Annals of Pure and Applied Logic 164 (2013) 1079–1100

Contents lists available at SciVerse ScienceDirect

Annals of Pure and Applied Logic

www.elsevier.com/locate/apal

Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus

Bob Coecke [a], Edward Grefenstette [a], Mehrnoosh Sadrzadeh [b,*]

[a] Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom
[b] School of Electronics and Computer Science, Queen Mary University of London, Mile End Road, London, E1 4NS, United Kingdom

ARTICLE INFO

Article history:
Received 12 December 2011
Accepted 13 March 2013
Available online 17 June 2013

MSC:
03
18
68
81

Keywords:
Categorial grammars
Vector spaces
Monoidal categories
Lambek calculus
Pregroups
Distributional semantics

ABSTRACT

The Distributional Compositional Categorical (DisCoCat) model is a mathematical framework that provides compositional semantics for meanings of natural language sentences. It consists of a computational procedure for constructing meanings of sentences, given their grammatical structure in terms of compositional type-logic, and given the empirically derived meanings of their words. For the particular case that the meaning of words is modelled within a distributional vector space model, its experimental predictions, derived from real large scale data, have outperformed other empirically validated methods that could build vectors for a full sentence. This success can be attributed to a conceptually motivated mathematical underpinning, something which the other methods lack, by integrating *qualitative* compositional type-logic and *quantitative* modelling of meaning within a category-theoretic mathematical framework. The type-logic used in the DisCoCat model is Lambek's pregroup grammar. Pregroup types form a posetal compact closed category, which can be passed, in a functorial manner, on to the compact closed structure of vector spaces, linear maps and tensor product. The diagrammatic versions of the equational reasoning in compact closed categories can be interpreted as the *flow of word meanings* within sentences. Pregroups simplify Lambek's previous type-logic, the Lambek calculus. The latter and its extensions have been extensively used to formalise and reason about various linguistic phenomena. Hence, the apparent reliance of the DisCoCat on pregroups has been seen as a shortcoming. This paper addresses this concern, by pointing out that one may as well realise a functorial passage from the original type-logic of Lambek, a monoidal bi-closed category, to vector spaces, or to any other model of meaning organised within a monoidal bi-closed category. The corresponding string diagram calculus, due to Baez and Stay, now depicts the flow of word meanings, and also reflects the structure of the parse trees of the Lambek calculus.

1. Introduction

Language is both empirical and compositional: we learn meanings of words by being exposed to linguistic practice, and we form sentences we've never heard before by composing words according to the rules of grammar. Various mathematical and formal models have sought to capture facets and aspects of language learning and formation. Compositional type-logical approaches [30] represent sentence formation rules based on formal syntactic analysis, using formalisms such as context free grammars [10,41], Lambek calculus [30], or Combinatorial Categorial Grammar [56]. Such formal approaches to grammar

* Corresponding author.
E-mail addresses: bob.coecke@cs.ox.ac.uk (B. Coecke), edward.grefenstette@cs.ox.ac.uk (E. Grefenstette), mehrnoosh.sadrzadeh@eecs.qmul.ac.uk (M. Sadrzadeh).

0168-0072/$ – see front matter
http://dx.doi.org/10.1016/j.apal.2013.05.009

align well with Frege's notion of compositionality, according to which the meaning of a sentence is a function of the meaning of its parts [19], but eschew the empirical nature of language, requiring pre-defined mathematical structures, domains and valuations to make sense.

Orthogonal to formal logical models, empirical approaches to semantics construct representations of individual words based on the contexts in which they are used. These models are often referred to as *distributional* or geometric models of semantics and are sometimes considered to be in line with the "meaning is use" view of Wittgenstein's philosophy of language [59]. Distributional models have been applied successfully to tasks such as thesaurus extraction [20,15], automated essay marking [36], and other semantically motivated natural language processing tasks. While these models reflect the empirical aspects of language learning that type-logical models lack, they in turn lack composition operations which would allow us to learn meanings of phrases based on the meanings of their parts. Developing models that could combine the strengths of the above two approaches has proved to be a challenge for computational linguistics and its applications to natural language processing (NLP).

The distributional compositional categorical (DisCoCat) model of meaning, developed in [13,14], provides a solution to the above problem. This framework, which realised the challenge proposed in [11] enables a combination of the type-logical and distributional models of meaning and resulted in a procedure for compositionally computing meaning vectors for sentences by exploiting the grammatical structure of sentences and the meaning vectors of the words therein. The framework was inspired by the category-theoretic high-level framework for modelling quantum protocols [2], were the corresponding string diagram calculus exposes flows of information between the systems involved in multi-system protocols such as quantum teleportation [12]. The DisCoCat model has meanwhile been experimentally validated for natural language tasks such as word-sense disambiguation within phrases [21,22].

The DisCoCat model relies on Lambek pregroups [32] as its base type-logic. In category-theoretic terms, these have a (non-symmetric) compact closed structure when considering types as objects and type reductions as morphisms. The DisCoCat exploits the fact that finite dimensional vector spaces can also be organised within a compact closed category. The first and mainly technical goal of this paper is to stress that the choice of a compact or monoidal type-logic is not crucial to the applicability of the procedure. To achieve this goal, we have tweaked the distributional compositional model of previous work from Lambek pregroups to Lambek monoids, hence developing a vector space model for the meaning of natural language sentences parsed within the Lambek calculus. In this paper, we develop a similar homographic passage via a functor from a monoidal bi-closed category of grammatical types and reductions to the symmetric monoidal closed category of finite dimensional vector spaces.

This functorial passage is another contribution of this paper and gives rise to an interesting analogy with Topological Quantum Field Theory (TQFT) [4,5,28]. A TQFT is also a monoidal functor from the category of cobordisms into the category of vector spaces and linear maps. From the perspective of TQFT, our DisCoCat models form a 'Grammatical Quantum Field Theory' obtained by replacing the monoidal category of cobordisms in a TQFT by a certain partially ordered monoid which accounts for grammatical structure. This analogy of the compositional distributional model of meaning with TQFT was first pointed out by Louis Crane at a workshop in Oxford, August 2008. Similar to the original model-theoretic framework of meaning by Montague [41], this semantic framework is obtained via a homomorphic passage from sentence formation rules to compositions of meanings of words. However, contrary to the Montague's model, meanings of words and sentences are expressed in terms of vectors and vector compositions rather than in terms of sets and set-theoretic operations.

As a result of the compactness of Lambek pregroups, the mechanism of how meanings of words interact to produce meanings of sentences has a purely diagrammatic form, which admits an intuitive interpretation in terms of *information flow*. By *information flow* we mean the topology of the two-dimensional graphical representation of the operations that produces the meaning of sentences from the meaning of words. Mathematically, these are expressed in the graphical language of the particular category in which we model the meaning of words and sentences [54], a practice tracing back to Penrose's work in the early 1970s [47], that was turned into a formal discipline by Joyal and Street in the 1990s [25]. These diagrams, for the particular case of compact closed categories, were extensively exploited in the earlier DisCoCat models. Here we show how the clasp-string calculus of Baez and Stay [6] can be used to provide diagrams for information flows that arise in the Lambek monoids, which are not compact. Our ambition is to use this work as a starting point for providing vector space meaning for more expressive natural language sentences such as those parsed with Combinatorial Categorial Grammars (CCGs) or Lambek–Grishin calculus [44,8]. The expressive powers of these grammars go beyond that of Lambek grammars, which are context free.

