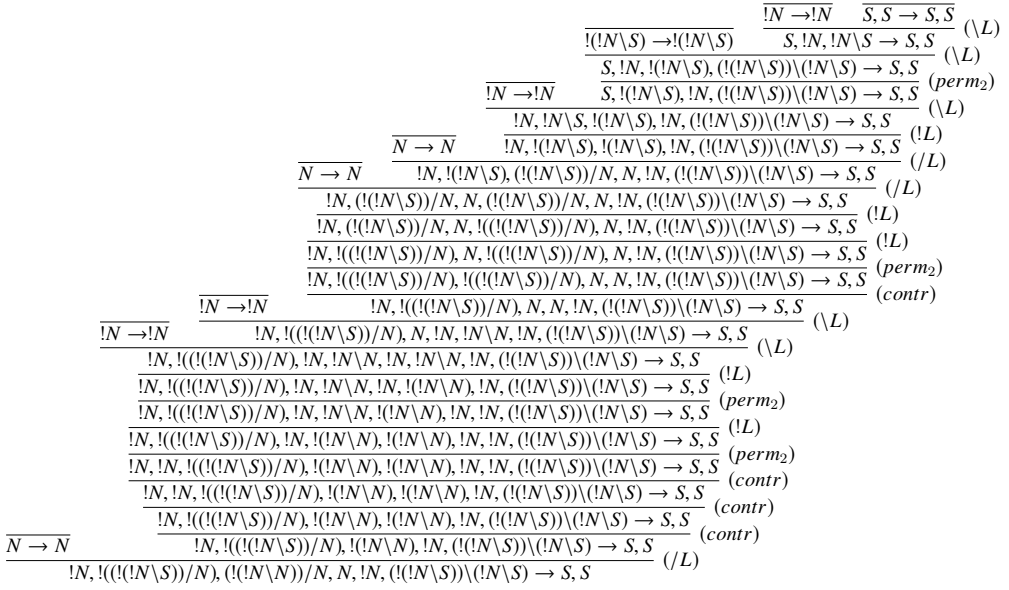


Figure 4. The derivation of the sloppy reading: 'John likes John's code, Bill likes Bill's code, in $!\mathbf{L}^*$

Despite the difficulty of gaining quick qualitative information from the derivations, we have already shown that we can derive anaphora, ellipsis and anaphora with ellipsis in both of its possible strict and sloppy readings. The ability of our calculus to distinguish between the strict and sloppy readings is a desirable property, and more importantly, this distinction carries over into the vector space semantics, as we demonstrate next.

4. Vector Space Semantics of $!\mathbf{L}^*$

In this section we introduce the vector space semantics of $!\mathbf{L}^*$, as defined in our previous work (McPheat et al., 2021). We briefly summarise the categorical semantics in 4.4, but refer the reader to (McPheat et al., 2021) for

full technical details. We will point out any practical use of the categorical semantics in the examples of this paper, but please note that it is also not necessary to do so; the vector space semantics provides what we need for our linguistic motivations. We also introduce the diagrammatic calculus of (McPheat et al., 2021) with the exception that here we omit the use of thick strings as to minimise the amount of string diagrammatic machinery present in this paper, as the diagrams are not the main contribution. This will be in 4.2; the diagrammatic calculus is as a didactic tool to let the reader visualise the derivations in section 3. The semantics of these derivations is presented in 5. Before we define the semantics, we first need to understand tensor algebras, and comultiplications on them as these structures will allow us to define semantics for ! and the (*contr*)-rule.

4.1 Tensor Algebras

We recall the definition of a tensor algebra, and show how to construct a Fermionic Fock space from it. Dualised versions of these constructions were used to interpret ! of full linear logic (Blute et al., 1994), and then for $!\mathbf{L}^*$ (McPheat et al., 2021). We briefly introduce tensor algebras, Fermionic Fock spaces and their duals below, and refer to (McPheat et al., 2021) for further details on these constructions.

Definition 1 *Given a vector space V, the* ***tensor algebra*** *TV is defined as*

$$TV := \bigoplus_{n\geq 0} V^{\otimes n} = \mathbb{R} \oplus V \oplus (V \otimes V) \oplus (V \otimes V \otimes V) \oplus \cdots .$$

*We call the terms $V^{\otimes n}$ the n-**th layer** of TV.*

There is a monoid structure on TV given by a multiplication $m : TV \otimes TV \to TV$ defined by layer-wise concatenation i.e.

$$m((v_1 \otimes \cdots \otimes v_n) \otimes (w_1 \otimes \cdots \otimes w_m)) := v_1 \otimes \cdots \otimes v_n \otimes w_1 \otimes \cdots \otimes w_m,$$

and a unit $u : \mathbb{R} \longrightarrow TV$ given by $u(1) := 1$.

It is in fact the case that T is a free functor $\mathbf{Vect}_{\mathbb{R}} \longrightarrow \mathbf{Alg}_{\mathbb{R}}$. Thus defining a monad on $\mathbf{Vect}_{\mathbb{R}}$ by composing with the forgetful functor $\mathbf{Alg}_{\mathbb{R}} \longrightarrow \mathbf{Vect}_{\mathbb{R}}$, where $\mathbf{Alg}_{\mathbb{R}}$ is the category of real associative algebras.

Clearly, TV is infinite-dimensional for any nonzero vector space V and an inappropriate choice for our semantics of !. Instead, we will interpret ! using Fermionic Fock Spaces, which have similar properties as Tensor algebras, but can be made finite dimensional. To define a Fermionic Fock Space we will need alternating tensor products.

Definition 2 *Given a (real) vector space V with basis $(e_i)_{i\in I}$ respectively, we define the n-**fold alternating tensor product** $V^{\wedge n}$ of V as:*

$$\overbrace{V \wedge V \wedge \cdots \wedge V}^{n\text{-times}} := V^{\otimes n}/U,$$

where U is the vector space spanned by vectors of the form

$$(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_n}) - \operatorname{sgn}(\sigma)(e_{\sigma(i_1)} \otimes e_{\sigma(i_2)} \otimes \cdots \otimes e_{\sigma(i_n)}).$$

In the above, $i_j \in I$ for each $1 \leq j \leq n$ and each permuation σ on n-symbols. Vectors in $V \wedge V$ are linear combinations of equivalence classes of simple tensors, denoted $e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n}$. The key point in this definition is that the vector $e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n}$ is equal to the vector $\operatorname{sgn}(\sigma)(e_{\sigma(i_1)} \wedge e_{\sigma(i_2)} \wedge \cdots \wedge e_{\sigma(i_n)})$. For instance, if $n = 2$, we have that $e_1 \wedge e_2 = -e_2 \wedge e_1$, since the sign of the permutation (12) is -1.

Note

Basis vectors in $V^{\wedge n}$ with repeated factors are zero. Consider a vector $e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n}$ where WLOG the first two factors are the same: $e_{i_1} = e_{i_2}$. Thus we have the equality

$$\begin{aligned} e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n} &= \operatorname{sgn}(12)(e_{(12)i_1} \wedge e_{(12)i_2} \wedge \cdots \wedge e_{(12)i_n}) \\ &= -e_{i_2} \wedge e_{i_1} \wedge \cdots \wedge e_{i_n} \\ &= -e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n}. \end{aligned}$$

However if a vector equals its negative, it must have been zero to begin with, thus confirming that $e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n} = 0$.

With the definition of alternating tensor products under our belts, we can quickly define the Fermionic Fock space construction in close analogy to the tensor algebra definition.

Definition 3 *Given a vector space V its **Fermionic Fock space** is the vector space*

$$\mathcal{F}V := \bigoplus_{n \geq 0} V^{\wedge n} = \mathbb{R} \oplus V \oplus (V \wedge V) \oplus (V \wedge V \wedge V) \oplus \cdots .$$

The space $\mathcal{F}V$ also has a monoidal structure, defined exactly as we did for TV, the only difference being that the multiplication on $\mathcal{F}V$ is alternating. $\mathcal{F}V$ is also known as the Grassmanian algebra of V, or the antisymmetric tensor algebra of V.

$\mathcal{F}$ is also a free functor, this time of the form $\mathbf{Vect}_{\mathbb{R}} \longrightarrow \mathbf{AAlg}_{\mathbb{R}}$, where $\mathbf{AAlg}_{\mathbb{R}}$ is the category of alternating real associative algebras.

