Mathematical Foundations for a Compositional Distributional Model of Meaning

Bob Coecke*, Mehrnoosh Sadrzadeh*, Stephen Clark†

coecke, mehrs@comlab.ox.ac.uk – stephen.clark@cl.cam.ac.uk

*Oxford University Computing Laboratory
†University of Cambridge Computer Laboratory

Abstract

We propose a mathematical framework for a unification of the distributional theory of meaning in terms of vector space models, and a compositional theory for grammatical types, for which we rely on the algebra of Pregroups, introduced by Lambek. This mathematical framework enables us to compute the meaning of a well-typed sentence from the meanings of its constituents. Concretely, the type reductions of Pregroups are 'lifted' to morphisms in a category, a procedure that transforms meanings of constituents into a meaning of the (well-typed) whole. Importantly, meanings of whole sentences live in a single space, independent of the grammatical structure of the sentence. Hence the inner-product can be used to compare meanings of arbitrary sentences, as it is for comparing the meanings of words in the distributional model. The mathematical structure we employ admits a purely diagrammatic calculus which exposes how the information flows between the words in a sentence in order to make up the meaning of the whole sentence. A variation of our 'categorical model' which involves constraining the scalars of the vector spaces to the semiring of Booleans results in a Montague-style Boolean-valued semantics.

1 Introduction

The symbolic [13] and distributional [36] theories of meaning are somewhat orthogonal with competing pros and cons: the former is compositional but only qualitative, the latter is non-compositional but quantitative. For a discussion of these two competing paradigms in Natural Languge Processing see [15]. Following [39] in the context of Cognitive Science, where a similar problem exists between the

connectionist and symbolic models of mind, [6] argued for the use of the tensor product of vector spaces and pairing the vectors of meaning with their roles. In this paper we will also use tensor spaces and pair vectors with their grammatical types, but in a way which overcomes some of the shortcomings of [6]. One shortcoming is that, since inner-products can only be computed between vectors which live in the same space, sentences can only be compared if they have the same grammatical structure. In this paper we provide a procedure to compute the meaning of any sentence as a vector within a single space. A second problem is the lack of a method to compute the vectors representing the grammatical type; the procedure presented here does not require such vectors.

The use of Pregroups for analysing the structure of natural languages is a recent development by Lambek [19] and builds on his original Lambek (or Syntactic) calculus [18], where types are used to analyze the syntax of natural languages in a simple equational algebraic setting. Pregroups have been used to analyze the syntax of a range of different languages, from English and French to Polish and Persian [32], and many more; for more references see [23, 21].

But what is particularly interesting about Pregroups, and motivates their use here, is that they share a common structure with vector spaces and tensor products, when passing to a category-theoretic perspective. Both the category of vector spaces, linear maps and the tensor product, as well as pregoups, are examples of so-called compact closed categories. Concretely, Pregroups are posetal instances of the categorical logic of vector spaces, where juxtaposition of types corresponds to the monoidal tensor of the monoidal category. The mathematical structure within which we compute the meaning of sentences will be a compact closed category which combines the two above. The meanings of words are vectors in vector spaces, their grammatical roles are types in a Pregroup, and tensor product of vector spaces paired with the Pregroup composition is used for the composition of (meaning, type) pairs.

Type-checking is now an essential fragment of the overall categorical logic, and the reduction scheme to verify grammatical correctness of sentences will not only provide a statement on the well-typedness of a sentence, but will also assign a vector in a vector space to each sentence. Hence we obtain a theory with both Pregroup analysis and vector space models as constituents, but which is inherently compositional and assigns a meaning to a sentence given the meanings of its words. The vectors $\overrightarrow{s}$ representing the meanings of sentences all live in the same meaning space S. Hence we can compare the meanings of any two sentences $\overrightarrow{s}, \overrightarrow{t} \in S$ by computing their inner-product $\langle \overrightarrow{s} | \overrightarrow{t} \rangle$.

Compact closed categories admit a beautiful purely diagrammatic calculus that simplifies the meaning computations to a great extent. They also provide reduction diagrams for typing sentences; these allow for a high level comparison of the

grammatical patterns of sentences in different languages [33]. This diagrammatic structure, for the case of vector spaces, was recently exploited by Abramsky and the second author to expose the *flows of information* withing quantum information protocols [1, 7, 9]. Here, they will expose the flow of information between the words that make up a sentence, in order to produce the meaning of the whole sentence. Note that the connection between linguistics and physics was also identified by Lambek himself [22].

Interestingly, a Montague-style Boolean-valued semantics emerges as a simplified variant of our setting, by restricting the vectors to range over $\mathbb{B} = \{0, 1\}$, where sentences are simply true or false. Theoretically, this is nothing but the passage from the category of vector spaces to the category of relations as described in [8]. In the same spirit, one can look at vectors ranging over $\mathbb{N}$ or $\mathbb{Q}$ and obtain degrees or probabilities of meaning. As a final remark, in this paper we only set up our general mathematical framework and leave a practical implementation for future work.

2 Two 'camps' within computational linguistics

We briefly present the two domains of Computational Linguistics which provide the linguistic background for this paper, and refer the reader to the literature for more details.

2.1 Vector space models of meaning

The key idea behind vector space models of meaning [36] can be summed up by Firth's oft-quoted dictum that "you shall know a word by the company it keeps". The basic idea is that the meaning of a word can be determined by the words which appear in its contexts, where context can be a simple n-word window, or the argument slots of grammatical relations, such as the direct object of the verb *eat*. Intuitively, *cat* and *dog* have similar meanings (in some sense) because cats and dogs sleep, run, walk; cats and dogs can be bought, cleaned, stroked; cats and dogs can be small, big, furry. This intuition is reflected in text because *cat* and *dog* appear as the subject of *sleep*, *run*, *walk*; as the direct object of *bought*, *cleaned*, *stroked*; and as the modifiee of *small*, *big*, *furry*.

Meanings of words can be represented as vectors in a high-dimensional "meaning space", in which the orthogonal basis vectors are represented by context words. To give a simple example, if the basis vectors correspond to *eat*, *sleep*, and *run*, and the word *dog* has *eat* in its context 6 times (in some text), *sleep* 5 times, and *run*

7 times, then the vector for *dog* in this space is (6,5,7).[1] The advantage of representing meanings in this way is that the vector space gives us a notion of distance between words, so that the inner product (or some other measure) can be used to determine how close in meaning one word is to another. Computational models along these lines have been built using large vector spaces (tens of thousands of context words/basis vectors) and large bodies of text (up to a billion words in some experiments). Experiments in constructing thesauri using these methods have been relatively successful. For example, the top 10 most similar nouns to *introduction*, according to the system of [11], are *launch, implementation, advent, addition, adoption, arrival, absence, inclusion, creation*.

The other main approach to representing lexical semantics is through an ontology or semantic network, typically manually created by lexicographers or domain experts. The advantages of vector-based representations over hand-built ontologies are that:

- they are created objectively and automatically from text;
- they allow the representation of gradations of meaning;
- they relate well to experimental evidence indicating that the human cognitive system is sensitive to distributional information [34, 40].

Vector-based models of word meaning have been fruitfully applied to many language processing tasks. Examples include lexicon acquisition [16, 26], word sense discrimination and disambiguation [36, 28], text segmentation [5], language modelling [2], and notably document retrieval [35]. Within cognitive science, vector-based models have been successful in simulating a wide variety of semantic processing tasks ranging from semantic priming [27, 24, 29] to episodic memory [17], and text comprehension [24, 14, 25]. Moreover, the cosine similarities obtained within vector-based models have been shown to substantially correlate with human similarity judgements [29] and word association norms [12, 17].

2.2 Algebra of Pregroups as a type-categorial logic

We provide a brief overview of the algebra of Pregroups from the existing literature and refer the reader for more details to [19, 20, 21, 4].

A *partially ordered monoid* $(P, \leq, \cdot, 1)$ is a partially ordered set, equipped with a monoid multiplication $- \cdot -$ with unit 1, where for $p, q, r \in P$, if $p \leq q$ then we have $r \cdot p \leq r \cdot q$ and $p \cdot r \leq q \cdot r$. A *Pregroup* $(P, \leq, \cdot, 1, (-)^l, (-)^r)$ is a partially

[1] In practice the counts are typically weighted in some way to reflect how informative the contextual element is with respect to the meaning of the target word.

ordered monoid whose each element $p \in P$ has a *left adjoint* p^l and a *right adjoint* p^r, i.e. the following hold:

$$p^l \cdot p \leq 1 \leq p \cdot p^l \qquad \text{and} \qquad p \cdot p^r \leq 1 \leq p^r \cdot p\,.$$

Some properties of interest in a Pregroup are:

- Adjoints are unique.
- Adjoints are order reversing: $p \leq q \implies q^r \leq p^r$ and $q^l \leq p^l$.
- The unit is self adjoint: $1^l = 1 = 1^r$.
- Multiplication is self adjoint:$(p \cdot q)^r = q^r \cdot p^r$ and $(p \cdot q)^l = q^l \cdot p^l$.
- Opposite adjoints annihilate each other: $(p^l)^r = p = (p^r)^l$.
- Same adjoints iterate: $p^{ll}p^l \leq 1 \leq p^r p^{rr}, p^{lll}p^{ll} \leq 1 \leq p^{rr}p^{rrr}, \ldots$.

An example of a Pregorup from arithmetic is the set of all monotone unbounded maps on integers $f\colon \mathbb{Z} \to \mathbb{Z}$. In this Pregroup, function composition is the monoid multiplication and the identity map is its unit, the underlying order on integers lifts to an order on the maps whose Galois adjoints are their Pregroup adjoints, defined:

$$f^l(x) = min\{y \in \mathbb{Z} \mid x \leq f(y)\} \qquad f^r(x) = max\{y \in \mathbb{Z} \mid f(y) \leq x\}$$

Recall that a Lambek Calculus $(P, \leq, \cdot, 1, /, \backslash)$ is also a partially ordered monoid, but there it is the monoid multiplication that has a right $- \backslash -$ and a left $-/-$ adjoint. Roughly speaking, the passage from Lambek Calculus to Pregroups can be thought of as replacing the two adjoints of the monoid multiplication with the two adjoints of the elements. One can define a translation between a Lambek Calculus and a Pregroup by sending $(p\backslash q)$ to $(p^r \cdot q)$ and (p/q) to $(p \cdot q^l)$, and via the lambda calculus correspondence of the former think of the adjoint types of a Pregroup as function arguments.