Finally, drawing a connection with games seems appropriate in the context of this special issue. Application of games to interpreting and formalising natural language traces back to the 'dialogical logic' of Lorenz and Lorenzen [39] who used the dialogue analogy to develop a game semantic model for formulae of intuitionistic logic. Later, a classical logic version of the theory with a model theoretic focus was developed by Hintikka and Sandu [23]. A proof-theoretic approach led to the use of linear logic, and proof nets, e.g. see [29,37]. Independently, another line of research was pursued by linguists who also used the term 'dialogue games' to provide a semantic model for real-life human–computer dialogues. One of the original proposals of this line was based on Grice's pragmatic philosophy of language and used component programs and specifications to model dialogues and queries; the setting was applied to online sale tools [38]. Later on, a formal model based on belief revision and Bayesian update was developed for this approach [52]. Our work can be seen as bridging these two (abstract logical and applied linguistic) communities. Our starting point is a Lambek calculus, with a proof theory similar to that of intuitionistic multiplicative linear logic. In this calculus, the grammatical structure of a sentence is represented as

a derivation in a proof tree and depicted in diagrams similar to proof nets. We interpret these derivations in vector spaces, seen as a monoidal closed category, and depict the grammatical and semantic interactions via Baez–Stay diagrams. General linear logic proof nets are quite different from Baez–Stay diagrams, but their compact variants introduced in [3,1] resemble the compact closed string diagrams used in the pregroup derivations.

2. Partial order structures in linguistics

Application of partially ordered algebras to linguistics originated in the seminal work of Lambek in the 50's [30]. In his debut work, Lambek showed how a partially ordered residuated monoid can be used to analyse the syntactic structure of a fragment of English. He later developed a decision procedure for this setting, based on a cut-free sequent calculus. This calculus can be seen as the father of linear logic, as it shares with it a monoidal tensor, in a non-commutative form, and hence two linear implications. Lambek's approach was based on a partial order of grammatical types; similar ideas have been present in the work of Bar-Hillel [7] but were not formalised in algebraic and proof theoretic forms. About half a century later in the late 90s, Lambek simplified his original residuated monoids in favour of a simpler partial order, which he called a pregroup [34]. Pregroups have been applied to analyse various different languages, for references see [35]. In this section we review these two structures and their application to natural language.

2.1. Lambek monoids

Lambek calculus [30] is usually a reference to Lambek's sequent calculus; this calculus is similar to that of intuitionistic multiplicative bi-linear logic, but lacks negation. It has one main binary operation, which is non-commutative, hence has a right and a left implication. This calculus is sound and complete with regard to partially ordered residuated monoids.

Recall that for two order-preserving maps $f : A \to B$ and $g : B \to A$ on two partially ordered sets A and B, we say that f is the *left adjoint* to, or the left *residual* of, g (or equivalently, g is the *right adjoint* to f or its left *residual*), denoted $f \dashv g$, iff

$$\forall a \in A, b \in B, \quad f(a) \leqslant b \Leftrightarrow a \leqslant g(b)$$

The above condition is equivalent to the following:

$$\forall b \in B, \quad f\big(g(b)\big) \leqslant b, \quad \text{and} \quad \forall a \in A, \quad a \leqslant g\big(f(a)\big)$$

Based on the above definition, a residuated monoid is defined as follows:

Definition 2.1. A residuated monoid $(L, \leqslant, \cdot, 1, \multimap, \circ\!\!-)$ is a partially ordered set $(L, \leqslant)$, equipped with a monoid structure $(L, \cdot, 1)$ that preserves the partial order, that is for all $a, b, c \in L$, we have:

$$\text{If} \quad a \leqslant b \quad \text{then} \quad a \cdot c \leqslant b \cdot c \quad \text{and} \quad c \cdot a \leqslant c \cdot b$$

The unit element 1 satisfies the following for all $a \in L$:

$$1 \cdot a = a \cdot 1 = a$$

That the monoid is residuated means that $\multimap$ and $\circ\!\!-$ are the two adjoints of $\cdot$, that is, $a \cdot (-) \dashv a \multimap (-)$ and $(-) \cdot b \dashv (-) \circ\!\!- b$, explicitly we have:

$$b \leqslant a \multimap c \Leftrightarrow a \cdot b \leqslant c \Leftrightarrow a \leqslant c \circ\!\!- b \tag{1}$$

or, equivalently, using the corollaries of these adjunctions, we have:

$$a \cdot (a \multimap c) \leqslant c, \quad c \leqslant a \multimap (a \cdot c), \qquad (c \circ\!\!- b) \cdot b \leqslant c, \quad c \leqslant (c \cdot b) \circ\!\!- b \tag{2}$$

These structures are also referred to as *residuated lattices* in the literature. But strictly speaking, a residuated monoid is a residuated lattice with the exclusion of its lattice operations, hence the main operation of this structure is a monoid multiplication.

We refer to a partially ordered residuated monoid as a *Lambek monoid*. These monoids are applied to the encoding of the grammatical structure of natural language, whereby elements of the algebra denote grammatical types, the monoid multiplication is the juxtaposition of these types, and its unit is the empty type. The right and left adjoints are used to denote function-types; these encode the types of the words that have a relational role, for example adjectives, verbs, adverbs, conjunctives, and relative pronouns. This application procedure is formalised via the following structures.

Table 1
Type assignments for the toy language Σ in a Lambek monoid.

men	dogs	cute	kill	to kill	do	not
n	n	$n \circ\!\!- n$	$(n \multimap s) \circ\!\!- n$	$(\sigma \multimap j) \circ\!\!- n$	$(n \multimap s) \circ\!\!- (\sigma \multimap j)$	$(\sigma \multimap j) \circ\!\!- (\sigma \multimap j)$

Definition 2.2. For Σ the set of words of a natural language and $\mathcal{B}$ a set of basic grammatical types, a Lambek type-dictionary is a binary relation D, defined as

$$D \subseteq \Sigma \times T(\mathcal{B})$$

where $T(\mathcal{B})$ is the free Lambek monoid generated over $\mathcal{B}$ (for the free construction see [30]).

Definition 2.3. A Lambek grammar G is a pair $\langle D, S\rangle$, where D is a Lambek type-dictionary and $S \subset \mathcal{B}$ is a set of designated types, containing types such as that of a declarative sentence s, and a question q.

A Lambek grammar is used to define the grammatical sentences of a language as follows.

Definition 2.4. A string of words $w_1 w_2 \cdots w_n$ each of them from Σ is said to be grammatical iff for $1 \leqslant i \leqslant n$, there exists a (w_i, t_i) in the dictionary D, such that for the designated type s of a sentence in S, the following partial order holds in $T(\mathcal{B})$:

$$t_1 \cdot t_2 \cdot \cdots \cdot t_n \leqslant s$$

As an example, consider a simple language that contains the following words:

$$\Sigma = \{\text{men}, \text{dogs}, \text{cute}, \text{kill}, \text{do}, \text{not}\}$$

For the sake of this example suppose that in this language one can only form declarative sentences 'men kill dogs', 'men do not kill dogs', and 'men kill cute dogs'. A Lambek type-dictionary that generates the grammatical sentences of this language has the following basic types:

$$\mathcal{B} = \{n, s, j, \sigma\}$$

Here, n stands for a noun phrase, s for a declarative sentence, j for the infinitive of a verb, and σ is a marker type. The type assignments to words of Σ are presented in Table 1.