We recall the well-known fact that given a finite dimensional vector space V, we can easily see that $\mathcal{F}V$ is also finite dimensional by applying the pigeonhole principle to the dimension of V and the number of layers in $\mathcal{F}V$. Concretely, note that for any $n > \dim V$, we must repeat some factor of $e_{i_1} \wedge e_{i_2} \wedge \cdots \wedge e_{i_n} \in V^{\wedge n}$, but since $V^{\wedge n}$ is spanned by such vectors, we have $V^{\wedge n} = 0$. Thus, $\mathcal{F}V = \bigoplus_{n=0}^{\dim V} V^{\wedge n}$ for finite dimensional V.

The finite dimensionality property gives us an even nicer way to write down a comonoid structure on $\mathcal{F}(V)$, since for finite dimensional vector spaces V with a chosen basis $(e_i)_{i \in I}$, we have $V \cong V^*$ by taking the set $(f_i)_{i \in I}$ as a basis for V^*, where every functional f_j is defined on the basis of V as $f_j(e_i) = \delta_{ij}$. Thus, $(\mathcal{F}V)^* \cong \mathcal{F}V$, meaning we can define the comultiplication on $\mathcal{F}V$. Of course this formally carries little significance, but practically it is easier to work with elements of $\mathcal{F}V$ rather than duals. The dualising process has a more fundamental importance to the categorical semantics of (McPheat et al., 2021), as outlined in 4.4.

The assumption that we can choose a basis is informed by our application domain. For any application of this theory, we either work with vector spaces

built from term-term matrices and thus we fix a set of context words as the bases (Grefenstette and Sadrzadeh, 2011; Kartsaklis et al., 2013) or learn the vectors and tensors by machine learning on a fixed set of features (Wijnholds et al., 2020; Kartsaklis et al., 2019). This makes the availability of bases immediate.

The monoidal product $m : \mathcal{F}V \otimes \mathcal{F}V \to \mathcal{F}V$ is given by layerwise concatenation as it was for tensor algebras. Dualising this product gives us a coproduct $\Delta : \mathcal{F}V \to \mathcal{F}V \otimes \mathcal{F}V$, defined by mapping vectors $\tilde{v} \in \mathcal{F}V$ to all possible vectors in $\mathcal{F}V \otimes \mathcal{F}V$ that multiply to give $\tilde{v}$. Explicitly, this is:

$$\Delta(\tilde{v}) := \sum_{\substack{\tilde{u},\tilde{w}\in\mathcal{F}V \\ \tilde{v}=m(\tilde{u}\otimes\tilde{w})}} \tilde{u} \otimes \tilde{w}$$

This mapping is hard to use in practice, since if $\dim V = n$ then, $\dim \mathcal{F}V = 2^n$, making the comultiplication Δ a $2^n \times (2^n)^2$ matrix. Any practical applications of this would use $n \geq 100$ at the very least, making the mapping quite difficult to use. We overcome this problem by considering other comultiplications which we will define in 5.

4.2 String Diagrams

The formalisation of categorical reasoning using string diagrams was first achieved in (Joyal and Street, 1991), then systematically extended in (Selinger, 2010) to autonomous, braided and traced categories, to name only a few. In the same style, (Baez and Stay, 2011) interpret monoidal biclosed categories in their clasped diagrammatic calculus. In this section, we go over the clasp diagrams in order to later represent our derivations in a more legible format. The use of clasp diagrams to depict Lambek calculus derivations is possible, since the categorical semantics of Lambek calculus is a monoidal bi-closed category. This was initiated in the work of (Coecke et al., 2013) and proved coherent in (Wijnholds, 2017). In (McPheat et al., 2021), we added some more structure in the form of Δ-maps and the diagrammatic !-modality. This extension of the clasp diagrams requires a new proof of coherence, which constitutes work in progress.

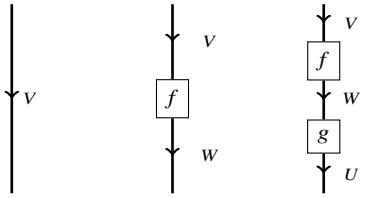


Figure 5. Vector spaces, linear maps and composition of linear maps, in terms of string diagrams

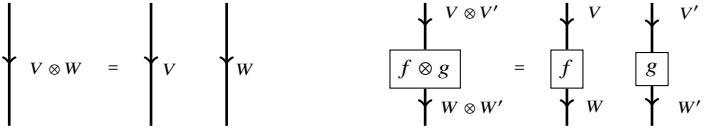


Figure 6. Tensor products of vector spaces on the left, and of linear maps on the right.

We recall how to draw objects morphisms and composition diagrammatically in figure 5.

Tensor products of vector spaces and linear maps are drawn side by side as in figure 6. We use the clasp notation of (Baez and Stay, 2011) to draw vector spaces of the form $V \Rightarrow W$ and $W \Leftarrow V$ as depicted in figure 7. Recall that these are the set of linear maps $V \to W$, which gets its vector space structure pointwise.

Figure 7. Right facing and left facing clasp diagrams representing vector spaces containing $\Rightarrow, \Leftarrow$.

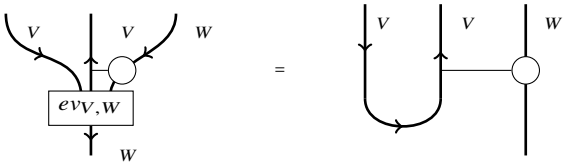


Figure 8. (Left) evaluation drawn as a cup

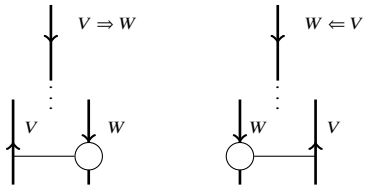


Figure 9. How we identify strings with complex labels and their diagrams

Recall that evaluation of linear maps is a linear map itself of the form $\mathrm{ev}_{V,W} : V \otimes (V \Rightarrow W) \to W :: v \otimes f \mapsto f(v)$. We have a concise depiction of evaluation of linear maps in our diagrams, namely as "cups", as seen in figure 8. Note also that $\Leftarrow$ and $\Rightarrow$ are isomorphic, and so there is only one evaluation map. This is easiest to see by considering the two (a priori distinct) evaluation maps $v \otimes f \mapsto f(v)$ and $f \otimes v \mapsto f(v)$. Since the tensor product is symmetric, we can get the first map from the other by first applying symmetry and vice versa, making the two maps isomorphic. We choose to distinguish between $\Rightarrow$ and $\Leftarrow$ in our presentation of the semantics because it keeps us closer to the $!\mathbf{L}^*$-syntax. Work is being done to define a tensor product on vector spaces which is non-symmetric, thus distinguishing between $\Rightarrow$ and $\Leftarrow$ in (Correia et al., 2019).

In some cases it will simplify our diagrams significantly if we can interchange between our string diagrammatic conventions for vector spaces defined using the $\Rightarrow$ and $\Leftarrow$ operations. We will denote these equalities using vertical dots as shown in figure 9.

4.3 Vector Space Semantics for $!\mathbf{L}^*$

We can now define the part of the vector space semantics for $!\mathbf{L}^*$ necessary for the derivations of anaphora and ellipsis. We also show how to interpret the rules of $!\mathbf{L}^*$ using the diagrams of the previous section, which will enable us to read the derivations of anaphora and ellipsis in a far more legible manner. For a definition of vector space semantics for the rest of $!\mathbf{L}^*$ and the proof of soundness, we refer the reader to the original paper (McPheat et al., 2021).