Pregroups formalize grammar of natural languages in the same way as type-categorial logics do. One starts by fixing a set of basic grammatical roles and a partial ordering between them, then freely generating a Pregroup of these types, the existence of which have been proved. In this paper, we present two examples from English: positive and negative transitive sentences[2], for which we fix the following basic types:

[2]By a negative sentence we mean one with a negation operator, such as *not*, and a positive sentence one without a negation operator.

n: noun	s: declarative statement
j: infinitive of the verb	σ: glueing type

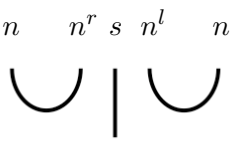
Compound types are formed from these by taking adjoints and juxtaposition. A type (basic or compound) is assigned to each word of the dictionary. We define that if the juxtaposition of the types of the words within a sentence reduces to the basic type s, then the sentence is grammatical. It has been shown that this procedure is decidable. In what follows we use an arrow $\rightarrow$ for $\leq$ and drop the $\cdot$ between juxtaposed types. The example sentence "John likes Mary", has the following type assignment[3]:

John	likes	Mary
n	$(n^r s n^l)$	n

and it is grammatical because of the following reduction:

$$n(n^r s n^l)n \rightarrow 1 s n^l n \rightarrow 1 s 1 \rightarrow s$$

Reductions are depicted diagrammatically, that of the above is:



Reduction diagrams depict the grammatical structure of sentences in one dimension, as opposed to the two dimensional trees of type-categorial logics. This feature becomes useful in applications such as comparing the grammatical patterns of different languages; for some examples see [33].

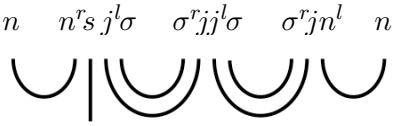
We type the negation of the above sentence as follows:

John	does	not	like	Mary
n	$(n^r s j^l \sigma)$	$(\sigma^r j j^l \sigma)$	$(\sigma^r j n^l)$	n

which is grammatical because of the following reduction:

$$n\,(n^r s j^l \sigma)\,(\sigma^r j j^l \sigma)\,(\sigma^r j n^l)\,n \rightarrow s$$

depicted diagrammatically as follows:



[3]The brackets are only for the purpose of clarity of exposition and are not part of the mathematical presentation.

The types used here for "does" and "not" are not the original ones, e.g. as suggested in [21], but are rather obtainable from the procedure later introduced in [30]. The difference between the two is in the use of the glueing types; once these are deleted from the above, the original types are retrieved. The motivation behind introducing these glueing types is their crucial role in the development of a discourse semantics for Pregroups [30]. Our motivation, as will be demonstrated in section 4, is that these allow for the information to flow and be acted upon in the sentence and as such assist in constructing the meaning of the whole sentence. Interestingly, we have come to realize that these new types can also be obtained by translating into the Pregroup notation the types of the same words from a type-categorial logic approach, up to the replacement of the intermediate n's with σ's.

3 Modeling a language in a concrete category

Our mathematical model of language will be category-theoretic. Category theory is usually not conceived as the most evident part of mathematics, so let us briefly state why this passage is essential. The reader may consult the category theory tutorial [10] which covers the background on the kinds of categories that are relevant here. Also the survey of graphical languages for monoidal categories [38] could be useful – note that Selinger refers to *'non-commutative' compact closed categories* as (both left and right) *planar autonomous categories*. So why do we use categories?

1. The passage from $\{\text{true}, \text{false}\}$-valuations (as in Montague semantics) to quantitative meaning spaces requires a mathematical structure that can store this additional information, but which at the same time retains the compositional structure. Concrete monoidal categories do exactly that:

   - the axiomatic structure, in particular the monoidal tensor, captures compositionality;
   - the concrete objects and corresponding morphisms enable the encoding of the particular model of meaning one uses, here vector spaces.

2. The structural morphisms of the particular categories that we consider, compact closed categories, will be the basic building blocks to construct the morphisms that represent the 'from-meaning-of-words-to-meaning-of-a-sentence'-process.

3. Even in a purely syntactic setting, the lifting to categories will allow us to reason about the grammatical structures of different sentences as first class citizens of the formalism. This will enable us to provide more than just

a yes-no answer about the grammatical structure of a phrase, i.e. if it is grammatical or not. As such, the categorical setting will, for instance, allow us to distinguish and reason about ambiguities in grammatical sentences, where their different grammatical structures gives rise to different meaning interpretations.

We first briefly recall the basic notions of the theory of monoidal categories, before explaining in more detail what we mean by this 'from-meaning-of-words-to-meaning-of-a-sentence'-process.

3.1 Monoidal categories

Here we consider the non-symmetric case of a compact closed category, non-degenerate Pregroups being examples of essentially non-commutative compact closed categories. The formal definition of monoidal categories is somewhat involved. It does admit an intuitive operational interpretation and an elegant, purely diagrammatic calculus. A (strict) monoidal category $\mathbf{C}$ requires the following data and axioms:

- a family $|\mathbf{C}|$ of *objects*;
  - for each ordered pair of objects (A, B) a corresponding set $\mathbf{C}(A, B)$ of *morphisms*; it is convenient to abbreviate $f \in \mathbf{C}(A, B)$ by $f : A \to B$;
  - for each ordered triple of objects (A, B, C), each $f : A \to B$, and $g : B \to C$, there is a *sequential composite* $g \circ f : A \to C$; we moreover require that:
$$(h \circ g) \circ f = h \circ (g \circ f)\,;$$
  - for each object A there is an *identity morphism* $1_A : A \to A$; for $f : A \to B$ we moreover require that:
$$f \circ 1_A = f \qquad \text{and} \qquad 1_B \circ f = f\,;$$
- for each ordered pair of objects (A, B) a *composite object* $A \otimes B$; we moreover require that:
$$(A \otimes B) \otimes C = A \otimes (B \otimes C)\,; \tag{1}$$
- there is a *unit object* I which satisfies:
$$\mathrm{I} \otimes A = A = A \otimes \mathrm{I}\,; \tag{2}$$

- for each ordered pair of morphisms $(f : A \to C, g : B \to D)$ a *parallel composite* $f \otimes g : A \otimes B \to C \otimes D$; we moreover require *bifunctoriality* i.e.

$$(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2)\,. \tag{3}$$

There is a very intuitive operational interpretation of monoidal categories. We think of the objects as *types of systems*. We think of a morphism $f : A \to B$ as a *process* which takes a system of type A as input and provides a system of type B as output, i.e. given any state ψ of the system of type A, it produces a state $f(\psi)$ of the system of type B. Composition of morphisms is sequential application of processes. The compound type $A \otimes B$ represents *joint systems*. We think of I as the trivial system, which can be either 'nothing' or 'unspecified'. More on this intuitive interpretation can be found in [8, 10].

Morphisms $\psi : \mathrm{I} \to A$ are called *elements* of A. At first this might seem to be a double use of terminology: if A were to be a set, then $x \in A$ would be an element, rather than some function $x : \mathrm{I} \to A$. However, one easily sees that elements in $x \in A$ are in bijective correspondence with functions $x : \mathrm{I} \to A$ provided one takes I to be a singleton set. The same holds for vectors $\overrightarrow{v} \in V$, where V is a vector space, and linear maps $\overrightarrow{v} : \mathbb{R} \to V$. In this paper we take the liberty to jump between these two representations of a vector $\overrightarrow{v} \in V$, when using them to represent meanings.

In the standard definition of monoidal categories the 'strict' equality of eqs. (1,2) is not required but rather the existence of a *natural isomorphism* between $(A \otimes B) \otimes C$ and $A \otimes (B \otimes C)$. We assume strictness in order to avoid *coherence conditions*. This simplification is justified by the fact that each monoidal category is categorically equivalent to a strict one, which is obtained by imposing appropriate congruences. Moreover, the graphical language which we introduce below represents (free) strict monoidal categories. This issue is discussed in detail in [10].

So what is particularly interesting about these monoidal categories is indeed that they admit a graphical calculus in the following sense [38]:

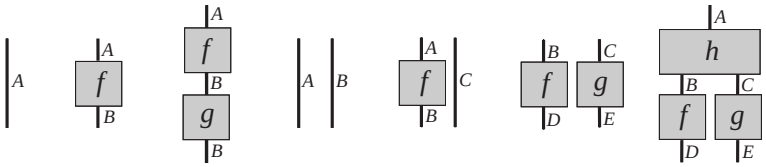
> *An equational statement between morphisms in a monoidal category is provable from the axioms of monoidal categories if and only if it is derivable in the graphical language*.

This fact moreover does not only hold for ordinary monoidal categories, but also for many kinds that have additional structure, including the compact closed categories that we will consider here.

Graphical language for monoidal categories. In the graphical calculus for monoidal categories we depict morphisms by boxes, with incoming and outgoing wires labelled by the corresponding types, with sequential composition depicted by connecting matching outputs and inputs, and with parallel composition depicted by locating boxes side by side. For example, the morphisms

$$1_A \qquad f \qquad g \circ f \qquad 1_A \otimes 1_B \qquad f \otimes 1_C \qquad f \otimes g \qquad (f \otimes g) \circ h$$

are depicted as follows in a top-down fashion:

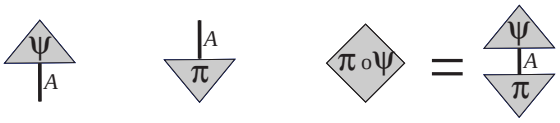


When representing morphisms in this manner by boxes, eq.(3) comes for free [10]!

The unit object I is represented by 'no wire'; for example

$$\psi : \mathrm{I} \to A \qquad \pi : A \to \mathrm{I} \qquad \pi \circ \psi : \mathrm{I} \to \mathrm{I}$$

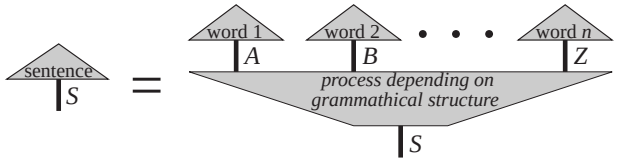
are depicted as:



3.2 The 'from-meaning-of-words-to-meaning-of-a-sentence' process

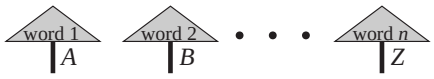
Monoidal categories are widely used to represent processes between systems of varying types, e.g. data types in computer programs. The process which is central to this paper is the one which takes the meanings of words as its input and produces the meaning of a sentence as output, within a fixed type S (Sentence) that allows the representation of meanings of all well-typed sentences.