The Lambek type-dictionary D corresponding to the type assignments of Table 1 is as follows:

$$\begin{aligned}&\{(\text{men}, n), (\text{dogs}, n), (\text{cute}, n \circ\!\!- n), \big(\text{kill}, (n \multimap s) \circ\!\!- n\big), \big(\text{to kill}, (\sigma \multimap j) \circ\!\!- n\big),\\ &\quad \big(\text{do}, (n \multimap s) \circ\!\!- (\sigma \multimap j)\big), \big(\text{not}, (\sigma \multimap j) \circ\!\!- (\sigma \multimap j)\big)\}\end{aligned}$$

Note that the verb 'kill' has two types, represented by two pairs in the type dictionary: the first one $(\text{kill}, (n \multimap s) \circ\!\!- n)$ is for its transitive role, e.g. in the sentence 'men kill dogs' and the second one $(\text{kill}, (\sigma \multimap j) \circ\!\!- n)$ for its infinitive role, e.g. in the sentence 'men do not kill dogs'.

By Definition 2.4, the sentence 'men kill dogs' is grammatical, since if we apply the monoid multiplication to the types that correspond to the words, we obtain the term $n \cdot (n \multimap s) \circ\!\!- n \cdot n$, which has the following reduction:

$$\begin{aligned}n \cdot \big((n \multimap s) \circ\!\!- n\big) \cdot n &\leqslant\\ n \cdot (n \multimap s) &\leqslant s\end{aligned}$$

The sentence 'men kill cute dogs' is also grammatical; it has the following reduction:

$$\begin{aligned}n \cdot \big((n \multimap s) \circ\!\!- n\big) \cdot (n \circ\!\!- n) \cdot n &\leqslant\\ n \cdot \big((n \multimap s) \circ\!\!- n\big) \cdot n &\leqslant\\ n \cdot (n \multimap s) &\leqslant s\end{aligned}$$

Similarly, the sentence 'men do not kill dogs' is grammatical, according to the following reduction:

$$\begin{aligned}n \cdot \big((n \multimap s) \circ\!\!- (\sigma \multimap j)\big) \cdot \big((\sigma \multimap j) \circ\!\!- (\sigma \multimap j)\big) \cdot \big((\sigma \multimap j) \circ\!\!- n\big) \cdot n &\leqslant\\ n \cdot \big((n \multimap s) \circ\!\!- (\sigma \multimap j)\big) \cdot \big((\sigma \multimap j) \circ\!\!- (\sigma \multimap j)\big) \cdot (\sigma \multimap j) &\leqslant\\ n \cdot \big((n \multimap s) \circ\!\!- (\sigma \multimap j)\big) \cdot (\sigma \multimap j) &\leqslant\\ n \cdot (n \multimap s) &\leqslant s\end{aligned}$$

Table 2
Type assignments for the toy language Σ in a Lambek pregroup.

men	dogs	cute	kill	to kill	do	not
n	n	$n \cdot n^l$	$n^r \cdot s \cdot n^l$	$\sigma^r \cdot j \cdot n^l$	$n^r \cdot s \cdot j^l \cdot \sigma$	$\sigma^r \cdot j \cdot j^l \cdot \sigma$

2.1.1. Lambek pregroups

In 1999, Lambek revisited his monoidal structures and introduced a simplification [32]. Instead of working in a partially ordered residuated monoid, he argued for a pregroup, which has one non-residuated binary operation, but where each element of the partial order is required to have a left and a right adjoint. More precisely, we have:

Definition 2.5. A Lambek pregroup is a partially ordered unital monoid where each element has a left and a right adjoint $(P, \leqslant, \cdot, 1, (-)^l, (-)^r)$. That is, for every $p \in P$, there is a p^r and a p^l in P, which satisfy the following four inequalities:

$$p \cdot p^r \leqslant 1 \leqslant p^r \cdot p$$
$$p^l \cdot p \leqslant 1 \leqslant p \cdot p^l$$

From this definition it follows that adjoints are unique and reverse the order, that is if we have $p \leqslant q$ for $p, q \in P$ then it follows that $q^l \leqslant p^l$ and also that $q^r \leqslant p^r$. One can also show that the unit is self-adjoint, that is $1^r = 1 = 1^l$, that opposite adjoints cancel out, that is $(p^r)^l = (p^l)^r = p$, but same adjoints iterate, for instance $(p^r)^r$ is not necessarily equal to p and neither is $(p^l)^l$. Another nice property is that the monoid multiplication is self-adjoint, that is $(p \cdot q)^r = q^r \cdot p^r$ and also $(p \cdot q)^l = q^l \cdot p^l$. These properties are all proved and elaborated on by Lambek [32].

Apart from linguistics, pregroups have concrete applications in other fields such as to number theory [33]. Notably, an example of a pregroup structure on natural numbers is the set of all unbounded monotone functions on the set of integers $\mathbb{Z}$. Here, the partial order is the natural ordering of integers extended to functions, that is for $f, g \in \mathbb{Z}^{\mathbb{Z}}$, we have:

$$f \leqslant g \quad \text{iff} \quad f(n) \leqslant g(n), \quad \forall n \in \mathbb{Z}$$

The monoid multiplication is the composition of functions and its unit is the identity function, that is for $n \in \mathbb{Z}$ we have:

$$(f \cdot g)(n) = f\big(g(n)\big) \quad \text{and} \quad 1(n) = n$$

The left and right adjoints of each function are computed by using their canonical definitions and via the supremum and infimum operations on integers, again extended to functions, as follows:

$$f^r(n) = \vee\big\{m \in \mathbb{Z} \,\big|\, f(m) \leqslant n\big\} \qquad f^l(n) = \wedge\big\{m \in \mathbb{Z} \,\big|\, n \leqslant f(m)\big\}$$

As an example, take $f(x) = 2x$, then compute $f^r(x) = \lfloor x/2 \rfloor$ and $f^l(x) = \lfloor (x+1)/2 \rfloor$, where $\lfloor x \rfloor$ is the biggest integer less than or equal to x (for details of these computations and more examples see [33]).

Pregroups are applied to the analysis of syntax in the same way as Lambek monoids. The types are translated from their monoidal implicative form to a pregroup adjoint form as follows:

$$p \multimap q \rightsquigarrow p^r \cdot q \qquad p \circ\!\!- q \rightsquigarrow p \cdot q^l$$

The pregroup version of Table 1 is shown in Table 2.

The reduction corresponding to the sentence 'men kill dogs' is as follows:

$$n \cdot n^r \cdot s \cdot n^l \leqslant 1 \cdot s \cdot 1 = s$$

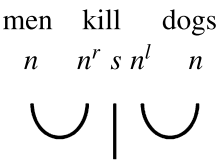
For 'men kill cute dogs', we have:

$$n \cdot n^r \cdot s \cdot n^l \cdot n \cdot n^l \cdot n \leqslant 1 \cdot s \cdot 1 \cdot 1 = s$$

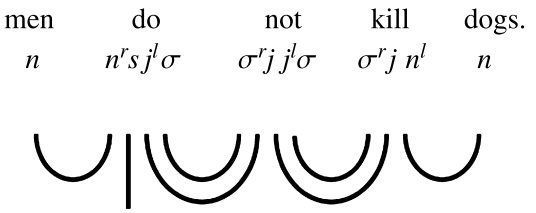
And for 'men do not kill dogs' we have:

$$\begin{aligned} n \cdot n^r \cdot s \cdot j^l \cdot \sigma \cdot \sigma^r \cdot j \cdot j^l \cdot \sigma \cdot \sigma^r \cdot j \cdot n^l \cdot n \leqslant \\ 1 \cdot s \cdot j^l \cdot 1 \cdot j \cdot j^l \cdot 1 \cdot j \cdot 1 = \\ s \cdot j^l \cdot j \cdot j^l \cdot j \leqslant \\ s \cdot 1 \cdot 1 = s \end{aligned}$$

The pregroup reductions are usually depicted in *cancellation diagrams* using curved strings (cups), which connect the types that are being cancelled in each step, and lines, which depict types which have not been cancelled. Nested strings are used to denote multiple steps. For example, the cancellation diagram of 'men kill dogs' is as follows (the $\cdot$'s are dropped).