Definition 4 *We inductively define a semantic map, which using conventional notation we denote by " $[\![\ ]\!]$", on formulas and derivable sequents of $!\mathbf{L}^*$. This map sends formulas to finite dimensional real vector spaces, and derivable sequents to linear maps. We give the inductive definition below.*

- *To atomic types N and S of $!\mathbf{L}^*$ we assign vector spaces as follows*

$$[\![N]\!] := V_N \qquad [\![S]\!] := V_S$$

- *To complex types of $!\mathbf{L}^*$ we assign:*

$$\begin{array}{rcl} [\![A, B]\!] & := & [\![A]\!] \otimes [\![B]\!] \\ [\![A \backslash B]\!] & := & [\![A]\!] \Rightarrow [\![B]\!] \\ [\![B / A]\!] & := & [\![B]\!] \Leftarrow [\![A]\!] \\ [\![!A]\!] & := & \mathcal{F}[\![A]\!] \end{array}$$

In the above definition, for any two vector spaces V, W, the spaces $V \Rightarrow W$ and $W \Leftarrow V$ denote the set of linear maps from W to V. The space $\mathcal{F}V$ for a finite dimensional vector space V is the Fermionic Fock Space of V. For a finite list of formulas $\Gamma = \{A_1, A_2, \ldots, A_n\}$ we define $[\![\Gamma]\!] := [\![A_1]\!] \otimes [\![A_2]\!] \otimes \cdots \otimes [\![A_n]\!]$.

Derivable sequents $\Gamma \longrightarrow A$ are interpreted as linear maps $[\![\Gamma]\!] \longrightarrow [\![A]\!]$. Since sequents are not typically labelled, we add lower case roman letters $(f, g, h, \ldots)$ to name linear maps when needed. To define the interpretation a derivable sequent $\Gamma \longrightarrow A$ in practice, one essentially builds it from the root of the derivation up, following the below interpretations of the proof rules of $!\mathbf{L}^*$. As we introduce the semantics of all the rules of $!\mathbf{L}^*$ we also show how to depict them as string diagrams in $\mathbf{FdVect}_{\mathbb{R}}$.

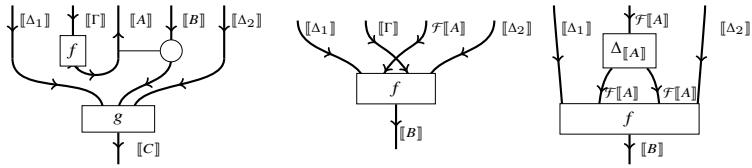
We begin by interpreting the axiom rule $A \longrightarrow A$ of $!\mathbf{L}^*$. This is simply interpreted as the existence of an identity map $id_{[\![A]\!]} : [\![A]\!] \longrightarrow [\![A]\!]$. Diagrammatically, the axiom rule is depicted with a string labelled $[\![A]\!]$. The $(\backslash L)$ and $(/L)$-rules are interpreted very similarly, so we will show one and let the reader deduce the other. We recall the syntax of the $(\backslash L)$-rule on the left below, and lay out the semantics on the right and define it below

$$\frac{\Gamma \longrightarrow A \quad \Delta_1, B, \Delta_2 \longrightarrow C}{\Delta_1, \Gamma, A\backslash B, \Delta_2 \longrightarrow C}\ (\backslash L) \qquad \frac{f : [\![\Gamma]\!] \longrightarrow [\![A]\!] \quad g : [\![\Delta_1]\!] \otimes [\![B]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![C]\!]}{h : [\![\Delta_1]\!] \otimes [\![\Gamma]\!] \otimes [\![A]\!] \Rightarrow [\![B]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![C]\!]}\ (\backslash L)$$

where we are given f and g, and define h as

$$h := g \circ (id_{\Delta_1} \otimes \mathrm{ev}_{[\![A]\!],[\![B]\!]} \otimes id_{\Delta_2}) \circ (id_{\Delta_1} \otimes f \otimes id_{[\![A]\!] \Rightarrow [\![B]\!]} \otimes id_{\Delta_1})$$

which can be visualised diagrammatically in figure 10a.



(a) **(b)** **(c)**

Figure 10. Diagrammatic interpretation of structural $!\mathbf{L}^*$ rules.

Next we have the semantics of the permutation rules, which are immediate from the symmetry of the tensor product in $\mathbf{FdVect}_{\mathbb{R}}$. Diagrammatically, these rules say that we may cross strings representing Fock spaces as done in figure 10b.

Finally, we have the contraction rule. This is interpreted using the comonoidal comultiplication introduced in 4.1.

$$\frac{\Delta_1, !A, !A, \Delta_2 \longrightarrow B}{\Delta_1, !A, \Delta_2 \longrightarrow B}\ (contr) \qquad \frac{f : [\![\Delta_1]\!] \otimes [\![!A]\!] \otimes [\![!A]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![B]\!]}{f^c : [\![\Delta_1]\!] \otimes [\![!A]\!] \otimes [\![\Delta_2]\!] \longrightarrow [\![B]\!]}\ (contr)$$

where given f, we define f^c as $f^c := f \circ (id_{[\![\Delta_1]\!]} \otimes \Delta_{[\![A]\!]} \otimes id_{[\![\Delta_2]\!]})$, where $\Delta_{[\![A]\!]}$ is a comultiplication $\mathcal{F}[\![A]\!] \to \mathcal{F}[\![A]\!] \otimes \mathcal{F}[\![A]\!]$. Diagrammatically, this rule lets us 'split' strings corresponding to !-ed formulas, as in figure 10c.

Note that the Δ-box only specifies the type of the map Δ, meaning that we can do a derivation in $!\mathbf{L}^*$, interpret it as string diagram, and then choose whichever instances of Δ as we please.

It is worth pointing out that we do not aim to define a *complete* model. As probably already noted by the reader, interpreting $!\mathbf{L}^*$ in terms of vector spaces could never be complete; take for example the distinction between $\backslash$ and $/$ in $!\mathbf{L}^*$ which is clearly not carried into the semantics. However, this is common practice in the DisCoCat line of research (Coecke et al., 2010, 2013) and has also been employed in set theoretic semantics of Lambek calculus (Van Benthem, 1988). The choice of vector spaces as an interpretion of $!\mathbf{L}^*$ reflects that one can convey the same meanings in syntacically different languages. Consider for example the English sentence "*John likes Mary*" and the Farsi sentence "*John Mary-ra Doost-darad(likes)*". The syntaxes are distinct, but the meanings are the same. This is despite the fact that the order of words is very different in the two languages.

Finding complete models of $!\mathbf{L}^*$ is an interesting pursuit, although not the goal of the current paper. Work in this direction has begun in (Correia et al., 2019; Greco et al., 2020), where the use of different nonsymmetric tensor products are investigated. One of our reviewers has suggested specifying subspaces of atomic vector spaces in order to distinguish between spaces and their duals. This also takes us closer to a complete model. We have also considered defining a model in C^*-algebras, which have a nonsymmetric tensor product.

4.4 Categorical Vector Space Semantics

For readers who are familiar with categorical semantics of linear logic such as (Melliès, 2014), and diagrammatic reasoning, you may have noticed some category-theoretic constructions, which we have not mentioned; we briefly outline them here and again refer the reader to (McPheat et al., 2021) for full detail. The category theoretic machinery is incredibly useful and succinct for

proving soundness results, but can be cumbersome in practice when looking at concrete applications as we do in this paper.

We mention that $\mathcal{F}$ is a free functor $\mathbf{FdVect}_{\mathbb{R}} \to \mathbf{AAlg}_{\mathbb{R}}$, and then mention dualising spaces $\mathcal{F}V$. Combining these two facts actually defines comonad on $\mathbf{FdVect}_{\mathbb{R}}$. First of all, since $\mathcal{F}$ is free, we have a a right adjoint forgetful functor $\mathbf{AAlg}_{\mathbb{R}} \to \mathbf{FdVect}_{\mathbb{R}}$, which when composed form a *monad* $U\mathcal{F} : \mathbf{FdVect}_{\mathbb{R}} \to \mathbf{FdVect}_{\mathbb{R}}$. Then, pre and post-compose the functor $U\mathcal{F}$ with the vector space dual, which as shown in (Bruguières and Virelizier, 2007), defines a comonad.