Diagrammatically we represent it as follows:



where all triangles represent meanings, both of words and sentences. For example, the triangle labeled 'word 1' represents the meaning of word 1 which is of grammatical type A, and the triangle labeled 'sentence' represents the meaning of the

whole sentence. The concatenation (word 1)$\cdot \ldots \cdot$ (word n) is the sentence itself, which is of grammatical type $A \otimes \ldots \otimes Z$, and the way in which the list of meanings of words:



becomes the meaning of a sentence:

within the fixed type S, is mediated by the grammatical structure. The concrete manner in which grammatical structure performs this role will be explained below. This method will exploit the common mathematical structure which vector spaces (used to assign meanings to words in a language) and Pregroups (used to assign grammatical structure to sentences) share, namely compact closure.

3.3 Compact closed categories

A monoidal category is *compact closed* if for each object A there are also objects A^r and A^l, and morphisms

$$\eta^l : \mathrm{I} \to A \otimes A^l \quad \epsilon^l : A^l \otimes A \to \mathrm{I} \quad \eta^r : \mathrm{I} \to A^r \otimes A \quad \epsilon^r : A \otimes A^r \to \mathrm{I}$$

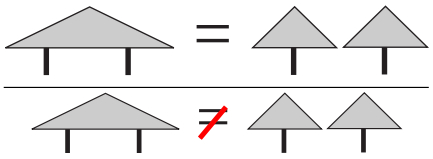
which satisfy:

$$(1_A \otimes \epsilon^l) \circ (\eta^l \otimes 1_A) = 1_A \qquad (\epsilon^r \otimes 1_A) \circ (1_A \otimes \eta^r) = 1_A$$

$$(\epsilon^l \otimes 1_{A^l}) \circ (1_{A^l} \otimes \eta^l) = 1_{A^l} \quad (1_{A^r} \otimes \epsilon^r) \circ (\eta^r \otimes 1_{A^r}) = 1_{A^r}$$

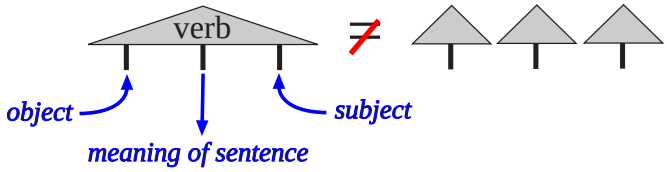
Compact closed categories are in a sense orthogonal to cartesian categories, such as the category of sets and functions with the cartesian product as the monoidal structure. Diagrammatically, in a cartesian category the triangles representing meanings of type $A \otimes B$ could always be decomposed into a triangle representing meanings of type A and a triangle representing meanings of type B:

$$\frac{\textbf{Cartesian}}{\textbf{non-Cartesian}} =$$



But if we consider a verb, then its grammatical type is $n^r s n^l$, that is, of the form $N \otimes S \otimes N$ within the realm of monoidal categories. Clearly, to compute the

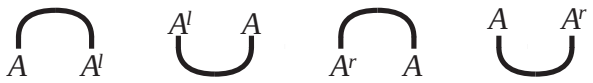
meaning of the whole sentence, the meaning of the verb will need to *interact* with the meaning of both the object and subject, so it cannot be decomposed into three disconnected entities:



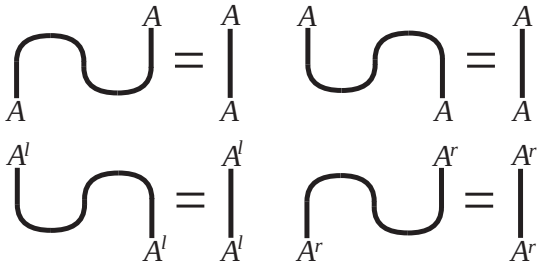
In this graphical language, the topology (i.e. either being connected or not) represents when interaction occurs. In other words, 'connectedness' encodes 'correlations'.

That we cannot always decompose triangles representing meanings of type $A \otimes B$ in compact closed categories can be immediately seen in the graphical calculus of compact closed categories, which explicitly introduces wires between different types, and these will mediate flows of information between words in a sentence. A fully worked out example of sentences of this type is given in section 4.1.

Graphical language for compact closed categories. When depicting the morphisms $\eta^l, \epsilon^l, \eta^r, \epsilon^r$ as (read in a top-down fashion)



rather than as triangles, the axioms of compact closure simplify to:



i.e. they boil down to 'yanking wires'.

Vector spaces, linear maps and tensor product as a compact closed category. Let **FVect** be the category which has vector spaces over the base field $\mathbb{R}$ as objects, linear maps as morphisms and the vector space tensor product as the monoidal tensor. In this category, the tensor is commutative, i.e. $V \otimes W \cong W \otimes V$, and left and right adjoints are the same, i.e. $V^l = V^r$ so we denote either by V^*, which

is the identity maps, i.e. $V^* = V$. To simplify the presentation we assume that each vector space comes with an inner product, that is, it is an *inner-product space*. For the case of vector space models of meaning this is always the case, since we consider a fixed base, and a fixed base canonically induces an inner-product. The reader can verify that compact closure arises, given a vector space V with base $\{\overrightarrow{e}_i\}_i$, by setting $V^l = V^r = V$,

$$\eta^l = \eta^r : \mathbb{R} \to V \otimes V :: 1 \mapsto \sum_i \overrightarrow{e}_i \otimes \overrightarrow{e}_i \tag{4}$$

and

$$\epsilon^l = \epsilon^r : V \otimes V \to \mathbb{R} :: \sum_{ij} c_{ij}\, \overrightarrow{v}_i \otimes \overrightarrow{w}_j \mapsto \sum_{ij} c_{ij} \langle \overrightarrow{v}_i | \overrightarrow{w}_j \rangle \,. \tag{5}$$

In equation 4 we have that $\epsilon^l = \epsilon^r$ is the inner-product extended by linearity to the whole tensor product. Recall that if $\{\overrightarrow{e}_i\}_i$ is a base for V and if $\{\overrightarrow{e}'_i\}_i$ is a base for W then $\{\overrightarrow{e}_i \otimes \overrightarrow{e}'_j\}_{ij}$ is a base for $V \otimes W$. In the base $\{\overrightarrow{e}_i \otimes \overrightarrow{e}_j\}_{ij}$ for $V \otimes V$ the linear map $\epsilon^l = \epsilon^r : V \otimes V \to \mathbb{R}$ has as its matrix the row vector which has entry 1 for the base vectors $\overrightarrow{e}_i \otimes \overrightarrow{e}_i$ and which has entry 0 for the base vectors $\overrightarrow{e}_i \otimes \overrightarrow{e}_j$ with $i \neq j$. The matrix of $\eta^l = \eta^r$ is the column vector obained by transposition.

In eq. (5), the weighted sum $\sum_{ij} c_{ij}\, \overrightarrow{v}_i \otimes \overrightarrow{w}_j$ denotes a typical vector in a tensor space $V \otimes W$, where c_{ij}'s enumerate all possible weights for the tensored pair of base vectors $\overrightarrow{v}_i \otimes \overrightarrow{w}_j$. If in the definition of $\epsilon^l = \epsilon^r$ we apply the restriction that $\overrightarrow{v}_i = \overrightarrow{w}_i = \overrightarrow{e}_i$, which we can do if we stipulate that $\epsilon^l = \epsilon^r$ is a linear map, then it simplifies to

$$\epsilon^l = \epsilon^r : V \otimes V \to \mathbb{R} :: \sum_{ij} c_{ij}\, \overrightarrow{e}_i \otimes \overrightarrow{e}_j \mapsto \sum_i c_{ii} \,.$$

A Pregroup as a compact closed category. A Pregroup is an example of a *posetal category*, that is, a category which is also a poset. For a category this means that for any two objects there is either one or no morphism between them. In the case that this morphism is of type $A \to B$ then we write $A \leq B$, and in the case it is of type $B \to A$ we write $B \leq A$. The reader can then verify that the axioms of a category guarantee that the relation $\leq$ on $|C|$ is indeed a partial order. Conversely, any partially ordered set $(P, \leq)$ is a category. For 'objects' $p, q, r \in P$ we take $[p \leq q]$ to be the singleton $\{p \leq q\}$ whenever $p \leq q$, and empty otherwise. If $p \leq q$ and $q \leq r$ we define $p \leq r$ to be the composite of the 'morphisms' $p \leq q$ and $q \leq r$.

A partially ordered monoid is a monoidal category with the monoid multiplication as tensor on objects; whenever $p \leq r$ and $q \leq z$ then we have $p \cdot q \leq r \cdot z$

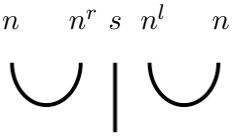
by monotonicity of monoid multiplication, and we define this to be the tensor of 'morphisms' $[p \leq r]$ and $[q \leq z]$. Bifunctoriality, as well as any equational statement between morphisms in posetal categories, is trivially satisfied, since there can only be one morphism between any two objects.

Finally, each Pregroup is a compact closed category for

$$\eta^l = [1 \leq p \cdot p^l] \qquad \epsilon^l = [p^l \cdot p \leq 1]$$

$$\eta^r = [1 \leq p^r \cdot p] \qquad \epsilon^r = [p \cdot p^r \leq 1]$$

and so the required equations are again trivially satisfied. Diagrammatically, the under-links representing the type reductions in a Pregroup grammar are exactly the 'cups' of the compact closed structure. The symbolic counterpart of the diagram of the reduction of a sentence with a transitive verb



is the following morphism:

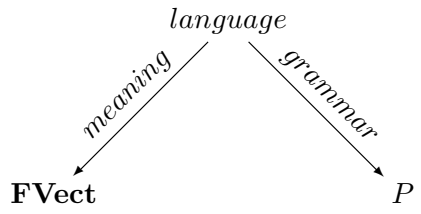
$$\epsilon_n^r \otimes 1_s \otimes \epsilon_n^l : n \otimes n^r \otimes s \otimes n^l \otimes n \to s\,.$$

3.4 Categories representing both grammar and meaning

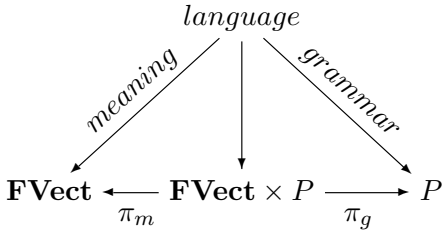
We have described two aspects of natural language which admit mathematical presentations:

1. vector spaces can be used to assign meanings to words in a language;
2. Pregroups can be used to assign grammatical structure to sentences.