The diagram for the grammatical structure of 'men do not kill dogs' is as follows:



These diagrams provide an intuitive reading of the grammatical structure of the sentence. For instance, in the first diagram we read that for a transitive sentence to be grammatical, the verb has to interact with its subject and object, depicted via the curved strings, hence producing a sentence via the line.

The challenge of using type-logics to formalise natural language grammar resides in assigning the right basic and compound types to the words of the language such that they would generate all the grammatical statements of the language and do not over-generate. Different type dictionaries have been suggested for different languages, e.g. see [34,42]. The types used in this paper are from [49]; these are chosen to keep the setting simple and to be able to parse our example sentences. Parsing a more complex language needs a more elaborate type-dictionary.

Additional operators, either in terms of modalities or new additive binary connectives have been added to type-logics to increase their expressive power [45,58,43,27]. The expressive power of Lambek monoids and pregroups is the same as that of *context-free* grammars of the Chomsky hierarchy [48,9]. The extensions with modalities and additives increase it to weak variants of *context-sensitive* such as *mildly context-sensitive*. In the proof-theoretic calculi of these algebras, the grammatical reductions are depicted via proof nets [53,46], which offer a richer analysis of the decomposition of types.

The above type-logical structures do represent the grammatical structure of a language in a compositional way, but do not offer a model for the lexical semantics of the words of a language. In such type-logics, words are only modelled by their grammatical roles, hence words that have the same grammatical role but different lexical meanings cannot be distinguished from one another. For instance, the words 'dog' and 'men' are both represented only by their grammatical role n, ignoring the fact that they have different lexical meanings. The same problem exists for transitive verbs such as 'kill' and 'eat', adjectives such as 'cute' and 'green', and so on. For a subspace of a real vector space, see Fig. 1.

3. Distributional models of meaning

Orthogonal to the type-logical models, distributional models of meaning are mainly concerned with lexical semantics. Best described by a quotation by Firth that "You shall know a word by the company it keeps." [18], these models are based on the dictum that words that often appear in the same context have similar meanings. For example, the words 'cat' and 'dog' often appear in the context of 'pet', 'furry', 'owner', and 'food', hence they have similar meanings. Similarly, 'kill' and 'murder' often appear in the context of 'police', 'gun', and 'arrest', hence these also have similar meanings.

To formalise this idea, one builds a finite dimensional vector space N whose basis vectors are the context words. Ideally, the context words are all the lemmatised words of a corpus of interest. Typically, these are reduced to a more refined set, based on the domain of application. One then fixes a window of k words and builds a vector for each word, representing its lexical meaning. We denote the meaning vector of a word by $\overrightarrow{\text{word}} = \sum_i c_i \vec{n}_i$, for c_i a real number and $\vec{n}_i$ a basis vector of N. The c_i weights are obtained by first counting how many times a word has appeared within k (e.g. 5) words of each context and then normalising this count. The most popular normalisation measure is Term-Frequency (TF) divided by Inverse-Document-Frequency (IDF); it assigns a degree of importance to the appearance of a word in a document. TF/IDF is a proportion for the number of times a word appears in the corpus to the frequency of the total number of words in the corpus.

The distance between the meaning vectors, for instance the cosine of their angle, provides a good measure of similarity of meaning. For example, in the vector space of Fig. 1, the angle between meaning vectors of 'cat' and 'dog' is small and so is the angle between meaning vectors of 'kill' and 'murder'. Various similarity measures have been implemented on large scale data (up to a billion words) to build high dimensional vector spaces (tens of thousands of basis vectors). These have been successfully applied to automatic generation of thesauri; for example see [15].

Contrary to the type-logical models, the empirical distributional models ignore the grammatical structure of language. For instance, these models do not offer a canonical way of building meaning vectors for sentences. A compositional solution to this problem should use vector composition operators to combine meaning vectors of words and obtain meaning vectors for sentences. The simple operations widely studied in the literature, for instance in [40], are addition $(+)$ and component-wise

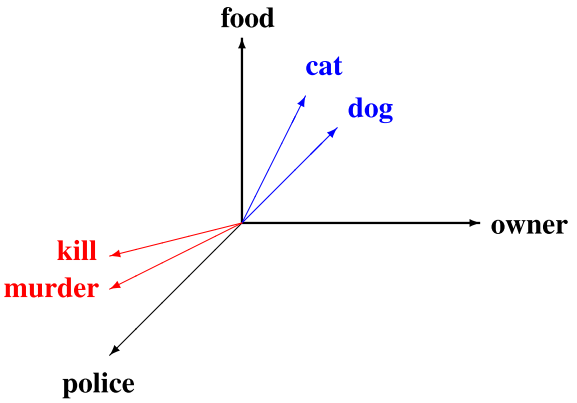


Fig. 1. A subspace of a real vector space model of meaning.

multiplication ($\odot$). These are commutative and hence do not even respect the word order. If $\overrightarrow{vw} = \vec{v} + \vec{w}$ or $\vec{v} \odot \vec{w}$, then $\overrightarrow{vw} = \overrightarrow{wv}$, leading to unwelcome equalities such as the following:

$$\overrightarrow{\text{men kill dogs}} = \overrightarrow{\text{dogs kill men}}$$

Inspired by the connectionist model of meaning in Cognitive Science [55], a combination of Kronecker product (which is non-commutative) and syntactic relations has been suggested in [11] as a possible solution. The problem with this model is that the dimensionalities of sentence vectors differ for sentences with different grammatical structures, barring them from being compared. For instance, in this model one cannot compare meanings of the two sentences 'men kill dogs' and 'men kill', since they live in different spaces.

4. Distributional compositional categorical model of meaning

In previous work [13,21], we combined the pregroup type-logic with the distributional model and developed a framework that produces vectors for meanings of sentences, from their grammatical structure and the vectors of the words therein. This framework, summarised below, is the Cartesian product of two compact closed categories: that of a pregroups with that of finite dimensional vector spaces.

4.1. Pregroups and vector spaces as compact closed categories

Lambek's type-logics closely relate to abstract categorical structures [31,50]. To see this connection, we recall some definitions from category theory. A compact closed category has objects A, B, morphisms $f : A \to B$, a monoidal tensor $A \otimes B$ that has a unit I, and for each object A two objects A^r and A^l together with the following morphisms:

$$A \otimes A^r \xrightarrow{\epsilon^r} I \xrightarrow{\eta^r} A^r \otimes A$$
$$A^l \otimes A \xrightarrow{\epsilon^l} I \xrightarrow{\eta^l} A \otimes A^l$$

The above satisfy the following equalities, where 1_A is the identity morphism on object A:

$$\left(1_A \otimes \epsilon^l\right) \circ \left(\eta^l \otimes 1_A\right) = 1_A \qquad \left(\epsilon^r \otimes 1_A\right) \circ \left(1_A \otimes \eta^r\right) = 1_A$$
$$\left(\epsilon^l \otimes 1_A\right) \circ \left(1_{A^l} \otimes \eta^l\right) = 1_{A^l} \qquad \left(1_{A^r} \otimes \epsilon^r\right) \circ \left(\eta^r \otimes 1_{A^r}\right) = 1_{A^r}$$

These inequalities are known as the *yanking* equalities. Note we do not assume symmetry of the tensor.