To explicitly define the structure of this comonad we will make use of the isomorphism $(U\mathcal{F}V^*)^* \cong U\mathcal{F}V$ to simplify the mathematics. The counit of the comonad $(\varepsilon_V : U\mathcal{F}V \to V)_{V \in \mathbf{FdVect}_{\mathbb{R}}}$ is defined using projection onto the first layer. The comultiplication of the comonad $(\delta_V : U\mathcal{F}V \to U\mathcal{F}U\mathcal{F}V)_{V \in \mathbf{FdVect}_{\mathbb{R}}}$ is given by inclusion into the first layer. That is, $\varepsilon_V(v_0, v_1, v_2 \wedge v_3, \ldots) = v_1 \in V$, and $\delta_V(\tilde{v}) := (0, \tilde{v}, 0, 0, \ldots)$

There is an alternative categorical interpretation of the relevant modality ! provided by (Jacobs, 1994) using what the author calls "relevant monads". However, these monads are the categorical semantics of a smaller fragment of logic which is only concerned with contraction, as opposed to $!\mathbf{L}^*$ which considers ! to be responsible for both contraction and permutation. Further, for the semantics of ! to be sound for the full logic of $!\mathbf{L}^*$, we need ! to be a comonad rather than a monad. The comonad counit propoerty is necessary to prove soundness of the $(!L)$-rule of the calculus. This is the approach taken by the paper (Blute et al., 1994) showing that Fock spaces can been used to interpret the linear logic !-modality and provide a sound semantics for the full logic. There is a natural connection to a comonadic interpretation rather than a monadic one, again suggesting that relevant monads are not the immediate best semantics for the ! of $!\mathbf{L}^*$. However considering a smaller fragment of $!\mathbf{L}^*$, or perhaps a Lambek calculus with a soft exponential modality as in (Lafont, 2004) may allow the use of contraction monads.

5. Examples of Vector Space Semantics of Anaphora and Ellipsis

In this section we show how to interpret each of the derivations from section 3 in our vector space semantics. In each example we first draw the string diagram corresponding to the derivation which firstly gives us a far more readable version of the sequent derivation, and secondly gives us the linear map corresponding to the meaning. We demonstrate how to extract the linear map from the diagram, and show how to evaluate it for each example.

We will use the following notation in the coming subsections. Superscript tildes denote vectors in Fock spaces (i.e. $\tilde{v} \in \mathcal{F}V$). We fix bases $(n_i)_{i\in I}$ and $(s_j)_{j\in J}$ for spaces $[\![N]\!]$, $[\![S]\!]$ respectively. Basis elements marked with asterisks denote basis elements in dual spaces $[\![N]\!]^*$, $[\![S]\!]^*$, i.e. $n_i^* : [\![N]\!] \to \mathbb{R} :: n_{i'} \mapsto \delta ii'$. We also recall that you may define a basis of a tensor product of vector spaces $V \otimes W$, as the tensor products of the basis vectors of V and W. In particular for spaces of the form $[\![N]\!] \Rightarrow [\![S]\!]$ we work with the basis $(n_i^* \otimes s_j)_{i\in I, j\in J}$. We also fix a number $k \in \mathbb{R}$ and use boldface $\mathbf{k}$ to denote a vector of k's in the appropriate vector space, e.g. $\mathbf{k} = (k, k) \in \mathbb{R}^2$ or $\mathbf{k} = (k, k, k, k) \in \mathbb{R}^4$ and so on.

In the following examples we will write out the semantics of the derivations from section 3 using two different instances of the Δ-map. However there is a way to write out the semantics without specifying a Δ-map, by using **Sweedler notation**. This is a notation for abstract comultiplication maps $\Delta : V \to V \otimes V$ where we write $\Delta(v) := v_{(1)} \otimes v_{(2)}$. Once Δ is specified, we substitute the relevant terms in for $v_{(1)}$ and $v_{(2)}$. We use this notation when appropriate to simplify or generalise the computations. The Δ-maps we use are called the **k-extension** and **basis copy** maps. $\mathbf{k}$-extension maps vectors v to $v \otimes \mathbf{k} + \mathbf{k} \otimes v$ (this was called *cofree-inspired* in (McPheat et al., 2021)). The basis copy map is defined on the basis of the relevant vector space as $e_i \mapsto e_i \otimes e_i$, and extended linearly to the whole space. After extending, one can give two mathematically equivalent versions of this map by gathering the coefficients to the left or right factor. That is, if we apply basis copying to a vector $v = \sum_i C_i v_i$ we get $\Delta(v) = \sum_i C_i(v_i \otimes v_i)$, and by gathering the coefficients on the left factor we get $\Delta(v) = v \otimes \mathbf{1}$, or on the right we get

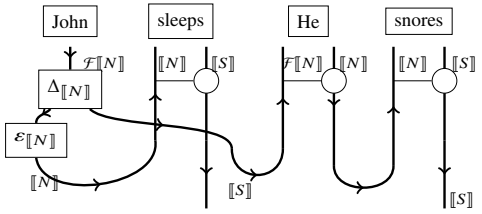


Figure 11. Anaphora diagram

$\Delta(v) = \mathbf{1} \otimes v$. Although mathematically equivalent, a choice must be made to implement this model. We call the maps gathering the coefficients on the left(right) basis copying a(b) (these were called *cogebra a(b)* in (McPheat et al., 2021)).

5.1 Semantics of Anaphora

We begin with drawing the diagram corresponding to the derivation in figure 1. Reading the derivation from bottom to top, we draw the diagram from top to bottom[4]. The resulting string diagram in figure 11, clearly shows that the meaning of '*John*' is being copied by the $\Delta_{[\![N]\!]}$ map, and then one of the copies is sent to '*He*', which is in turn sends the meaning of '*John*' to the input of '*snores*'. This is far easier to see using string diagrams rather than the sequent calculus derivation. The vector spaces at the top and the bottom of the diagram (corresponding respectively to the semantics of the left and right hand sides of the sequent $!N, N\backslash S, !N\backslash N, N\backslash S \longrightarrow S, S$) tell us that this diagram defines a linear map, say f, of type

$$\mathcal{F}[\![N]\!] \otimes [\![N]\!] \otimes [\![S]\!] \otimes \mathcal{F}[\![N]\!] \Rightarrow [\![N]\!] \otimes [\![N]\!] \Rightarrow [\![S]\!] \rightarrow [\![S]\!] \otimes [\![S]\!]\,.$$

Explicitly, taking the vectors $\widetilde{John} \in \mathcal{F}[\![N]\!]$, $\overrightarrow{sleeps}, \overrightarrow{snores} \in [\![N]\!] \Rightarrow [\![S]\!]$ and $\overrightarrow{He} \in \mathcal{F}[\![N]\!] \Rightarrow [\![N]\!]$, we may define f (in Sweedler notation) as:

$$f(\widetilde{John} \otimes \overrightarrow{sleeps} \otimes \overrightarrow{He} \otimes \overrightarrow{snores}) := \overrightarrow{sleeps}(\varepsilon_{[\![N]\!]}(\widetilde{John}_{(1)})) \otimes \overrightarrow{snores}(\overrightarrow{He}(\widetilde{John}_{(2)})).$$

[4]For a step-by-step example of how to draw these diagrams, we refer the reader to (McPheat et al., 2021) where we present how to draw a string diagram from a $!\mathbf{L}^*$-derivation.

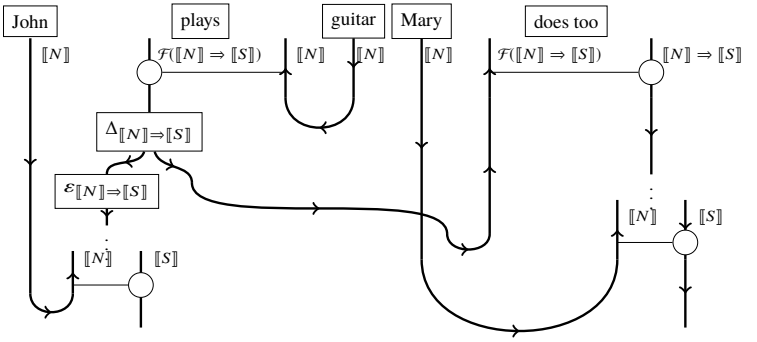


Figure 12. Ellipsis string diagram

By using **k**-extension or basis copying for the $\Delta_{\llbracket N\rrbracket}$-map, f becomes:

k-extension :
$f(\widetilde{John} \otimes \overrightarrow{sleeps} \otimes \overrightarrow{He} \otimes \overrightarrow{snores}) :=$
$\overrightarrow{sleeps}(\varepsilon_{\llbracket N\rrbracket}(\widetilde{John})) \otimes \overrightarrow{snores}(\overrightarrow{He}(\mathbf{k})) + \overrightarrow{sleeps}(\varepsilon_{\llbracket N\rrbracket}(\mathbf{k})) \otimes \overrightarrow{snores}(\overrightarrow{He}(\widetilde{John}))$
Basis copy :
$f(\tilde{n}_i \otimes (n^*_{i'} \otimes s_j) \otimes (\tilde{n}^*_{i''} \otimes n_{i'''}) \otimes (n^*_{i''''} \otimes s_{j'})) := n^*_{i'}(n_i)s_j \otimes \tilde{n}^*_{i''}(\tilde{n}_i)n^*_{i''''}(n_{i'''})s_{j'}$