When we organize these vector spaces as a monoidal category by also considering linear maps, and tensor products both of vector spaces and linear maps, then these two mathematical objects share common structure, namely compact closure. We can think of these two compact closed structures as two structures that we can *project* out of a language, where P is the free Pregroup generated from the basic types of a natural language:



We aim for a mathematical structure that unifies both of these aspects of language, that is, in which the compositional structure of Pregroups would lift to the level of assigning meaning to sentences and their constituents, or dually, where the structure of assigning meaning to words comes with a mechanism that enables us to compute the meaning of a sentence. The compact closed structure of $\mathbf{FVect}$ alone is too degenerate for this purpose since $A^l = A^r = A$. Moreover, there are canonical isomorphisms $V \otimes W \to W \otimes V$ which translate to posetal categories as $a \cdot b = b \cdot a$, and in general we should not be able to exchange words in a sentence without altering its meaning. Therefore we have to refine types to retain the full grammatical content obtained from the Pregroup analysis. There is an easy way of doing this: rather than objects in $\mathbf{FVect}$ we will consider objects in the product category $\mathbf{FVect} \times P$:



Explicitly, $\mathbf{FVect} \times P$ is the category which has pairs (V, a) with V a vector space and $a \in P$ a grammatical type as objects, and the following pairs as morphisms:

$$(f : V \to W \, , \, p \leq q) \, ,$$

which we can also write as

$$(f, \leq) : (V, p) \to (W, q).$$

Note that if $p \not\leq q$ then there are no morphisms of type $(V, p) \to (W, q)$. It is easy to verify that the compact closed structure of $\mathbf{FVect}$ and P lifts componentwise to one on $\mathbf{FVect} \times P$. The structural morphisms in this new category are now:

$$(\eta^l, \leq) : (\mathbb{R}, 1) \to (V \otimes V, \, p \cdot p^l) \qquad (\eta^r, \leq) : (\mathbb{R}, 1) \to (V \otimes V, \, p^r \cdot p)$$

$$(\epsilon^l, \leq) : (V \otimes V, \, p^l \cdot p) \to (\mathbb{R}, 1) \qquad (\epsilon^r, \leq) : (V \otimes V, \, p \cdot p^r) \to (\mathbb{R}, 1)$$

3.5 Meaning of a sentence as a morphism in $\mathbf{FVect} \times P$.

Definition 3.1. *We refer to an object (W, p) of $\mathbf{Fvect} \times P$ as a* meaning space . *This consists of a vector space W in which the meaning of a word lives $\overrightarrow{w} \in W$ and the grammatical type p of the word.*

Definition 3.2. *We define the vector* $\overrightarrow{w_1 \cdots w_n}$ *of the meaning of a string of words* $w_1 \cdots w_n$ *to be*

$$\overrightarrow{w_1 \cdots w_n} := f(\overrightarrow{w_1} \otimes \cdots \otimes \overrightarrow{w_n})$$

where for (W_i, p_i) *meaning space of the word* w_i*, the linear map* f *is built by substituting each* p_i *in* $[p_1 \cdots p_n \leq x]$ *with* W_i*.*

Thus for $\alpha = [p_1 \cdots p_n \to x]$ a morphism in $\mathbf{P}$ and $f = \alpha[p_i \setminus W_i]$ a linear map in $\mathbf{Fvect}$, the following is a morphism in $\mathbf{Fvect} \times P$:

$$(W_1 \otimes \cdots \otimes W_n,\ p_1 \cdots p_n) \xrightarrow{(f, \leq)} (X, x)$$

We call f the 'from-meaning-of-words-to-meaning-of-a-sentence' map.

According to this formal definition, the procedure of assigning meaning to a string of words can be roughly described as follows:

1. Assign a grammatical type p_i to each word w_i of the string, apply the axioms and rules of the Pregroup grammar to reduce these types to a simpler type $p_1 \cdots p_n \to x$. If the string of words is a sentence, then the reduced type x should be the basic grammatical type s of a sentence[4].

2. Assign a vector space to each word of the sentence based on its syntactic type assignment. For the purpose of this paper, we prefer to be flexible with the manner in which these vector spaces are built, e.g. the vector spaces of the words with basic types like noun may be atomic and built according to the usual rules of the distributional model; the vector spaces of the words with compound types like verbs are tensor spaces.

3. Consider the vector of the meaning of each word in the spaces built above, take their tensor, and apply to it the diagram of the syntactic reduction of the string, according to the meaning spaces of each word. This will provide us with the meaning of the string.

3.6 Comparison with the connectionist proposal

Following the solution of connectionists [39], Pulman and the third author argued for the use of tensor products in developing a compositional distributional model of meaning [6]. They suggested that to implement this idea in linguistics one can,

[4]By Lambek's switching lemma [19] the epsilon maps suffice for the grammatical reductions and thus x already exists in the type of one of the words in the string.

for example, traverse the parse tree of a sentence and tensor the vectors of the meanings of words with the vectors of their roles:

$$(\overrightarrow{John} \otimes \overrightarrow{subj}) \otimes \overrightarrow{likes} \otimes (\overrightarrow{Mary} \otimes \overrightarrow{obj})$$

This vector in the tensor product space should then be regarded as the meaning of the sentence "John likes Mary."

The tensors $(\overrightarrow{John} \otimes \overrightarrow{subj})$ and $(\overrightarrow{Mary} \otimes \overrightarrow{obj})$ in the above are pure tensors, and thus can be considered as a pair of vectors, i.e. $(\overrightarrow{John}, \overrightarrow{subj})$ and $(\overrightarrow{Mary}, \overrightarrow{obj})$. These are pairs of a meaning of a word and its grammatical role, and almost the same as the pairs considered in our approach, i.e. that of a meaning space of each word. A minor difference is that, in the above, the grammatical role $\overrightarrow{p}$ is a genuine vector, whereas in our approach this remains a grammatical type. If needed, our approach can easily be adapted to also allow types to be represented in a vector space.

A more conceptual difference between the two approaches lies in the fact that the above does not assign a grammatical type to the verb, i.e. treats $\overrightarrow{likes}$ as a single vector. Whereas in our approach, the vector of the verb itself lives in a tensor space.

4 Computing the meaning of example sentences

In what follows we use the steps above to assign meaning to positive and negative transitive sentences[5].

4.1 Positive Transitive Sentence

A positive sentence with a transitive verb has the Pregroup type $n(n^r s n^l)n$. We assume that the meaning spaces of the subject and object of the sentence are atomic and are given as (V, n) and (W, n). The meaning space of the verb is compound and is given as $(V \otimes S \otimes W, n^r s n^l)$. The 'from-meaning-of-words-to-meaning-of-a-sentence' linear map f is the linear map which realizes the following structural morphism in $\mathbf{FVect} \times P$:

$$\left(V \otimes T \otimes W\,, n(n^r s n^l)n\right) \xrightarrow{(f,\leq)} (S, s)\,,$$

and arises from a syntactic reduction map; in this case we obtain:

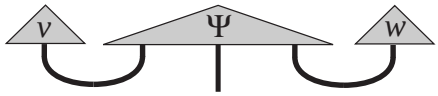
$$f = \epsilon_V \otimes 1_S \otimes \epsilon_W : V \otimes (V \otimes S \otimes W) \otimes W \to S\,.$$

[5]For the negative example, we use the idea and treatment of previous work [31], in that we use eta maps to interpret the logical meaning of "does" and "not", but extend the details of calculations, diagrammatic representations, and corresponding comparisons.

Noting the isomorphism $V \otimes S \otimes W \cong V \otimes W \otimes S \cong V^* \otimes W^* \to S$ obtained from the commutativity of tensor in the $\mathbf{FVect}$ and that $V^* = V$ and $W^* = W$ therein, and the universal property of the tensor with respect to product, we can think about the meaning space of a verb $V \otimes W \otimes S$ as a function space $V \times W \to S$. So the meaning vector of each transitive verb can be thought of as a function that inputs a subject from V and an object from W and outputs a sentence in S.

In the graphical calculus, the linear map of meaning is depicted as follows:

The matrix of f has $dim(V)^2 \times dim(S) \times dim(W)^2$ columns and $dim(S)$ rows, and its entries are either 0 or 1. When applied to the vectors of the meanings of the words, i.e. $f(\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w}) \in S$ for $\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w} \in V \otimes S \otimes W$ we obtain, diagrammatically:



This map can be expressed in terms of the inner-product as follows. Consider the typical vector in the tensor space which represents the type of verb:

$$\Psi = \sum_{ijk} c_{ijk} \overrightarrow{v}_i \otimes \overrightarrow{s}_j \otimes \overrightarrow{w}_k \in V \otimes S \otimes W$$

then

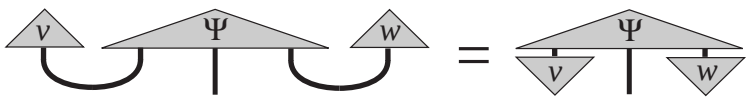
$$\begin{aligned} f(\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w}) &= \epsilon_V \otimes 1_S \otimes \epsilon_W(\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w}) \\ &= \sum_{ijk} c_{ijk} \langle \overrightarrow{v} | \overrightarrow{v}_i \rangle \overrightarrow{s}_j \langle \overrightarrow{w}_k | \overrightarrow{w} \rangle \\ &= \sum_j \left(\sum_{ik} c_{ijk} \langle \overrightarrow{v} | \overrightarrow{v}_i \rangle \langle \overrightarrow{w}_k | \overrightarrow{w} \rangle \right) \overrightarrow{s}_j \,. \end{aligned}$$

This vector is the meaning of the sentence of type $n(n^r s n^l)n$, and assumes as given the meanings of its constituents $\overrightarrow{v} \in V$, $\overrightarrow{\Psi} \in T$ and $\overrightarrow{w} \in W$, obtained from data using some suitable method.