A pregroup is a compact closed category [50], to which we refer as *Preg*. The elements of the partially ordered set $p, q \in P$ are the objects of *Preg*, the partial order relation provides the morphisms, that is, there exists a morphism of type $p \to q$ iff $p \leqslant q$; we denote this morphism by $[p \leqslant q]$. Compositions of morphisms are given by transitivity and the identities by reflexivity of the partial order. The monoid multiplication and its unit (denoted by 1 rather than I) provide the tensor of the category, and the four adjoint inequalities provide the epsilon and eta morphisms, that is we have:

$$\epsilon^r = \left[p \cdot p^r \leqslant 1\right] \qquad \epsilon^l = \left[p^l \cdot p \leqslant 1\right]$$
$$\eta^r = \left[1 \leqslant p^r \cdot p\right] \qquad \eta^l = \left[1 \leqslant p \cdot p^l\right]$$

Finite dimensional vector spaces also form a compact closed category [2], we refer to it as *FVect*. Finite dimensional vector spaces V, W are objects of this category; linear maps $f : V \to W$ are its morphisms, and composition is the composition of

linear maps. The tensor product between the spaces $V \otimes W$ is the monoidal tensor whose unit is a field, in our case $\mathbb{R}$. As opposed to the tensor of the pregroup, this tensor is symmetric, hence we have a natural isomorphism $V \otimes W \cong W \otimes V$. As a result of symmetry of the tensor, the two adjoints collapse to one and we obtain $V^l \cong V^r \cong V^*$, where V^* is the dual of V. Since the basis vectors of our vector spaces are fixed, we furthermore obtain that $V^* \cong V$. Finally, a vector $\overrightarrow{v} \in V$ is represented by the morphism $\mathbb{R} \xrightarrow{\overrightarrow{v}} V$.

Given a basis $\{r_i\}_i$ for a vector spaces V, the epsilon maps are given by the inner product extended by linearity, that is we have:

$$\epsilon^l = \epsilon^r : \quad V^* \otimes V \to \mathbb{R}$$

$$:: \quad \sum_{ij} c_{ij} \psi_i \otimes \phi_j \mapsto \sum_{ij} c_{ij} \langle \psi_i \mid \phi_j \rangle$$

Eta maps are defined as follows:

$$\eta^l = \eta^r : \quad \mathbb{R} \to V \otimes V^*$$

$$:: \quad 1 \mapsto \sum_i r_i \otimes r_i$$

Here $1 \in \mathbb{R}$ is the number 1, and the above assignment extends to all other numbers by linearity.

A DiscoCat is the Cartesian product of *FVect* and *Preg*. This is a category: its objects are pairs (W, p) for W a finite dimensional vector space and p a pregroup type. Its morphisms are pairs of morphisms $(f : V \to W, [p \leqslant q])$, for f a linear map and $p \leqslant q$ a pregroup partial order. Composition is obtained in a pointwise fashion by composing linear maps and transitivity of the order. The identity morphism for an object (V, p) is a pair of identity morphisms $(1_V, [p \leqslant p])$ from *FVect* and *Preg*.

Definition 4.1. The distributional compositional categorical (DisCoCat) model of meaning is the category *FVect* × *Preg*, equipped with a tensor product given by pointwise tensor of *Preg* and *FVect*, that is $(V, p) \otimes (W, q) = (V \otimes W, p \cdot q)$, whose unit is $(\mathbb{R}, 1)$.

It has been shown in [13] that the compact structure carries over from *Preg* and *FVect* to a DisCoCat, with epsilon and eta maps given in a pointwise fashion as follows:

$$\epsilon^r = \left(V \otimes V^* \to \mathbb{R}, \left[p \cdot p^r \leqslant 1\right]\right) \qquad \epsilon^l = \left(V^* \otimes V \to \mathbb{R}, \left[p^l \cdot p \leqslant 1\right]\right)$$
$$\eta^r = \left(\mathbb{R} \to V^* \otimes V, \left[1 \leqslant p^r \cdot p\right]\right) \qquad \eta^l = \left(\mathbb{R} \to V \otimes V^*, \left[1 \leqslant p \cdot p^l\right]\right)$$

4.2. Meaning vectors for strings of words

The pair $(\overrightarrow{w} \in W, p)$ is an object of *Fvect* × *Preg*. We use this object to represent the *semantics* of a word w and refer to it as the *meaning* of the word w. It consists of a vector space W where the meaning vector $\overrightarrow{w}$ of word w that has pregroup type p lives. Based on this notion, the meaning vector representing the semantics of a sentence is obtained according to the following definition.

Definition 4.2. We define the meaning vector $\overrightarrow{w_1 \cdots w_n}$ 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 $(\overrightarrow{w_i} \in W_i, p_i)$ meaning of the word w_i, the linear map f is built by substituting each p_i in the pregroup reduction map of the string $[p_1 \cdot \cdots \cdot p_n \leqslant x]$ with W_i.

Thus for $\alpha = [p_1 \cdot \cdots \cdot p_n \leqslant x]$ a morphism in *Preg* and $f = \alpha[p_i / W_i]$ a linear map in *FVect*, the following is a morphism in *FVect* × *Preg*:

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

For example, to assign a meaning vector to an adjective-noun phrase, we start with the meanings of adjective and noun, which have the following forms:

$$\left(\overrightarrow{\text{adj}} \in W \otimes W, n \cdot n^l\right) \qquad (\overrightarrow{\text{noun}} \in W, n)$$

Here, we are assigning the vector space W to the noun $\overrightarrow{\text{noun}}$ and assuming that the distributional meaning vectors of the nouns live in it, that is $\overrightarrow{\text{noun}} \in W$. For the meaning vector of the adjective, it is to assumed to be an element of $W \otimes W$,

hence representable by $\sum_{lm} c_{lm} \vec{w}_l \otimes \vec{w}_m$, for $\vec{w}_l, \vec{w}_m$ basis vectors of W. The meaning vectors of the noun and adjective are represented by the following morphisms:

$$\mathbb{R} \xrightarrow{\overrightarrow{\text{noun}}} W \qquad \mathbb{R} \xrightarrow{\overrightarrow{\text{adj}}} W \otimes W$$