5.2 Semantics of Ellipsis

We proceed in this example as we did for the anaphora example, by reading the derivation of the ellipsis (figure 2) from the root, and the diagram from the top, as presented in figure 12. By inspecting the strings at the top and bottom of the diagram, we see that this defines a linear map, say g, of type

$$\llbracket N\rrbracket \otimes (\mathcal{F}(\llbracket N\rrbracket \Rightarrow \llbracket S\rrbracket)) \Leftarrow \llbracket N\rrbracket \otimes \llbracket N\rrbracket \otimes \llbracket N\rrbracket \otimes \mathcal{F}(\llbracket N\rrbracket \Rightarrow \llbracket S\rrbracket) \Rightarrow (\llbracket N\rrbracket \Rightarrow \llbracket S\rrbracket) \rightarrow \llbracket S\rrbracket \otimes \llbracket S\rrbracket.$$

If we consider vectors

$$\overrightarrow{John}, \overrightarrow{guitar}, \overrightarrow{Mary} \in \llbracket N\rrbracket, \quad \overrightarrow{plays} \in \mathcal{F}(\llbracket N\rrbracket \Rightarrow \llbracket S\rrbracket) \Leftarrow \llbracket N\rrbracket,$$
$$\overrightarrow{does\ too} \in \mathcal{F}(\llbracket N\rrbracket \Rightarrow \llbracket S\rrbracket) \Rightarrow (\llbracket N\rrbracket \Rightarrow \llbracket S\rrbracket)$$

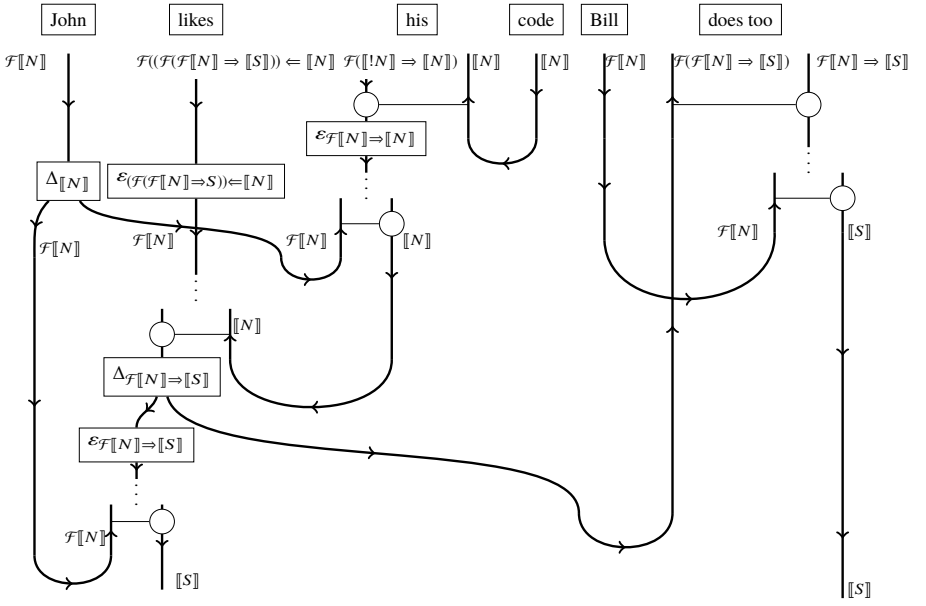


Figure 13. Strict Diagram

we can define g as

$$g(\overrightarrow{John} \otimes \overrightarrow{plays} \otimes \overrightarrow{guitar} \otimes \overrightarrow{Mary} \otimes \overrightarrow{does\,too}) :=$$
$$\varepsilon_{[\![N]\!] \Rightarrow [\![S]\!]}((\overrightarrow{plays}(\overrightarrow{guitar})_{(1)})(\overrightarrow{John}) \otimes \overrightarrow{does\,too}(\overrightarrow{plays}(\overrightarrow{guitar})_{(2)}, \overrightarrow{Mary}).$$

Note that function $\overrightarrow{plays}$ is a function of type $[\![N]\!] \to \mathcal{F}([\![N]\!] \Rightarrow [\![S]\!])$ and is first being evaluated at the noun $\overrightarrow{guitar}$. This lets us define

$$\widetilde{plays\,guitar} \in \mathcal{F}([\![N]\!] \Rightarrow [\![S]\!]) \quad \text{as} \quad \widetilde{plays\,guitar} := \overrightarrow{plays}(\overrightarrow{guitar}).$$

We may also consider taking $\overrightarrow{does\,too}$ to be a projection, like ε, which maps $\widetilde{plays\,guitar}$ to a vector $\overrightarrow{plays\,guitar} \in [\![N]\!] \Rightarrow [\![S]\!]$. This makes the

definition of g slightly neater:

$$g(\overrightarrow{John} \otimes \overrightarrow{plays} \otimes \overrightarrow{guitar} \otimes \overrightarrow{Mary} \otimes \overrightarrow{does\,too}) :=$$
$$\varepsilon_{[\![N]\!]\Rightarrow[\![S]\!]}(\widetilde{plays\ guitar}_{(1)})(\overrightarrow{John}) \otimes \overrightarrow{plays\ guitar}(\overrightarrow{Mary}).$$

The mathematical form of g looks very similar to the natural language semantics '*John plays guitar, Mary does too*'.

Finally, when specifying the Δ-maps to be one of the **k**-extension and basis copying we get:

k-extension

$$g(\overrightarrow{John} \otimes \overrightarrow{plays} \otimes \overrightarrow{guitar} \otimes \overrightarrow{Mary} \otimes \overrightarrow{does\,too}) :=$$
$$\varepsilon_{[\![N]\!]\Rightarrow[\![S]\!]}((\overrightarrow{plays}(\overrightarrow{guitar}))(\overrightarrow{John}) \otimes \overrightarrow{does\,too}(\mathbf{k}, \overrightarrow{Mary}) +$$
$$\varepsilon_{[\![N]\!]\Rightarrow[\![S]\!]}(\mathbf{k})(\overrightarrow{John}) \otimes \overrightarrow{does\,too}(\overrightarrow{plays}(\overrightarrow{guitar}, \overrightarrow{Mary})$$

Basis copy

$$g(n_{i_1} \otimes (\widetilde{n^*_{i_2} \otimes s_{j_1}} \otimes n^*_{i_3}) \otimes n_{i_4} \otimes n_{i_5} \otimes ((\widetilde{n^*_{i_6} \otimes s_{j_2}})^* \otimes n^*_{i_7} \otimes s_{j_3})) :=$$
$$n^*_{i_2}(n_{i_1})s_j n^*_{i_3}(n_{i_4}) \otimes (\widetilde{n^*_{i_6} \otimes s_{j_2}})^*(\widetilde{n^*_{i_2} \otimes s_{j_1}})n^*_{i_7}(n_{i_5})s_{j_3}.$$

5.3 Anaphora with Ellipsis

An important contribution to note in the following is that the linear maps for the strict and sloppy semantics are distinct, thus showing that not only does $!\mathbf{L}^*$ provide a syntactic distinction between strict and sloppy, but also so does the vector space semantics. This overcomes the difficulty faced in (Wijnholds and Sadrzadeh, 2019b) where there are indeed two different syntactic derivations, but the corresponding vector spaces semantics collapses and they become equal to each other.

As we have done in 5.1 and 5.2 we first present the diagrams of the strict and sloppy derivations of (figures 3 and 4), which is done in figures 13 and 14 respectively.