Note that, in Dirac notation, $f(\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w})$ is written as:

$$\left(\langle \epsilon_V^r | \otimes 1_S \otimes \langle \epsilon_V^r | \right) \left| \overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w} \right\rangle .$$

Also, the diagrammatic calculus tells us that:



where the reversed triangles are now the corresponding Dirac-bra's, or in vector space terms, the corresponding functionals in the dual space. This simplifies the expression that we need to compute to:

$$(\langle \overrightarrow{v} | \otimes 1_S \otimes \langle \overrightarrow{v} |)| \overrightarrow{\Psi} \rangle$$

As mentioned in the introduction, our focus in this paper is not on how to practically exploit the mathematical framework, which would require substantial further research, but to expose the mechanisms which govern it. To show that this particular computation (i.e. the 'from-meaning-of-words-to-meaning-of-a-sentence'-process) does indeed produce a vector which captures the meaning of a sentence, we explicitly compute $f(\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w})$ for some simple examples, with the intention of providing the reader with some insight into the underlying mechanisms and how the approach relates to existing frameworks.

Example 1. One Dimensional Truth-Theoretic Meaning. Consider the sentence

$$\textit{John likes Mary}. \tag{6}$$

We encode this sentence as follows; we have:

$$\overrightarrow{\textit{John}} \in V, \quad \overrightarrow{\textit{likes}} \in T, \quad \overrightarrow{\textit{Mary}} \in W$$

where we take V to be the vector space spanned by men and W the vector space spanned by women. In terms of context vectors this means that each word is its own and only context vector, which is of course a far too simple idealisation for practical purposes. We will conveniently assume that all men are referred to as *male*, using indices to distinguish them: m_i. Thus the set of vectors $\{\overrightarrow{m}_i\}_i$ spans V. Similarly every woman will be referred to as *female* and distinguished by f_j, for some j, and the set of vectors $\{\overrightarrow{f}_j\}_j$ spans W. Let us assume that *John* in sentence (6) is m_3 and that *Mary* is f_4.

If we are only interested in the truth or falsity of a sentence, we have two choices in creating the sentence space S: it can be spanned by two basis vectors $|0\rangle$ and $|1\rangle$ representing the truth values of *true* and *false*, or just by a single vector $\overrightarrow{1}$, which we identify with *true*, the origin $\overrightarrow{0}$ is then identified with *false* (so we use Dirac notation for the basis to distinguish between the origin $\overrightarrow{0}$ and the $|0\rangle$ basis vector). This latter approach might feel a little unintuitive, but it enables us

to establish a convenient connection with the relational Montague-style models of meaning, which we shall present in the last section of the paper.

The transitive verb $\overrightarrow{\textit{likes}}$ is encoded as the *superposition*:

$$\overrightarrow{\textit{likes}} = \sum_{ij} \overrightarrow{m_i} \otimes \overrightarrow{\textit{likes}}_{ij} \otimes \overrightarrow{f_j}$$

where $\overrightarrow{\textit{likes}}_{ij} = \overrightarrow{1}$ if m_i likes f_j and $\overrightarrow{\textit{likes}}_{ij} = \overrightarrow{0}$ otherwise. Of course, in practice, the vector that we have constructed here would be obtained automatically from data using some suitable method.

Finally, we obtain:

$$\begin{aligned} f\left(\overrightarrow{m}_3 \otimes \overrightarrow{\textit{likes}} \otimes \overrightarrow{f}_4\right) &= \sum_{ij} \langle \overrightarrow{m}_3 \mid \overrightarrow{m}_i \rangle \, \overrightarrow{\textit{likes}}_{ij} \langle \overrightarrow{f}_j \mid \overrightarrow{f}_4 \rangle \\ &= \sum_{ij} \delta_{3i} \, \overrightarrow{\textit{likes}}_{ij} \, \delta_{j4} \\ &= \overrightarrow{\textit{likes}}_{34} = \begin{cases} \overrightarrow{1} & m_3 \text{ likes } f_4 \\ \overrightarrow{0} & o.w. \end{cases} \end{aligned}$$

So we indeed obtain the correct truth-value meaning of our sentence. We are not restricted to the truth-value meaning; on the contrary, we can have, for example, degrees of meaning, as shown in section 5.

Example 1b. Two Dimensional Truth-Theoretic Meaning. It would be more intuitive to assume that the sentence space S is spanned by two vectors $|0\rangle$ and $|1\rangle$, which stand for *false* and *true* respectively. In this case, the computing of the meaning map proceeds in exactly the same way as in the one dimensional case. The only difference is that when the sentence "John likes Mary" is false, the vector $\textit{likes}_{ij}$ takes the value $|0\rangle$ rather than just the origin $\overrightarrow{0}$, and if it is true it takes the value $|1\rangle$ rather than $\overrightarrow{1}$.

4.2 Negative Transitive Sentence

The types of a sentence with negation and a transitive verb, for example "John does not like Mary", are:

$$n\,(n^r s j^l \sigma)\,(\sigma^r j j^l \sigma)\,(\sigma^r j n^l)\,n$$

Similar to the positive case, we assume the vector spaces of the subject and object are atomic $(V, n), (W, n)$. The meaning space of the auxiliary verb is $(V \otimes S \otimes J \otimes V, n^r s j^l \sigma)$, that of the negation particle is $(V \otimes J \otimes J \otimes V, \sigma^r j j^l \sigma)$, and that

of the verb is $(V \otimes J \otimes W, \sigma^r jn^l)$. The 'from-meaning-of-words-to-meaning-of-a-sentence' linear map f is:

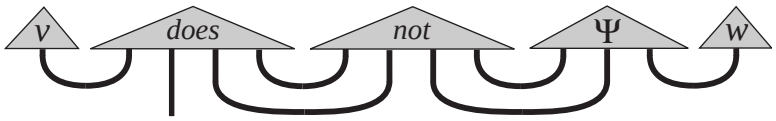
$$f = (1_S \otimes \epsilon_J \otimes \epsilon_J) \circ (\epsilon_V \otimes 1_S \otimes 1_{J^*} \otimes \epsilon_V \otimes 1_J \otimes 1_{J^*} \otimes \epsilon_V \otimes 1_J \otimes \epsilon_W) :$$
$$V \otimes (V^* \otimes S \otimes J^* \otimes V) \otimes (V^* \otimes J \otimes J^* \otimes V) \otimes (V^* \otimes J \otimes W^*) \otimes W \to S$$

and depicted as:

When applied to the meaning vectors of words one obtains:

$$f(\overrightarrow{v} \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w})$$

which is depicted as:



where $\overrightarrow{does}$ and $\overrightarrow{not}$ are the vectors corresponding to the meanings of "does" and "not". Since these are logical function words, we may decide to assign meaning to them manually and without consulting the document. For instance, for *does* we set:

$$S = J \qquad \text{and} \qquad \overrightarrow{does} = \sum_{ij} \overrightarrow{e}_i \otimes \overrightarrow{e}_j \otimes \overrightarrow{e}_j \otimes \overrightarrow{e}_i \in V \otimes J \otimes J \otimes V \,.$$

As explained in section 3.1, vectors in $V \otimes J \otimes J \otimes V$ can also be presented as linear maps of type $\mathbb{R} \to V \otimes J \otimes J \otimes V$, and in the case of *does* we have:

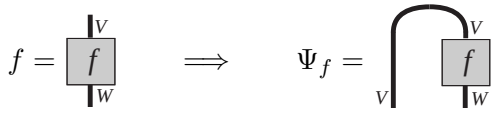
$$\overrightarrow{does} \simeq (1_V \otimes \eta_J \otimes 1_V) \circ \eta_V : \mathbb{R} \to V \otimes J \otimes J \otimes V$$

which shows that we only relied on structural morphisms.

As we will demonstrate in the examples below, by relying only on η-maps, *does* acts very much as an 'identity' with respect to the flow of information between the words in a sentence. This can be formalized in a more mathematical manner. There is a well-known bijective correspondence between linear maps of type $V \to W$ and vectors in $V \otimes W$. Given a linear map $f : V \to W$ then the corresponding vector is:

$$\Psi_f = \sum_i \overrightarrow{e}_i \otimes f(\overrightarrow{e}_i)$$

where $\{\overrightarrow{e}_i\}_i$ is a basis for V. Diagrammatically we have:



Take this linear map to be the identity on V and we obtain η_V.

The trick to implement *not* will be to take this linear map to be the linear matrix representing the logical *not*. Concretely, while the matrix of the identity is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, the matrix of the logical *not* is $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. In Dirac notation, the vector corresponding to the identity is $|00\rangle + |11\rangle$, while the vector corresponding to the logical *not* is $|01\rangle + |10\rangle$. While we have

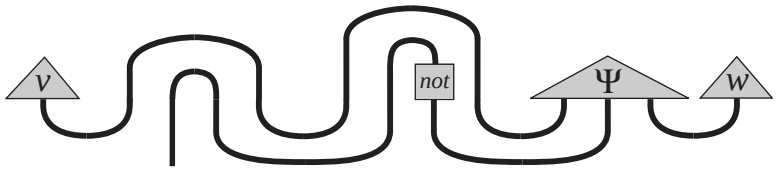
$$\overrightarrow{does} \;=\; \sum_i \overrightarrow{e}_i \otimes (|00\rangle + |11\rangle) \otimes \overrightarrow{e}_i \;\in\; V \otimes J \otimes J \otimes V \,,$$

we will set

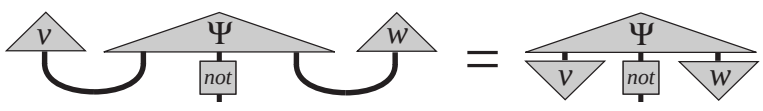
$$\overrightarrow{not} \;=\; \sum_i \overrightarrow{e}_i \otimes (|01\rangle + |10\rangle) \otimes \overrightarrow{e}_i \;\in\; V \otimes J \otimes J \otimes V \,.$$

Diagrammatically we have:

Substituting all of this in $f(\overrightarrow{v} \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w})$ we obtain, diagrammatically:

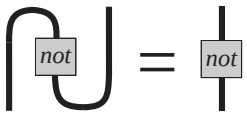


which by the diagrammatic calculus of compact closed categories is equal to:



(7)

since in particular we have that:



where the configuration on the left always encodes the transpose and the matrix of the *not* is obviously self-transpose. In the language of vectors and linear maps, the left hand side of eq. (7) is:

$$\left(\epsilon_V \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \epsilon_W\right)(\overrightarrow{v} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{w})\,.$$

Note that the above pictures are very similar to the ones encountered in [7, 9] which describe quantum informatic protocols such as quantum teleportation and entanglement swapping. There the morphisms η and ϵ encode Bell-states and corresponding measurement projectors.