The pregroup reduction map of the adjective-noun phrase is as follows:

$$\alpha = \left[n \cdot n^l \cdot n \leqslant n\right] = 1_n \cdot \epsilon_n$$

Substituting each type in α with the vector space associated with the words of that type, we obtain the linear map f corresponding to α to be the morphism $(1_W \otimes \epsilon_W)$. The meaning vector of an adjective-noun phrase is computed by applying the linear map corresponding to the pregroup parse of the adjective-noun phrase, that is $(1_W \otimes \epsilon_W)$, to the tensor product of the meaning vector of the adjective $\overrightarrow{\text{adj}}$ with the meaning vector of the noun $\overrightarrow{\text{noun}}$. The categorical morphism corresponding to this computation is as follows:

$$\begin{aligned}
\overrightarrow{\text{adj noun}} &:= (1_W \otimes \epsilon_W)(\overrightarrow{\text{adj}} \otimes \overrightarrow{\text{noun}}) \\
&\cong (1_W \otimes \epsilon_W)\big((\mathbb{R} \xrightarrow{\overrightarrow{\text{adj}}} W \otimes W) \otimes (\mathbb{R} \xrightarrow{\overrightarrow{\text{noun}}} W)\big) \\
&\cong \mathbb{R} \otimes \mathbb{R} \xrightarrow{\overrightarrow{\text{adj}} \otimes \overrightarrow{\text{noun}}} W \otimes W \otimes W \xrightarrow{1_W \otimes \epsilon_W} W \otimes \mathbb{R} \\
&\cong \mathbb{R} \xrightarrow{\overrightarrow{\text{adj}} \otimes \overrightarrow{\text{noun}}} W \otimes W \otimes W \xrightarrow{1_W \otimes \epsilon_W} W
\end{aligned}$$

The concrete value of the vector corresponding to the above morphism is:

$$\overrightarrow{\text{adj noun}} := (1_W \otimes \epsilon_W)\left(\left(\sum_{lm} c_{lm}\ \vec{w}_l \otimes \vec{w}_m\right) \otimes \overrightarrow{\text{noun}}\right) = \sum_{lm} c_{lm} \vec{w}_l \langle \vec{w}_m \mid \overrightarrow{\text{noun}} \rangle$$

The meaning vector of a sentence with an adjective-noun phrase is computed by substituting the above for the meaning vector of the adjective-noun phrase when computing the meaning of the sentence. For instance, consider the sentence 'men kill dogs'; we have the following for the meaning vectors of words:

$$(\overrightarrow{\text{men}} \in W, n), \qquad (\overrightarrow{\text{dogs}} \in W, n), \qquad \left(\overrightarrow{\text{kill}} \in W \otimes S \otimes W, n^r \cdot s \cdot n^l\right)$$

So we are assigning the vector space W to nouns and the vector space S to sentences; as a result the vector space of transitive verbs and in particular of 'kill' will become $W \otimes S \otimes W$; each verb being representable by $\sum_{ijk} c_{ijk} \vec{w}_i \otimes \vec{s}_j \otimes \vec{w}_k$, for $\vec{w}_i, \vec{w}_k$ basis vectors of W and $\vec{s}_j$ a basis vector of S. The pregroup reduction map of the sentence is as follows:

$$\alpha = \left[n \cdot n^r \cdot s \cdot n^l \cdot n \leqslant s\right] = \epsilon_n \cdot 1_s \cdot \epsilon_n$$

Substituting each type in α with the vector space associated with the words of that type, we obtain f to be the morphism $\epsilon_W \otimes 1_S \otimes \epsilon_W$. Given the distributional meanings of each word, that is $\overrightarrow{\text{men}}, \overrightarrow{\text{kill}}, \overrightarrow{\text{dogs}}$, the meaning of the sentence will be:

$$\overrightarrow{\text{men kill dogs}} = (\epsilon_W \otimes 1_S \otimes \epsilon_W)(\overrightarrow{\text{men}} \otimes \overrightarrow{kill} \otimes \overrightarrow{dogs}) : W \otimes W \otimes S \otimes W \otimes W \xrightarrow{\epsilon_W \otimes 1_S \otimes \epsilon_W} S$$

further simplified as follows:

$$\begin{aligned}
\overrightarrow{\text{men kill dogs}} &= f(\overrightarrow{\text{men}} \otimes \overrightarrow{kill} \otimes \overrightarrow{dogs}) \\
&= (\epsilon_W \otimes 1_S \otimes \epsilon_W)(\overrightarrow{\text{men}} \otimes \overrightarrow{\text{kill}} \otimes \overrightarrow{\text{dogs}}) \\
&= (\epsilon_W \otimes 1_S \otimes \epsilon_W)\left(\overrightarrow{\text{men}} \otimes \left(\sum_{ijk} c_{ijk} \vec{w}_i \otimes \vec{s}_j \otimes \vec{w}_k\right) \otimes \overrightarrow{\text{dogs}}\right) \\
&= \sum_{ijk} c_{ijk} \langle \overrightarrow{\text{men}} \mid \vec{w}_i \rangle \langle \vec{w}_k \mid \overrightarrow{\text{dogs}} \rangle \vec{s}_j
\end{aligned}$$

Now, we can compute the meaning vector of the sentence 'men kill cute dogs': the pregroup morphism of the grammatical reduction of this sentence is $\epsilon_n \cdot 1_s \cdot \epsilon_n \cdot \epsilon_n$, whose linear map will be $\epsilon_W \otimes 1_S \otimes \epsilon_W \otimes \epsilon_W$. Applying this map to the tensor product of the meaning vectors of the words provides us with the following meaning vector for the sentence:

$$\begin{aligned}
\overrightarrow{\text{men kill cute dogs}} &= (\epsilon_W \otimes 1_S \otimes \epsilon_W \otimes \epsilon_W)(\overrightarrow{\text{men}} \otimes \overrightarrow{\text{kill}} \otimes \overrightarrow{\text{cute}} \otimes \overrightarrow{\text{dogs}}) \\
&= \sum_{ijk} c_{ijk} \langle \overrightarrow{men} \mid \vec{w}_i \rangle \left\langle \vec{w}_k \mid \overrightarrow{\text{cute}}(\overrightarrow{\text{dogs}}) \right\rangle \vec{s}_j \\
&= \sum_{ijk} \sum_{lm} c_{ijk} c_{lm} \langle \overrightarrow{men} \mid \vec{w}_i \rangle \langle \vec{w}_k \mid \vec{w}_l \rangle \langle \vec{w}_m \mid \overrightarrow{dogs} \rangle \vec{s}_j
\end{aligned}$$

As an example of a more complicated sentence, consider 'men do not kill dogs'. For the meaning vectors of the auxiliary 'do', the infinitive 'kill', and the negation word 'not', we have the following [13,51]:

$$\left(\overrightarrow{\text{kill}} \in W \otimes S \otimes W, \sigma^r \cdot j \cdot n^l\right), \qquad \left(\overrightarrow{\text{do}} \in W \otimes S \otimes S \otimes W, n^r \cdot s \cdot j^l \cdot \sigma\right), \qquad \left(\overrightarrow{\text{not}} \in W \otimes S \otimes S \otimes W, \sigma^r \cdot j \cdot j^l \cdot \sigma\right)$$

These are obtained by making the semantic assumption that the vector spaces for j and s are the same, that is S, and that the vector spaces for n and σ are the same, that is W. The first assumption is justified by the fact that transitive and infinitive transitive verbs both have the same meaning, hence live in the same vector space. The second assumption is justified by the fact that the σ's are place holders for the subject. The pregroup morphism corresponding to the grammatical reduction of the sentence is as follows:

$$\alpha = \left(1_s \cdot \epsilon_j^l \cdot \epsilon_j^l\right) \circ \left(\epsilon_n^r \cdot 1_s \cdot 1_{j^l} \cdot \epsilon_\sigma^r \cdot 1_j \cdot 1_{j^l} \cdot \epsilon_\sigma^r \cdot 1_j \cdot \epsilon_n^l\right)$$

The linear map f corresponding to the above is:

$$(1_S \otimes \epsilon_S \otimes \epsilon_S) \circ (\epsilon_W \otimes 1_S \otimes 1_S \otimes \epsilon_W \otimes 1_S \otimes 1_S \otimes \epsilon_W \otimes 1_J \otimes \epsilon_W)$$

The meaning of 'not' is generated from a linear map $\overline{\text{not}} : S \to S$, which takes a sentence and negates it. The meaning of 'do' is generated from the identity 1_S. These are as follows:

$$\overrightarrow{\text{do}} := \left(1_W \otimes \left((1_S \otimes 1_S) \circ \eta_S\right) \otimes 1_W\right) \circ \eta_W \qquad \overrightarrow{\text{not}} := \left(1_W \otimes \left((1_S \otimes \overline{\text{not}}) \circ \eta_S\right) \otimes 1_W\right) \circ \eta_W$$

The corresponding computations provide the following meaning vector for the sentence [13,51]:

$$\overline{\text{not}}(\overrightarrow{\text{men kill dogs}})$$

This is the *negation* of the meaning of the positive version of the sentence.