Again, by inspecting the wires on the top and bottom of both diagrams, we see that they specify linear maps, say h_{strict}, h_{sloppy}, which are both of the following type:

$$\mathcal{F}[\![N]\!] \otimes \mathcal{F}(\mathcal{F}(\mathcal{F}[\![N]\!] \Rightarrow [\![S]\!]) \Leftarrow [\![N]\!]) \otimes \mathcal{F}(\mathcal{F}[\![N]\!] \Rightarrow [\![N]\!]) \Leftarrow [\![N]\!] \otimes [\![N]\!] \otimes$$
$$\mathcal{F}[\![N]\!] \otimes (\mathcal{F}(\mathcal{F}[\![N]\!] \Rightarrow [\![S]\!])) \Rightarrow (\mathcal{F}[\![N]\!] \Rightarrow [\![S]\!]) \rightarrow [\![S]\!] \otimes [\![S]\!].$$

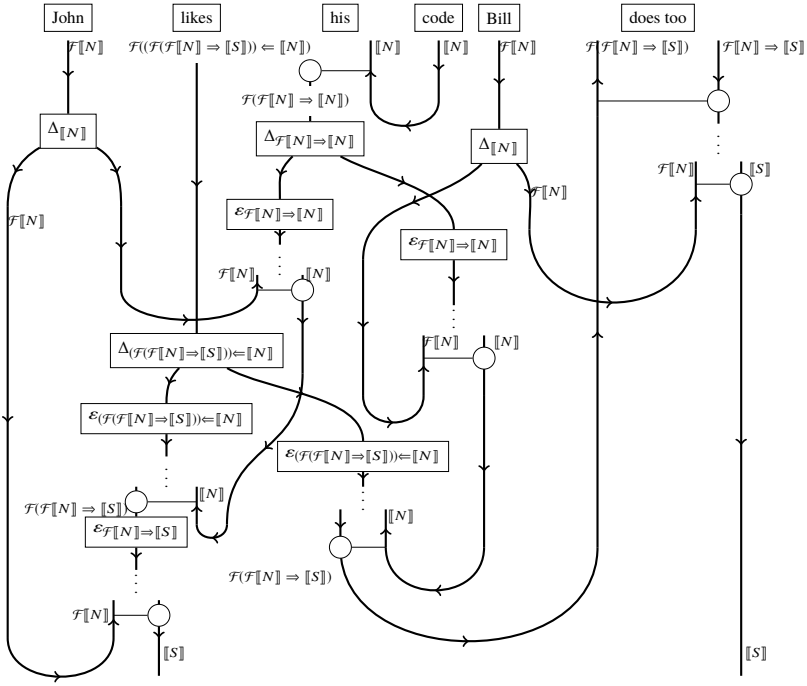


Figure 14. Sloppy Diagram

Taking vectors

$$\widetilde{John}, \widetilde{Bill} \in \mathcal{F}[\![N]\!], \quad \widetilde{likes} \in \mathcal{F}(\mathcal{F}(\mathcal{F}[\![N]\!] \Rightarrow [\![S]\!]) \Leftarrow [\![N]\!])$$
$$\overrightarrow{his} \in \mathcal{F}(\mathcal{F}[\![N]\!] \Rightarrow [\![N]\!]) \Leftarrow [\![N]\!], \quad \overrightarrow{code} \in [\![N]\!]$$
$$\overrightarrow{does\,too} \in (\mathcal{F}(\mathcal{F}[\![N]\!] \Rightarrow [\![S]\!])) \Rightarrow (\mathcal{F}[\![N]\!] \Rightarrow [\![S]\!])$$

we can define the maps h_{strict}, h_{sloppy} with Sweedler notation as:

$$h_{strict}(\widetilde{John} \otimes \widetilde{likes} \otimes \overrightarrow{his} \otimes \overrightarrow{code} \otimes \widetilde{Bill} \otimes \overrightarrow{does\,too}) :=$$
$$\varepsilon_{\mathcal{F}[\![N]\!]\Rightarrow[\![S]\!]}(\varepsilon_{\mathcal{F}(\mathcal{F}[\![N]\!]\Rightarrow[\![S]\!])\Leftarrow[\![N]\!]}(\widetilde{likes})(\varepsilon_{\mathcal{F}[\![N]\!]\Rightarrow[\![N]\!]}(\widetilde{his}(code)))_{(1)})(\widetilde{John}_{(2)})\otimes$$
$$\overrightarrow{does\,too}(\varepsilon_{\mathcal{F}(\mathcal{F}[\![N]\!]\Rightarrow[\![S]\!])\Leftarrow[\![N]\!]}(\widetilde{likes})(\varepsilon_{\mathcal{F}[\![N]\!]\Rightarrow[\![N]\!]}(\widetilde{his}(code)))_{(2)}))(\varepsilon_{[\![N]\!]}(\widetilde{Bill}))$$

$$h_{sloppy}(\widetilde{John} \otimes \widetilde{likes} \otimes \overrightarrow{his} \otimes \overrightarrow{code} \otimes \widetilde{Bill} \otimes \overrightarrow{does\,too}) :=$$
$$\varepsilon_{\mathcal{F}(\mathcal{F}[\![N]\!]\Rightarrow[\![S]\!])\Leftarrow[\![N]\!]}(\widetilde{likes}_{(1)})(\varepsilon_{\mathcal{F}[\![N]\!]\Rightarrow[\![N]\!]}(\widetilde{his}_{(1)})(\overrightarrow{code})(\widetilde{John}_{(2)}))(\widetilde{John}_{(1)})\otimes$$
$$\overrightarrow{does\,too}(\varepsilon_{\mathcal{F}(\mathcal{F}[\![N]\!]\Rightarrow[\![S]\!])\Leftarrow[\![N]\!]}(\widetilde{likes}_{(2)})((\varepsilon_{\mathcal{F}[\![N]\!]\Rightarrow[\![N]\!]}(\widetilde{his}_{(1)})(\overrightarrow{code})(\widetilde{Bill}_{(2)})))(\varepsilon_{[\![N]\!]}(\widetilde{Bill}_{(2)})$$

Finally, we show what the strict and sloppy maps look like when specifying Δ to be **k**-extension and basis copying, however we omit the **k**-extension map for the sloppy reading, as this takes about 16 lines to define, and becomes highly uninformative.

k-extension

$$h_{strict}(\widetilde{John} \otimes \widetilde{likes} \otimes \overrightarrow{his} \otimes \overrightarrow{code} \otimes \widetilde{Bill} \otimes \overrightarrow{does\,too}) :=$$
$$(likes(\overrightarrow{John}, his(\mathbf{k}, \overrightarrow{code})) \otimes \mathbf{k}(\overrightarrow{Bob}, his(\mathbf{k}, \overrightarrow{code}))+$$
$$\mathbf{k}(\overrightarrow{John}, his(\mathbf{k}, \overrightarrow{code}) \otimes likes(\overrightarrow{Bob}, his(\mathbf{k}, \overrightarrow{code})))+$$
$$likes(\mathbf{k}, his(\overrightarrow{John}, \overrightarrow{code})) \otimes \mathbf{k}(\overrightarrow{Bob}, his(\overrightarrow{John}, \overrightarrow{code}))+$$
$$\mathbf{k}(\mathbf{k}, his(\mathbf{k}, \overrightarrow{code})) \otimes likes(\overrightarrow{1}, his(\overrightarrow{John}, \overrightarrow{code}))$$

Basis copy

$$h_{strict}(\tilde{n}_{i_1} \otimes ((\widetilde{\tilde{n}^*_{i_2} \otimes s_{j_1}}) \otimes n^*_{i_3}) \otimes ((\widetilde{\tilde{n}^*_{i_4} \otimes n_{i_5}}) \otimes n^*_{i_6}) \otimes n_{i_7} \otimes \tilde{n}_{i_8} \otimes ((\widetilde{\tilde{n}^*_{i_9} \otimes s_{j_2}})^* \otimes \tilde{n}^*_{i_{10}} \otimes s_{j_3}))$$
$$:= (n^*_{i_2}(n_{i_1})s_{j_1}n^*_{i_3}(n_{i_5})n^*_{i_4}(n_{i_1})n^*_{i_6}(n_{i_7}))\otimes$$
$$(n^*_{i_{10}}(n_{i_8})s_{j_3}(n^*_{i_9} \otimes s_{j_2})^*(n^*_{i_2} \otimes s_{j_1})n^*_{i_3}(n_{i_5})n^*_{i_4}(n_{i_1})n^*_{i_6}(n_{i_7}))$$

$$h_{sloppy}(\tilde{n}_{i_1} \otimes ((\widetilde{\tilde{n}^*_{i_2} \otimes s_{j_1}}) \otimes n^*_{i_3}) \otimes ((\widetilde{\tilde{n}^*_{i_4} \otimes n_{i_5}}) \otimes n^*_{i_6}) \otimes n_{i_7} \otimes \tilde{n}_{i_8} \otimes ((\widetilde{\tilde{n}^*_{i_9} \otimes s_{j_2}})^* \otimes \tilde{n}^*_{i_{10}} \otimes s_{j_3}))$$
$$:= (n^*_{i_2}(n_{i_1})s_{j_1}n^*_{i_3}(n_{i_5})n^*_{i_4}(n_{i_1})n^*_{i_6}(n_{i_7}))\otimes$$
$$(n^*_{i_{10}}(n_{i_8})s_{j_3}(n^*_{i_9} \otimes s_{j_2})^*(n^*_{i_2} \otimes s_{j_1})n^*_{i_3}(n_{i_5})n^*_{i_4}(n_{i_8})n^*_{i_6}(n_{i_7}))$$