Example 2. Negative Truth-Theoretic Meaning. The meaning of the sentence

John does not like Mary

is calculated as follows. We assume that the vector spaces $S = J$ are spanned by the two vectors as in Example 1b, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. We assume that $|1\rangle$ stands for *true* and that $|0\rangle$ stands for *false*. Vector spaces V and W are as in the positive case above. The vector of *like* is as before:

$$\overrightarrow{like} = \sum_{ij} \overrightarrow{m_i} \otimes \overrightarrow{like}_{ij} \otimes \overrightarrow{f_j} \qquad \text{for} \qquad \overrightarrow{like}_{ij} = \begin{cases} |1\rangle & m_i \text{ likes } f_j \\ |0\rangle & o.w. \end{cases}$$

Setting $N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ we obtain:

$$(\epsilon_V \otimes N \otimes \epsilon_W)\left(\overrightarrow{m}_3 \otimes \overrightarrow{likes} \otimes \overrightarrow{f}_4\right) = \sum_{ij} \langle \overrightarrow{m}_3 \mid \overrightarrow{m_i} \rangle\, N(\overrightarrow{likes}_{ij}) \langle \overrightarrow{f}_j \mid \overrightarrow{f_4} \rangle =$$

$$\sum_{ij} \delta_{3i}\, N(\overrightarrow{likes}_{ij})\, \delta_{j4} \quad = \quad N(\overrightarrow{likes}_{34}) =$$

$$\begin{cases} |1\rangle & \overrightarrow{like}_{34} = |0\rangle \\ |0\rangle & \overrightarrow{like}_{34} = |1\rangle \end{cases} \quad = \quad \begin{cases} |1\rangle & m_3 \text{ does not like } f_4 \\ |0\rangle & o.w. \end{cases}$$

That is, the meaning of "John does not like Mary" is true if $\overrightarrow{like}_{34}$ is false, i.e. if the meaning of "John likes Mary" is false.

For those readers who are suspicious of our graphical reasoning, here is the full-blown symbolic computation. Abbreviating $|10\rangle + |01\rangle$ to $\overline{n}$ and $|00\rangle + |11\rangle$ to $\overline{d}$, and setting $f = h \circ g$ with

$$h = 1_J \otimes \epsilon_J \otimes \epsilon_J \quad \text{and} \quad g = \epsilon_V \otimes 1_J \otimes 1_J \otimes \epsilon_V \otimes 1_J \otimes 1_J \otimes \epsilon_V \otimes 1_J \otimes \epsilon_W$$

$$\begin{aligned}
f\left(\overrightarrow{m_3} \otimes \big(\sum_l \overrightarrow{m}_l \otimes \overline{d} \otimes \overrightarrow{m}_l\big) \otimes \big(\sum_k \overrightarrow{m}_k \otimes \overline{n} \otimes \overrightarrow{m}_k\big) \otimes \big(\sum_{ij} \overrightarrow{m}_i \otimes \overrightarrow{like}_{ij} \otimes \overrightarrow{f}_j\big) \otimes \overrightarrow{f}_4\right) \\
&= h\left(\sum_{ijkl} \langle \overrightarrow{m}_3 \mid \overrightarrow{m}_l \rangle \, \overline{d} \, \langle \overrightarrow{m}_l \mid \overrightarrow{m}_k \rangle \, \overline{n} \, \langle \overrightarrow{m}_k \mid \overrightarrow{m}_i \rangle \, \overrightarrow{like}_{ij} \langle \overrightarrow{f}_j \mid \overrightarrow{f}_4 \rangle\right) \\
&= h\left(\sum_{ijkl} \delta_{3l} \, \overline{d} \, \delta_{lk} \, \overline{n} \, \delta_{ki} \, \overrightarrow{like}_{ij} \delta_{j4}\right) \\
&= h\left(\overline{d} \otimes \overline{n} \otimes \overrightarrow{like}_{34}\right) \\
&= h\left((|00\rangle + |11\rangle) \otimes (|10\rangle + |01\rangle) \otimes \overrightarrow{like}_{34}\right) \\
&= h\left(|0010\overrightarrow{like}_{34}\rangle + |0001\overrightarrow{like}_{34}\rangle + |1110\overrightarrow{like}_{34}\rangle + |1101\overrightarrow{like}_{34}\rangle\right) \\
&= |0\rangle\langle 0 \mid 1\rangle\langle 0 \mid \overrightarrow{like}_{34}\rangle + |0\rangle\langle 0 \mid 0\rangle\langle 1 \mid \overrightarrow{like}_{34}\rangle + \\
&\quad |1\rangle\langle 1 \mid 1\rangle\langle 0 \mid \overrightarrow{like}_{34}\rangle + |1\rangle\langle 1 \mid 0\rangle\langle 1 \mid \overrightarrow{like}_{34}\rangle \\
&= |0\rangle\langle 1 \mid \overrightarrow{like}_{34}\rangle + |1\rangle\langle 0 \mid \overrightarrow{like}_{34}\rangle \\
&= \begin{cases} |1\rangle & \overrightarrow{like}_{34} = |0\rangle \\ |0\rangle & \overrightarrow{like}_{34} = |1\rangle \end{cases}
\end{aligned}$$

5 Comparing meanings of sentences

One of the advantages of our approach to compositional meaning is that the meanings of sentences are all vectors in the same space, so we can use the inner product to compare the meaning vectors. This measure has been referred to and widely used as a *degree of similarity* between meanings of words in the distributional approaches to meaning [36]. Here we extend it to strings of words as follows.

Definition 5.1. *Two strings of words* $w_1 \cdots w_k$ *and* $w'_1 \cdots w'_l$ have degree of similarity m *iff their Pregroup reductions result in the same grammatical type*[6]

[6] If one wishes to do so, meaning of phrases that do not have the same grammatical types can also be compared, but only after transferring them to a common dummy space.

and we have

$$\frac{1}{N \times N'} \left\langle f(\overrightarrow{w_1} \otimes \cdots \otimes \overrightarrow{w_k}) \mid f(\overrightarrow{w'_1} \otimes \cdots \otimes \overrightarrow{w'_l}) \right\rangle = m$$

for

$$N = \mid f(\overrightarrow{w_1} \otimes \cdots \otimes \overrightarrow{w_k}) \mid \qquad N' = \mid f(\overrightarrow{w'_1} \otimes \cdots \otimes \overrightarrow{w'_l}) \mid$$

where $\mid \overrightarrow{v} \mid$ *is the norm of* $\overrightarrow{v}$*, that is,* $\mid \overrightarrow{v} \mid^2 = \langle \overrightarrow{v} \mid \overrightarrow{v} \rangle$*, and* f, f' *are the meaning maps defined according to definition 3.2*

Thus we use this tool to compare meanings of positive sentences to each other, meanings of negative sentences to each other, and more importantly meanings of positive sentences to negative ones. For example, we compare the meaning of "John likes Mary" to "John loves Mary", the meaning of "John does not like Mary" to "John does not love Mary", and also the meaning of the latter two sentences to "John likes Mary" and "John loves Mary". To make the examples more interesting, we assume that "likes" has degrees of both "love" and "hate".

Example 3. Hierarchical Meaning. Similar to before, we have:

$$\overrightarrow{loves} = \sum_{ij} \overrightarrow{m_i} \otimes \overrightarrow{loves}_{ij} \otimes \overrightarrow{f_j} \qquad \overrightarrow{hates} = \sum_{ij} \overrightarrow{m_i} \otimes \overrightarrow{hates}_{ij} \otimes \overrightarrow{f_j}$$

where $\overrightarrow{loves}_{ij} = |1\rangle$ if m_i loves f_j and $\overrightarrow{loves}_{ij} = |0\rangle$ otherwise, and $\overrightarrow{hates}_{ij} = |1\rangle$ if m_i hates f_j and $\overrightarrow{hates}_{ij} = |0\rangle$ otherwise. Define *likes* to have degrees of *love* and *hate* as follows:

$$\overrightarrow{likes} = \frac{3}{4}\overrightarrow{loves} + \frac{1}{4}\overrightarrow{hates} = \sum_{ij} \overrightarrow{m_i} \otimes \left(\frac{3}{4}\overrightarrow{loves}_{ij} + \frac{1}{4}\overrightarrow{hates}_{ij}\right) \otimes \overrightarrow{f_j}$$

The meaning of our example sentence is thus obtained as follows:

$$\begin{aligned}
& f\left(\overrightarrow{m}_3 \otimes \overrightarrow{likes} \otimes \overrightarrow{f}_4\right) = f\left(\overrightarrow{m}_3 \otimes \left(\frac{3}{4}\overrightarrow{loves} + \frac{1}{4}\overrightarrow{hates}\right) \otimes \overrightarrow{f}_4\right) \\
&= \sum_{ij} \langle \overrightarrow{m}_3 \mid \overrightarrow{m_i} \rangle \left(\frac{3}{4}\overrightarrow{loves}_{ij} + \frac{1}{4}\overrightarrow{hates}_{ij}\right) \left\langle \overrightarrow{f}_j \mid \overrightarrow{f_4} \right\rangle \\
&= \sum_{ij} \delta_{3i} \left(\frac{3}{4}\overrightarrow{loves}_{ij} + \frac{1}{4}\overrightarrow{hates}_{ij}\right) \delta_{j4} \\
&= \frac{3}{4}\overrightarrow{loves}_{34} + \frac{1}{4}\overrightarrow{hates}_{34}
\end{aligned}$$

Example 4. Negative Hierarchical Meaning. To obtain the meaning of "John does not like Mary" in this case, one inserts $\frac{3}{4}\overrightarrow{loves}_{ij} + \frac{1}{4}\overrightarrow{hates}_{ij}$ for $\overrightarrow{likes}_{ij}$ in the calculations and one obtains:

$$h\left(\overline{d}\otimes\overline{n}\otimes\left(\frac{3}{4}\overrightarrow{loves}_{34}+\frac{1}{4}\overrightarrow{hates}_{34}\right)\right) \;=\; \frac{1}{4}\overrightarrow{\textit{loves}}_{34}+\frac{3}{4}\overrightarrow{\textit{hates}}_{34}$$

That is, the meaning of "John does not like Mary" is the vector obtained from the meaning of "John likes Mary" by swapping the basis vectors.