So far we have not said anything about the concrete shape of W and S. Moreover, for the general setting to work, we need concrete vectors for words with compound types whose vectors live in tensor spaces and cannot be directly obtained from distributional models. For example, the verb lives in $W \otimes S \otimes W$ and has to act on the subject and object, the adjective lives in $W \otimes W$ and has to modify the noun, the negation word $\overline{\text{not}}$ is a linear map that acts on the abstract space S and has to negate the meaning of a sentence and so on. Previous work presented some solutions for the limit truth-theoretic case, where S was the two-dimensional space of $|1\rangle$ and $|0\rangle$, and $\overline{\text{not}}$ was taken to be the linear map corresponding to the matrix of the swap gate [13]. Later, S was also taken to be a high dimensional vector spaces concretely built from the distributional models [21,22]. The next section summarises these results.

5. Concrete spaces and experimental results

In previous work, we provided an algorithm for building vectors for words with compound types and instantiated it for the particular case of intransitive and transitive sentences with simple subjects and objects [21,22]. Work in progress extends these cases to adjective-noun phrases in transitive sentences as subject and object. We evaluated the resulting vectors on a disambiguation task performed on real data obtained from the British National Corpus (BNC). This corpus consists of about 6 million sentences and 500,000 unique lemmas.

5.1. Concrete constructions

We build a vector space N from the BNC by taking its 2000 most occurring lemmas as basis vectors $\vec{n}_i$; this restriction is both for computational purposes and also to be able to compare our results to related work [40]. Vectors of N are built by counting co-occurrence and normalising by TF/IDF. We take W to be N and S to be $N \oplus (N \otimes N) \oplus (N \otimes N \otimes N)$. In the latter, the first N encodes meanings of intransitive sentences; these are unary relations; $N \otimes N$ encodes meanings of transitive sentences, which are binary relations, and $N \otimes N \otimes N$ is for meanings of ditransitive sentences, which are ternary relations. These relations are denoted by weighted sets of singletons, pairs, and triples, respectively. The set of singletons for an intransitive verb denotes the subjects to which the verb has been applied throughout the corpus. The set of pairs denotes the subject–object pairs that have been related by a transitive verb, etc. The weights represent *the extent according to which* the verb has acted on or related the nouns; these are learnt from the corpus in the following two ways. These are instantiations of a general procedure for building vectors for a word of any compound type [21].

1. **Categorical (1)**. The meaning vector of an intransitive verb $\overrightarrow{v} \in N$ is $\sum_i c_i \vec{n}_i$, where each c_i is obtained by summing the vectors of all the subjects of $\overrightarrow{v}$ throughout the BNC. The meaning vector of a transitive verb $\overrightarrow{tv} \in N \otimes N$ is $\sum_{ij} c_{ij} \vec{n}_i \otimes \vec{n}_j$, where each c_{ij} is the sum of the tensor products of all the subjects and objects that tv has related in the BNC, and similarly for ditransitive verbs.
2. **Categorical (2)**. Here we simply take the Kronecker products of the context vectors of the verbs, abusing the notation we still refer to it as $\overrightarrow{v}$. So for the intransitive verb, this is the vector obtained by a normalised count of co-occurrence $\overrightarrow{v} \in N$. For the transitive verb it is $\overrightarrow{v} \otimes \overrightarrow{v} \in N \otimes N$ and for the ditransitive it is $\overrightarrow{v} \otimes \overrightarrow{v} \otimes \overrightarrow{v} \in N \otimes N \otimes N$.

Table 3
Sample sentence pairs from the transitive dataset.

	Sentence 1	Sentence 2
1	the system met the criterion	the system visited the criterion
2	the system met the criterion	the system satisfied the criterion
3	the child met the house	the child visited the house
4	the child met the house	the child satisfied the house
5	the child showed interest	the child pictured interest
6	the child showed interest	the child expressed interest
7	the map showed the location	the map pictured the location
8	the map showed the location	the map expressed the location

Table 4
Results of the 1st and 2nd compositional disambiguation experiments.

Model	High	Low	ρ
Baseline	0.47	0.44	0.16
Add	0.90	0.90	0.05
Multiply	0.67	0.59	0.17
Categorical (1)	**0.73**	**0.72**	**0.21**
Categorical (2)	**0.34**	**0.26**	**0.28**
UpperBound	4.80	2.49	0.62

The above vectors are embedded in the spaces prescribed for them by a DisCoCat, that is $N \otimes S$ for $\overrightarrow{v}$ and $N \otimes S \otimes N$ for $\overrightarrow{tv}$, by diagonalisation, that is by padding the non-diagonal elements by zeros.

5.2. Evaluation task, dataset, and results

We evaluated these concrete methods on a disambiguation task. The general idea behind this task is that some verbs have more than one meaning and the sentences in which they appear disambiguate them. Suppose a verb v has two meanings a and b and we want to decide whether v means a or b whenever it occurs in a sentence s. To implement this task, we chose 10 ambiguous transitive verbs from the most frequent verbs of the BNC. For each verb, two different non-overlapping meanings were retrieved via the JCN measure of information content synonymy applied to WordNet synsets [24]. For instance one of our chosen verbs was 'meet', for which we obtained meaning a: 'visit' and meaning b: 'satisfy'. For each original verb, ten sentences containing that verb (in the same role) were retrieved from the BNC; for example, one such sentence for v: 'meet' is s: 'the system met the criterion'. For each such sentence, we generated two other sentences by substituting the verbs of those sentences by a and b, respectively. For instance, 'the system satisfied the criterion' and 'the system visited the criterion' were generated for the first meaning of 'meet'. This procedure provided us with 200 pairs of sentences. Note that the generated sentences only make sense for the correct meaning of the verb; for instance, 'the systems satisfied the criterion' does make sense, whereas 'the system visited the criterion' does not. The goal is to verify that the sentences 'the system met the criterion' and 'the system satisfied the criterion' have a high degree of semantic similarity, whereas the sentences 'the system met the criterion' and 'the system visited the criterion' have a low degree of similarity. The result of this verification disambiguates the verb. For the case of 'meet', we are verifying that it means 'satisfy' (and not 'visit') in the sentence 'the system met the criterion'.

We experimented with two datasets, one for intransitive sentences from [40] and an extension of it for transitive sentences, as described above (the extension to sentences with adjective–noun phrases and subject and object is straightforward and preserves the results). An example of the second dataset is provided in Table 3.