6. Experiments

We implement our copying operations on the disambiguation task of (Wijnholds and Sadrzadeh, 2019a), for the purpose of deciding which copying map does better in practice. The aim is to compare the performance of our linear

copying maps to each other and to the non linear copying map. We would also like to find out how well do the compositional models do in comparison to a non compositional verb only baselines and non grammatical compositional methods such as addition.

The disambiguation task of (Wijnholds and Sadrzadeh, 2019a), extends the original disambiguation task introduced in (Grefenstette and Sadrzadeh, 2011) with elliptic phrases. The original dataset of (Grefenstette and Sadrzadeh, 2011) worked with 10 ambiguous verbs and two of their meanings. An example is the verb *draw*, which is ambiguous between *depict* and *pull*. The ambiguous verb and each of its meanings are placed in subject-verb-object triples. For the verb *draw*, we have the sentences

S: man draw sword.
S_1: man depict sword.
S_2: man pull sword.

The dataset consists of pairs of triple (S, S_1) and (S, S_2). The aim of the task is to build vectors for S, S_1, S_2, compute the cosine distances between S, S_1 and S, S_2, in order to decide which meaning of the verb is the more appropriate one in S. Clearly, if S is closer to S_1, its first meaning is deemed more appropriate and if it is closer to S_2, its second meaning.

In (Wijnholds and Sadrzadeh, 2019a), the above dataset is extended to triples with elliptic phrases. It is hypothesised that the extended sentences provide a better base for disambiguation and indeed this hypothesis is verified in the paper. In the interest of space we do not go through the details of this hypothesis and the results and only provide an example. For the ambiguous verb *draw*, we now work with the following sentences:

S': man draw sword and artist does too.
S'_1: man depict sword and artist does too.
S'_2: man pull sword and artist does too.

Vectors for each sentence with elliptic phrase are built using the procedure described in subsection 5.2, where we choose to consider vectors in Fock spaces to be of the form $(0, v, 0, 0, \ldots)$ for practicality. For the verb, previous

experimentation has shown that cubes have a much lower performance than matrices. So although the derivational system prescribes a cube for a transitive verb, in practice it is better to approximate them by matrices. In this case, the cube contraction is approximated by a matrix multiplication followed by a pointwise vector multiplication. We follow previous work and implement the *Relational* verb matrices and the *Copy-Object* composition model. These two methods used in conjunction have consistently provided consistent good results in previous work, e.g. see (Grefenstette and Sadrzadeh, 2011; Kartsaklis et al., 2013; Milajevs et al., 2014; McPheat et al., 2021). The *Relational* method of building verb matrices sums the Kronecker product of the subjects and objects of the verb in the sentences across the corpus, resulting in the formula $\overline{V} := \sum_i \overrightarrow{s}_i \otimes \overrightarrow{o}_i$. Using this formula and employing a *Copy-Object* method in a sentence with elliptical phrase 'Sub1 Verb Obj and Sub2 does-too' results in the following formulae for the **k**-extension copying.

$$\begin{aligned} \textbf{k}\text{-extension}: \quad & ((\overline{V} \times \overrightarrow{Obj}) \odot \overrightarrow{Sub1} + (\mathbf{1} \odot \overrightarrow{Sub2})) + ((\mathbf{1} \odot \overrightarrow{Sub1}) + (\overline{V} \times \overrightarrow{Obj}) \odot \overrightarrow{Sub2}) \\ = \quad & ((\overline{V} \times \overrightarrow{Obj}) \odot \overrightarrow{Sub1} + \overrightarrow{Sub2}) + (\overrightarrow{Sub1} + (\overline{V} \times \overrightarrow{Obj}) \odot \overrightarrow{Sub2}) \end{aligned}$$

Here, following previous work we are interpreting the preposition '*and*' as addition, taking k to be 1, and interpreting the elliptic marker '*does-too*' as identity. The *full* model is where the copying operation is non-linear and provides us with two proper copies of the ellipsis verb phrase, resulting in the following formula:

$$\text{full}: ((\overline{V} \times \overrightarrow{Obj}) \odot \overrightarrow{Sub1}) + ((\overline{V} \times \overrightarrow{Obj}) \odot \overrightarrow{Sub2})$$

We use 100 dimensional pre-trained **word2vec** and **fasttext** vectors for the $\overrightarrow{Sub1}, \overrightarrow{Sub2}, \overrightarrow{Obj}$ vectors, as well as the $\overrightarrow{s}_i$ and $\overrightarrow{o}_i$ vectors used to build our *Relational* verb matrices.

Going through Table 2, we observe that both of the **word2vec** and **fasttext** vectors provide correlations close to the upper-bound, which are slightly higher for **word2vec**. In either case, the best performance is obtained by the full and the **k**-extension copying operations. Interestingly, the compositional models perform better than the non compositional verb-only baseline, which only provides a correlation of 0.24. This low correlation is improved to 0.31 when a non grammatical additive base line is used and to reaches its maximum

		full	basis copy(a)	basis copy(b)	k-extension
word2vec		0.44	0.34	0.42	0.44
fasttext		0.43	0.36	0.41	0.43
baselines					
verb only	0.24				
additive	0.31				
BERT phrase	0.36				
inter-annotator agreement	0.58				

Table 2. Spearman's ρ correlations for the ellipsis disambiguation task; upper bound is the inter annotator agreement score, computed in (Wijnholds, 2020).

when 0.35 when pre-trained **BERT** vectors are used. It is worth noting that the compositional state of the art of the Ellipsis Disambiguation dataset is 0.58, as reported in (Wijnholds et al., 2020) and is obtained via neural verb matrices. In the same paper, it is explained how fine-tuning a **BERT** model provides the best non compositional performance of 0.65. Given that the performances reported here are obtained via pre-trained vectors, we expect that fine tuning our vectors or matrices will provide better results than those reported here. We also expect that the use of neural verb matrices in our linear copying operations will improve the results.

7. Conclusion and Future Work

Following the style of (Coecke et al., 2013), where the authors develop a functorial vector space semantics and string diagrams for Lambek calculus, in another paper (McPheat et al., 2021) we developed a functorial categorical semantics for Lambek calculus with a Relevant Modality of (Kanovich et al., 2016). This logic extends Lambek calculus with a relevant modality that allows for limited contraction and permutation. The motivation for development of $!\mathbf{L}^*$ is the use of limited contraction and permutation to reason about

parasitic gaps in line of research initiated in (Morrill et al., 1990) and followed up on in (Morrill and Valentín, 2016) and (Morrill, 2017, 2018). In this paper, we do not go through the categorical model, as it is extensive and constitutes the contribution of another paper. Instead, we show how one can assign a vector space semantics to this calculus directly, without going through category theory, and with the help of the notion of tensor algebras and a particular finite kind used in Quantum Mechanics: Fermionic Fock Spaces. Inspired by the work of (Jäger, 1998, 2006) and (Wijnholds and Sadrzadeh, 2018, 2019b), for the first time, we apply the Lambek calculus with the Relevant Modality to reason about anaphora with ellipsis and develop closed form linear algebraic terms for the results of the corresponding derivations. We experiment with our model on the Ellipsis Disambiguation dataset of (Wijnholds and Sadrzadeh, 2019a) and observed that our $\mathbf{k}$-extension linear copying operation provides the same results as a full non linear copying operation. These models significantly outperform the non compositional and non grammatical baselines. They are, however, improved by a fine tuned **bert** model and also in a compositional model with neural verb matrices (Wijnholds et al., 2020). Our other contribution is that we show how this vector space semantics is able to distinguish between the two readings of the ambiguous anaphora with ellipsis cases. Indeed and as desired and aligned with the standard literature on coreference modelling (Bach, 2008), we obtain two different linear maps as semantics of these cases, one for the strict reading and one for the sloppy reading. This overcomes the weakness of previous work (Wijnholds and Sadrzadeh, 2019b), where a Lambek calculus with a copying modality was used to model coreference.