Example 5. Degree of similarity of positive sentences. The meanings of the distinct verbs $loves$, $likes$ and $hates$ in the different sentences propagate through the reduction mechanism and reveal themselves when computing inner-products between sentences in the sentence space. For instance, the sentence "John loves Mary" and "John likes Mary" have a degree of similarity of 3/4, calculated as follows:

$$\left\langle f\big(\overrightarrow{m}_3\otimes\overrightarrow{\textit{loves}}\otimes\overrightarrow{f}_4\big)\;\middle|\; f\big(\overrightarrow{m}_3\otimes\overrightarrow{\textit{likes}}\otimes\overrightarrow{f}_4\big)\right\rangle = \left\langle\overrightarrow{\textit{loves}}_{34}\mid\overrightarrow{\textit{likes}}_{34}\right\rangle$$

In the above, we expand the definition of $\overrightarrow{\textit{likes}}_{34}$ and obtain:

$$\left\langle\overrightarrow{\textit{loves}}_{34}\mid\tfrac{3}{4}\overrightarrow{\textit{loves}}_{34}+\tfrac{1}{4}\overrightarrow{\textit{hates}}_{34}\right\rangle =$$

$$\frac{3}{4}\left\langle\overrightarrow{\textit{loves}}_{34}\mid\overrightarrow{\textit{loves}}_{34}\right\rangle+\frac{1}{4}\left\langle\overrightarrow{\textit{loves}}_{34}\mid\overrightarrow{\textit{hates}}_{34}\right\rangle$$

and since $\overrightarrow{\textit{loves}}_{34}$ and $\overrightarrow{\textit{hates}}_{34}$ are always orthogonal, that is, if one is $|1\rangle$ then the other one is $|0\rangle$, we have that

$$\left\langle\overrightarrow{\textit{John loves Mary}}\mid\overrightarrow{\textit{John likes Mary}}\right\rangle=\frac{3}{4}\mid\overrightarrow{\textit{loves}}_{34}\mid^2$$

Hence the degree of similarity of these sentences is $\frac{3}{4}$. A similar calculation provides us with the following degrees of similarity. For notational simplicity we drop the square of norms from now on, i.e. we implicitly normalize meaning vectors.

$$\left\langle\overrightarrow{\textit{John hates Mary}}\mid\overrightarrow{\textit{John likes Mary}}\right\rangle =$$

$$\left\langle f\big(\overrightarrow{m}_3\otimes\overrightarrow{\textit{hates}}\otimes\overrightarrow{f}_4\big)\;\middle|\; f\big(\overrightarrow{m}_3\otimes\overrightarrow{\textit{likes}}\otimes\overrightarrow{f}_4\big)\right\rangle \;=\frac{1}{4}$$

$$\left\langle\overrightarrow{\textit{John loves Mary}}\mid\overrightarrow{\textit{John hates Mary}}\right\rangle =$$

$$\left\langle f\big(\overrightarrow{m}_3\otimes\overrightarrow{\textit{loves}}\otimes\overrightarrow{f}_4\big)\;\middle|\; f\big(\overrightarrow{m}_3\otimes\overrightarrow{\textit{hates}}\otimes\overrightarrow{f}_4\big)\right\rangle = 0\,.$$

Example 6. Degree of similarity of negative sentences. In the negative case, the meaning of the composition of the meanings of the auxiliary and negation markers ("does not"), applied to the meaning of the verb, propagates through the computations and defines the cases of the inner product. For instance, the sentences "John does not love Mary" and "John does not like Mary" have a degree of similarity of 3/4, calculated as follows:

$$\begin{aligned}
&\left\langle \overrightarrow{\textit{John does not love Mary}} \;\middle|\; \overrightarrow{\textit{John does not like Mary}} \right\rangle = \\
&\left\langle f\big(\overrightarrow{m}_3 \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{love} \otimes \overrightarrow{f}_4\big) \;\middle|\; f\big(\overrightarrow{m}_3 \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{like} \otimes \overrightarrow{f}_4\big) \right\rangle \\
&= \left\langle \overrightarrow{hates}_{34} \;\middle|\; \tfrac{1}{4}\overrightarrow{loves}_{34} + \tfrac{3}{4}\overrightarrow{hates}_{34} \right\rangle \\
&= \tfrac{1}{4}\left\langle \overrightarrow{hates}_{34} \;\middle|\; \overrightarrow{loves}_{34} \right\rangle + \tfrac{3}{4}\left\langle \overrightarrow{hates}_{34} \;\middle|\; \overrightarrow{hates}_{34} \right\rangle = \tfrac{3}{4}
\end{aligned}$$

Example 7. Degree of similarity of positive and negative sentences. Here we compare the meanings of positive and negative sentences. This is perhaps of special interest to linguists of distributional meaning, since these sentences do not have the same grammatical structure. That we can compare these sentences shows that our approach does not limit us to the comparison of meanings of sentences that have the same grammatical structure. We have:

$$\left\langle f\big(\overrightarrow{m}_3 \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{like} \otimes \overrightarrow{f}_4\big) \;\middle|\; f\big(\overrightarrow{m}_3 \otimes \overrightarrow{loves} \otimes \overrightarrow{f}_4\big) \right\rangle = \frac{1}{4}$$

$$\left\langle f\big(\overrightarrow{m}_3 \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{like} \otimes \overrightarrow{f}_4\big) \;\middle|\; f\big(\overrightarrow{m}_3 \otimes \overrightarrow{hates} \otimes \overrightarrow{f}_4\big) \right\rangle = \frac{3}{4}$$

The following is the most interesting case:

$$\begin{aligned}
&\left\langle f\big(\overrightarrow{m}_3 \otimes \overrightarrow{does} \otimes \overrightarrow{not} \otimes \overrightarrow{like} \otimes \overrightarrow{f}_4\big) \;\middle|\; f\big(\overrightarrow{m}_3 \otimes \overrightarrow{likes} \otimes \overrightarrow{f}_4\big) \right\rangle \\
&= \left\langle \frac{1}{4}\overrightarrow{loves}_{34} + \frac{3}{4}\overrightarrow{hates}_{34} \;\middle|\; \frac{3}{4}\overrightarrow{loves}_{34} + \frac{1}{4}\overrightarrow{hates}_{34} \right\rangle \\
&= (\frac{1}{4} \times \frac{3}{4})\left\langle \overrightarrow{loves}_{34} \;\middle|\; \overrightarrow{loves}_{34} \right\rangle + (\frac{3}{4} \times \frac{1}{4})\left\langle \overrightarrow{hates}_{34} \;\middle|\; \overrightarrow{hates}_{34} \right\rangle \\
&= (\frac{1}{4} \times \frac{3}{4}) + (\frac{3}{4} \times \frac{1}{4}) = \frac{3}{8}
\end{aligned}$$

This value might feel non-intuitive, since one expects that "like" and "does not like" have zero intersection in their meanings. This would indeed be the case had we used our original truth-value definitions. But since we have set "like" to have degrees of "love" and "hate", their intersection will no longer be 0.

Using the same method, one can form and compare meanings of many different types of sentences. In a full-blown vector space model, which has been automatically extracted from large amounts of text, we obtain 'imperfect' vector representations for words, rather than the 'ideal' ones presented here. But the mechanism of how the meanings of words propagate to the meanings of sentences remains the same.

6 Relations vs Vectors for Montague-style semantics

When fixing a base for each vector space we can think of $\mathbf{FVect}$ as a category of which the morphisms are matrices expressed in this base. These matrices have real numbers as entries. It turns out that if we consider matrices with entries not in $(\mathbb{R}, +, \times)$, but in any other semiring[7] $(R, +, \times)$, we again obtain a compact closed category. This semiring does not have to be a field, and can for example be the positive reals $(\mathbb{R}^+, +, \times)$, positive integers $(\mathbb{N}, +, \times)$ or even Booleans $(\mathbb{B}, \vee, \wedge)$.

In the case of $(\mathbb{B}, \vee, \wedge)$, we obtain an isomorphic copy of the category $\mathbf{FRel}$ of finite sets and relations with the cartesian product as tensor, as follows. Let X be a set whose elements we have enumerated as $X = \{x_i \mid 1 \leq i \leq |X|\}$. Each element can be seen as a column with a 1 at the row equal to its number and 0 in all other rows. Let $Y = \{y_j \mid 1 \leq j \leq |Y|\}$ be another enumerated set. A relation $r \subseteq X \times Y$ is represented by an $|X| \times |Y|$ matrix, where the entry in the ith column and jth row is 1 iff $(x_i, y_j) \in r$ or else 0. The composite $s \circ r$ of relations $r \subseteq X \times Y$ and $s \subseteq Y \times Z$ is

$$\{(x, z) \mid \exists y \in Y : (x, y) \in r, (y, z) \in s\}.$$

The reader can verify that this composition induces matrix multiplication of the corresponding matrices.

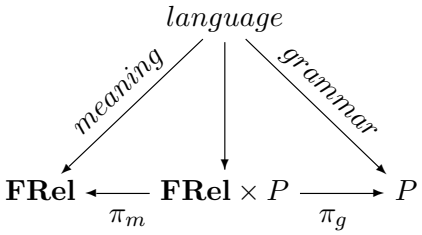
Interestingly, in the world of relations (but not functions) there is a notion of *superposition* [8]. The relations of type $r \subseteq \{*\} \times X$ (in matricial terms, all column vectors with 0's and 1's as entries) are in bijective correspondence with the subsets of X via the correspondence

$$r \mapsto \{x \in X \mid (*, x) \in r\}.$$

Each such subset can be seen as the superposition of the elements it contains. The *inner-product* of two subsets is 0 if they are disjoint and 1 if they have a non-empty intersection. So we can think of two disjoint sets as being *orthogonal*.