References

Bach, Kent. 2008. Review: Robert j. stainton: Words and thoughts: Subsentences, ellipsis, and the philosophy of language. *Mind* 117. doi:10.1093/mind/fzn105.

Baez, John and Michael Stay. 2011. Physics, topology, logic, and computation: A rosetta stone. In B. Coecke, (ed.), *New Structures in Physics*, volume 813 of *Lecture Notes in Physics. Springer*. Springer.

Barry, Guy, Mark Hepple, Neil Leslie, and Glyn Morill. 1995. Proof figures and

structural operations for categorial grammar. In *Proceedings of EACL 1995*, pages 198–203.

Blute, Richard, Robin Cockett, and Robert Seely. 2006. Differential categories. *Mathematical Structures in Computer Science* 16 (6): 1049–1083.

Blute, Richard, Prakash Panangaden, and Robert Seely. 1994. Fock space: a model of linear exponential types. *Manuscript, revised version of the MFPS paper Holomorphic models of exponential types in linear logic* pages 474–512.

Bruguières, Alain and Alexis Virelizier. 2007. Hopf monads. *Advances in Mathematics* doi:10.1016/j.aim.2007.04.011.

Chomsky, Noam. 1981. *Lectures on Government and Binding*. De Gruyter Mouton. doi:doi:10.1515/9783110884166.

Coecke, Bob, Edward Grefenstette, and Mehrnoosh Sadrzadeh. 2013. Lambek vs. lambek: Functorial vector space semantics and string diagrams for lambek calculus. *Annals of Pure and Applied Logic* 164 (11): 1079 – 1100.

Coecke, Bob, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical Foundations for Distributed Compositional Model of Meaning. Lambek Festschrift. *Linguistic Analysis* 36: 345–384.

Correia, Adriana, Michael Moortgat, and Henk Stoof. 2019. Density matrices with metric for derivational ambiguity.

Greco, Giuseppe, Fei Liang, Michael Moortgat, Alessandra Palmigiano, and Apostolos Tzimoulis. 2020. Vector spaces as kripke frames. *Journal of Applied Logics - IfCoLog Journal* 7 (5): 853–873.

Grefenstette, Edward and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Edinburgh, Scotland, UK: Association for Computational Linguistics.

Jacobs, Bart. 1994. Semantics of weakening and contraction. *Annals of Pure and Applied Logic* doi:10.1016/0168-0072(94)90020-5.

Jäger, Gerhard. 1998. A multi-modal analysis of anaphora and ellipsis. *University of Pennsylvania Working Papers in Linguistics* 5 (2): 2.

———. 2006. *Anaphora and type logical grammar*, volume 24. Springer Science & Business Media.

Joyal, André and Ross Street. 1991. The geometry of tensor calculus, i. *Advances in Mathematics* 88 (1): 55–112. doi:https://doi.org/10.1016/0001-8708(91)90003-P.

Kanovich, Max, Stepan Kuznetsov, and Andre Scedrov. 2016. Undecidability of the Lambek calculus with a relevant modality. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*

Bioinformatics) 9804 LNCS: 240–256. doi:10.1007/978-3-662-53042-9_14.

Kartsaklis, Dimitri, Sanjaye Ramgoolam, and Mehrnoosh Sadrzadeh. 2019. Linguistic matrix theory. *Annales de l'Institut Henri Poincaré* 6 (3): 385–426.

Kartsaklis, Dimitri, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 114–123. Sofia, Bulgaria: Association for Computational Linguistics.

Lafont, Yves. 2004. Soft linear logic and polynomial time. *Theoretical Computer Science* doi:10.1016/j.tcs.2003.10.018.

McPheat, Lachlan, Mehrnoosh Sadrzadeh, Hadi Wazni, and Gijs Wijnholds. 2021. Categorical vector space semantics for lambek calculus with a relevant modality (extended abstract). volume 333 of *Electronic Proceedings in Theoretical Computer Science*, pages 168–182. Open Publishing Association. doi: 10.4204/EPTCS.333.12.

Melliès, Paul-André. 2014. Categorical Semantics of Linear Logic (Chapter 8): 181–192. doi:10.1007/978-94-007-7548-0_9.

Milajevs, Dmitrijs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719. Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/D14-1079.

Moortgat, Michael. 1996. Multimodal linguistic inference. *Journal of Logic, Language and Information* 5 (3-4): 349–385.

Moot, Richard and Christian Retoré. 2012. *The Logic of Categorial Grammars A deductive account of natural language syntax and semantics*, volume 6850 of *LNCS*. Springer.

Morrill, Glyn. 2017. Grammar logicised: relativisation. *Linguistics and Philosophy* 40: 119–163.

———. 2018. A note on movement in logical grammar. *Journal of Language Modelling* 6: 353–363.

Morrill, Glyn, Neil Leslie, Mark Hepple, and Guy Barry. 1990. Categorial deductions and structural operations. In *Studies in Categorial Grammar, Edinburgh Working Papers in CognitiveScience*, volume 5, pages 1–21. Centre for Cognitive Science.

Morrill, Glyn and Josep Maria Merenciano Saladrigas. 1996. Generalising discontinuity. *Traitement Automatique des Langues* 27: 119–143.

Morrill, Glyn and Oriol Valentín. 2015. Computational coverage of tlg: Nonlinearity. In *Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science*, volume 32, pages 51–63. EasyChair Publications.

———. 2016. On the logic of expansion in natural language. In *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016) 9th International Conference, LACL 2016, Nancy, France, December 5-7, 2016, Proceedings 9*, pages 228–246. Springer.

Morrill, Glyn, Oriol Valentín, and Mario Fadda. 2011. The displacement calculus. *Journal of Logic, Language and Information* 20 (1): 1–48.

Muskens, Reinhard and Mehrnoosh Sadrzadeh. 2019. Static and dynamic vector semantics for lambda calculus models of natural language. *Journal of Language Modelling* .

Sadrzadeh, Mehrnoosh, Michael Moortgat, and Gijs Wijnholds. 2020. A frobenius algebraic analysis for parasitic gaps. *Journal of Applied Logics - IfCoLog Journal* 7 (5): 823–852.

Selinger, Peter. 2010. A survey of graphical languages for monoidal categories. In Bob Coecke, (ed.), *New Structures for Physics*, volume 813 of *Lecture Notes in Physics book series*, pages 289–355. Springer.

Van Benthem, Johan. 1988. The Lambek Calculus. pages 35–68. doi:10.1007/978-94-015-6878-4_3.

Wijnholds, Gijs. 2017. Coherent Diagrammatic Reasoning in Compositional Distributional Semantics. *LNCS Proceedings of the International Workshop on Logic, Language, Information, and Computation (WoLLIC)* 10388: 371–386.

———. 2020. *A Compositional Vector Space Model of Ellipsis and Anaphora*. Ph.D. thesis, School of Electronic Engineering and Computer Science, Queen Mary University London.

Wijnholds, Gijs and Mehrnoosh Sadrzadeh. 2018. Classical copying versus quantum entanglement in natural language: The case of VP-ellipsis. *Electronic Proceedings in Theoretical Computer Science, EPTCS* 283: 103–119.

———. 2019a. Evaluating composition models for verb phrase elliptical sentence embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 261–271. Minneapolis, Minnesota: Association for Computational Linguistics.

———. 2019b. A type-driven vector semantics for ellipsis with anaphora using lambek calculus with limited contraction. *Journal of Logic Language and Information* 28: 331–358.

Wijnholds, Gijs, Mehrnoosh Sadrzadeh, and Stephen Clark. 2020. Representation learning for type-driven composition. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 313–324. Online: Association for Computational Linguistics. doi:10.18653/v1/2020.conll-1.24.