[7] A semiring is a set together with two operations, addition and multiplication, for which we have a distributive law but no additive nor multiplicative inverses. Having an addition and multiplication of this kind suffices to have a matrix calculus.

Since the abstract nature of our procedure for assigning meaning to sentences did not depend on the particular choice of $\mathbf{FVect}$ we can now repeat it for the following situation:



In $\mathbf{FRel}\times P$ we recover a Montague-style Boolean semantics. The vector spaces in this setting are encodings of sets of individuals and relations over these sets. Inner products take intersections between the sets and eta maps produce new relations by connecting pairs that are not necessarily side by side.

In all our examples so far, the vector spaces of subject and object were essentially sets that were encoded in a vector space framework. This was done by assuming that each possible male subject is a base in the vector space of males and similarly for the female objects. That is why the meaning in these examples was a truth-theoretic one. We repeat our previous calculations for example 1 in the relational setting of $\mathbf{FRel} \times P$.

Example 1 revisited. Consider the singleton set $\{*\}$; we assume that it signifies the vector space S. We assume that the two subsets of this set, namely $\{*\}$ and $\emptyset$, will respectively identify *true* and *false*. We now have sets V, W and $T = V \times \{*\} \times W$ with

$$V := \{m_i\}_i\,, \quad \textit{likes} \subset T, \quad W := \{f_j\}_j$$

such that:

$$\textit{likes} := \{(m_i, *, f_j) \mid m_i \textit{ likes } f_j\} = \bigcup_{ij} \{m_i\} \times *_{ij} \times \{f_j\}$$

where $*_{ij}$ is either $\{*\}$ or $\emptyset$. So we obtain

$$f\left(\{m_3\} \times \textit{likes} \times \{f_4\}\right) = \bigcup_{ij} (\{m_3\} \cap \{m_i\}) \times *_{ij} \times \left(\{f_j\} \cap \{f_4\}\right) = *_{34}\,.$$

7 Future Work

This paper aims to lay a mathematical foundation for the new field of compositional distributional models of meaning in the realm of computational and mathematical

linguistics, with applications to language processing, information retrieval, artificial intelligence, and in a conceptual way to the philosophy of language. This is just the beginning and there is so much more to do, both on the practical and the theoretical sides. Here are some examples:

- On the logical side, our "not" matrix works by swapping basis and is thus essentially two dimensional. Developing a canonical matrix of negation, one that works uniformly for any dimension of the meaning spaces, constitutes future work. The proposal of [41] in using projection to the orthogonal subspace might be an option.

- A similar problem arises for the meanings of other logical words, such as "and", "or", "if then". So we need to develop a general logical setting on top of our meaning category $\mathbf{FVect} \times \mathbf{P}$. One subtlety here is that the operation that first come to mind, i.e. vector sum and product, do not correspond to logical connective of disjunction and conjunction (since e.g. they are not fully distributive). However, the more relaxed setting of vector spaces enables us to also encode words such as "but", whose meaning depends on the context and thus do not have a unique logical counterpart.

- Formalizing the connection with Montague-semantics is another future direction. Our above ideas can be generalized by proving a representation theorem for $\mathbf{Fvect} \times \mathbf{P}$ on the semiring of Booleans with respect to the category of $\mathbf{FRel}$ of sets and relations. It would then be interesting to see how the so called 'non-logical' axioms of Montague are manifested at that level, e.g. as adjoints to substitution to recover quantifiers.

- Along similar semantic lines, it would be good to have a Curry-Howard-like isomorphism between non-commutative compact closed categories, bicompact linear logic [4], a version of lambda calculus. This will enable us to automatically obtain computations for the meaning and type assignments of our categorical setting.

- Our categorical axiomatics is flexible enough to accommodate *mixed states* [37], so in principle we are able to study their linguistic significance, and for instance implement the proposals of [3].

- Finally, and perhaps most importantly, the mathematical setting needs to be implemented and evaluated, by running experiments on real corpus data. Efficiency and the complexity of our approach then become an issue and need to be investigated, along with optimization techniques.

Acknowledgements

Support from EPSRC Advanced Research Fellowship EP/D072786/1 and European Committee grant EC-FP6-STREP 033763 for Bob Coecke, EPSRC Postdoctoral Fellowship EP/F042728/1 for Mehrnoosh Sadrzadeh, and EPSRC grant EP/E035698/1 for Stephen Clark are gratefully acknowledged. We thank Keith Van Rijsbergen, Stephen Pulman, and Edward Grefenstette for discussions, and Mirella Lapata for providing relevant references for vector space models of meaning.

References

[1] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425. IEEE Computer Science Press, 2004. arXiv:quant-ph/0402130.

[2] Jerome R. Bellegarda. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296, 2000.

[3] P. Bruza and D. Widdows. Quantum information dynamics and open world science. In *Proceedings of AAAI Spring Symposium on Quantum Interaction*. AAAI Press, 2007.

[4] W. Buszkowski. Lambek grammars based on pregroups. *Logical Aspects of Computational Linguistics*, 2001.

[5] Freddy Choi, Peter Wiemer-Hastings, and Johanna Moore. Latent Semantic Analysis for text segmentation. In *Proceedings of the EMNLP Conference*, pages 109–117, 2001.

[6] S. Clark and S. Pulman. Combining symbolic and distributional models of meaning. In *Proceedings of AAAI Spring Symposium on Quantum Interaction*. AAAI Press, 2007.

[7] B. Coecke. Kindergarten quantum mechanics — lecture notes. In A. Khrennikov, editor, *Quantum Theory: Reconsiderations of the Foundations III*, pages 81–98. AIP Press, 2005. arXiv:quant-ph/0510032.

[8] B. Coecke. Introducing categories to the practicing physicist. In G. Sica, editor, *What is category theory?*, volume 30 of *Advanced Studies in Mathematics and Logic*, pages 45–74. Polimetrica Publishing, 2006. arXiv:0808.1032.

[9] B. Coecke. Quantum picturalism. *Contemporary physics*, 51:59–83, 2010. arXiv:0908.1787.

[10] B. Coecke and E. O. Paquette. Categories for the practicing physicist. In B. Coecke, editor, *New structures for physics*, Lecture Notes in Physics, pages 167–271. Springer, 2010. arXiv:0905.3010.

[11] James R. Curran. *From Distributional to Semantic Similarity*. PhD thesis, University of Edinburgh, 2004.

[12] G. Denhire and B. Lemaire. A computational model of children's semantic memory. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, pages 297–302, Chicago, IL, 2004.

[13] D.R. Dowty, R.E. Wall, and S. Peters. *Introduction to Montague Semantics*. Dordrecht, 1981.

[14] Peter W. Foltz, Walter Kintsch, and Thomas K. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse Process*, 15:285–307, 1998.

[15] G. Gazdar. Paradigm merger in natural language processing. In R. Milner and I. Wand, editors, *Computing Tomorrow: Future Research Directions in Computer Science*, pages 88–109. Cambridge University Press, 1996.

[16] Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, 1994.

[17] Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. Topics in semantic representation. *Psychological Review*, 114(2):211–244, 2007.

[18] J. Lambek. The mathematics of sentence structure. *American Mathematics Monthly*, 65, 1958.

[19] J. Lambek. Type grammar revisited. *Logical Aspects of Computational Linguistics*, 1582, 1999.

[20] J. Lambek. Iterated galois connections in arithmetics and linguistics. *Galois Connections and Applications, Mathematics and its Applications*, 565, 2004.

[21] J. Lambek. *From Word to Sentence*. Polimetrica, 2008.

[22] J. Lambek. Compact monoidal categories from linguistics to physics. In B. Coecke, editor, *New structures for physics*, Lecture Notes in Physics, pages 451–469. Springer, 2010.

[23] J. Lambek and C. Casadio, editors. *Computational algebraic approaches to natural language*. Polimetrica, Milan, 2006.

[24] T. K. Landauer and S. T. Dumais. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

[25] Michael D. Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In B.G. Bara, L.W. Barsalou, and M. Bucciarelli, editors, *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 1254–1259, Mahwah, NJ, 2005. Erlbaum.

[26] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 768–774, 1998.

[27] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments & Computers*, 28:203–208, 1996.

[28] Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 280–287, Barcelona, Spain, 2004.

[29] Scott McDonald. *Environmental Determinants of Lexical Processing Effort.* PhD thesis, University of Edinburgh, 2000.

[30] A. Preller. Towards discourse representation via pregroup grammars. *JoLLI*, 2007.

[31] A. Preller and M. Sadrzadeh. Bell states and negative sentences in the distributed model of meaning. In P. Selinger B. Coecke, P. Panangaden, editor, *Electronic Notes in Theoretical Computer Science, Proceedings of the 6th QPL Workshop on Quantum Physics and Logic*. University of Oxford, 2010.

[32] M. Sadrzadeh. Pregroup analysis of Persian sentences. In C. Casadio and J. Lambek, editors, *Computational algebraic approaches to natural language*. Polimetrica, 2006.

[33] M. Sadrzadeh. High level quantum structures in linguistics and multi agent systems. In J. van Rijsbergen P. Bruza, W. Lawless, editor, *Proceedings of the AAAI Spring Symposium on Quantum Interaction*. Stanford University, 2007.

[34] J. R. Saffran, E. L. Newport, and R. N. Asling. Word segmentation: The role of distributional cues. *Journal of Memory and Language*, 35:606–621, 1999.

[35] G Salton, A Wang, and C Yang. A vector-space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620, 1975.

[36] H. Schuetze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[37] P. Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical Computer Science*, 170:139–163, 2007.

[38] P. Selinger. A survey of graphical languages for monoidal categories. In B. Coecke, editor, *New structures for physics*, Lecture Notes in Physics, pages 275–337. Springer, 2010.

[39] P. Smolensky and G. Legendre. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar Vol. I: Cognitive Architecture Vol. II: Linguistic and Philosophical Implications*. MIT Press, 2005.

[40] D. P. Spence and K. C. Owens. Lexical co-occurrence and association strength. *Journal of Psycholinguistic Research*, (19):317–330, 1990.

[41] D. Widdows. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *41st Annual Meeting of the Association for Computational Linguistics*, Japan, 2003.