We built vectors for nouns by using the usual distributional co-occurrence method. Then built vectors for verbs using both of the methods described above, and finally built vectors for sentences using the DisCoCat prescription. After the sentence vectors were built, we measured the similarity of each pair using the cosine of their angles on the scale of the real numbers in $[0, 1]$. In order to judge the performance of our method, we followed guidelines from related work [40]. We distributed our data set among 25 volunteers who were asked to rank each pair based on how similar they thought they were. The ranking was between 1 and 7, where 1 was almost dissimilar and 7 almost identical. To be in line with related work [40], each pair was also given a HIGH or LOW classification by us. However, these scores are solely based on our personal judgements and on their own they do not provide a very reliable measure of comparison. The correlation of the model's similarity judgements with the human judgements was calculated using Spearman's ρ. This is a rank correlation coefficient ranging from -1 to 1 and provides a more robust metric (in contrast with the LOW/HIGH metric) by which models are ranked and compared.

The results of these calculations for our datasets are presented in Table 4. The additive and multiplicative rows have, as composition operation, vector addition and component-wise multiplication. The *Baseline* is from a non-compositional approach; it is obtained by comparing the verb vectors of each pair directly and ignoring their subjects and objects. The

UpperBound is set to be inter-annotator agreement. According to the ρ measure, both of our methods outperform the other methods.

6. A compositional distributional model of meaning on Lambek monoids

The contribution of this paper is to extend the DisCoCat model from pregroups to Lambek monoids. Rather than pairing *FVect* with a monoidal category, we will rely on a functorial passage from a monoidal category to *FVect*. There are two reasons to opt for a functorial passage. Firstly, it makes a closer connection to the original semantic models of natural language [41], which were based on a homomorphic passage from sentence formation rules to set theoretic operations. In our model, this homomorphic passage is formalised via a functor between categories of grammatical reductions and meaning. Contrary to those models [41], however, our model is not limited to sets and set-theoretic operations and is generalised to vectors and vector composition operations. This brings us to our second reason: a homomorphic passage to the category of vector spaces is not a one-off development especially tailored for our purposes. It is an example of a more general construction, namely, a passage long-known in Topological Quantum Field Theory (TQFT). This general passage was first developed in [4] in the context of TQFT and was given the name 'quantisation', as it adjoins 'quantum structure' (in terms of vectors) to a purely topological entity, namely the cobordisms representing the topology of manifolds. Later, this passage was generalised to abstract mathematical structures and recast in terms of functors whose co-domain was *FVect* [5,28]. This is exactly what is happening in our semantic framework: the sentence formation rules are formalised using type-logics and assigned quantitative values in terms of vector composition operations. This procedure makes our passage from grammatical structure to vector space meaning a 'quantisation' functor. Hence, one can say that what we are developing here is a grammatical quantum field theory for Lambek monoids. The detailed constructions of this paper can be worked out in a similar fashion for the pregroup-based framework of previous work [13,14].

6.1. Monoidal bi-closed categories

A Lambek monoid is a monoidal bi-closed category. Similar to the case of pregroups, its objects are the elements of the partial order and its morphisms are provided by the ordering relation. The monoid multiplication is a non-symmetric tensor, whose residuals are the left and right implications. To see this, we recall some definitions.

A monoidal bi-closed category is a category with a monoidal tensor $\otimes$ and its unit I, such that for all pairs of objects A, B of the category, there exist a pair of objects $A \multimap B$, $A \circ\!\!- B$ and a pair of morphisms as follows:

$$ev^l_{A,B} : A \otimes (A \multimap B) \to B \qquad ev^r_{A,B} : (A \circ\!\!- B) \otimes B \to A$$

These morphisms are referred to as *left* and *right evaluations*. They are such that for every pair of arrows $f : (A \otimes C) \to B$ and $g : (C \otimes B) \to A$ there exist two unique morphisms as follows:

$$\Lambda^l(f) : C \to A \multimap B \qquad \Lambda^r(g) : C \to A \circ\!\!- B$$

These morphisms are referred to as *left and right currying*; they make the following diagrams commute:

$$\begin{array}{ccc} A \otimes C & \xrightarrow{1_A \otimes \Lambda^l(f)} & A \otimes (A \multimap B) \\ & \searrow^{f} & \downarrow{\scriptstyle ev^r_{A,B}} \\ & & B \end{array} \qquad \begin{array}{ccc} C \otimes B & \xrightarrow{\Lambda^r(g) \otimes 1_B} & (A \circ\!\!- B) \otimes B \\ & \searrow^{g} & \downarrow{\scriptstyle ev^l_{A,B}} \\ & & A \end{array}$$

Given a morphism $f : A \to B$, its *name* $\ulcorner f \urcorner : I \xrightarrow{\Lambda^l(f)} A \multimap B$, is obtained by currying the morphism $(A \otimes I) \xrightarrow{f} B$ (dropping the precomposition with the ι_A morphism, i.e. $A \otimes I \xrightarrow{\iota_A} A$). In order to be coherent with the above left–right notation, we define a left and a right name for f, obtained by currying the morphisms $(A \otimes I) \xrightarrow{f} B$ and $(I \otimes A) \xrightarrow{f} B$, as follows:

$$\ulcorner f \urcorner^l : I \xrightarrow{\Lambda^l(f)} A \multimap B \qquad \ulcorner f \urcorner^r : I \xrightarrow{\Lambda^r(f)} B \circ\!\!- A$$

Evaluating these names makes the following diagrams commute:

$$\begin{array}{ccc} A \otimes I & \xrightarrow{1_A \otimes \ulcorner f \urcorner^l} & A \otimes (A \multimap B) \\ & \searrow^{f} & \downarrow{\scriptstyle ev^l_{A,B}} \\ & & B \end{array} \qquad \begin{array}{ccc} I \otimes A & \xrightarrow{\ulcorner f \urcorner^r \otimes 1_A} & (B \circ\!\!- A) \otimes A \\ & \searrow^{f} & \downarrow{\scriptstyle ev^r_{A,B}} \\ & & B \end{array}$$

In other words, we have the following two equations:

$$ev^l_{A,B} \circ \left(1_A \otimes \ulcorner f \urcorner^l\right) = f \qquad ev^r_{A,B} \circ \left(\ulcorner f \urcorner^r \otimes 1_A\right) = f$$

The names of the identity morphism $1_A : A \to A$ are obtained as special cases of the above construction. These are defined as follows:

$$\ulcorner 1_A \urcorner^l : I \xrightarrow{\Lambda^l(1_A)} A \multimap A \qquad \ulcorner 1_A \urcorner^r : I \xrightarrow{\Lambda^r(1_A)} A \circ\!\!- A$$

Evaluating the above leads to two similar commutative diagrams; the equations corresponding to these diagrams provide us with the monoidal version of *yanking*:

$$ev^l_{A,A} \circ \left(1_A \otimes \ulcorner 1_A \urcorner^l\right) = 1_A \qquad ev^r_{A,A} \circ \left(\ulcorner 1_A \urcorner^r \otimes 1_A\right) = 1_A$$

6.2. A quantisation functor for Lambek monoids

The quantisation functor preserves all the syntactic structure but "forgets" the order of the words. This order will be taken care of in our explicit definition of the meaning vector of a sentence in Definition 6.2.

Definition 6.1. Given a Lambek monoid $\mathcal{L}$ and the category of finite dimensional vector spaces *FVect* over $\mathbb{R}$, the quantisation functor $\mathcal{Q} : \mathcal{L} \to FVect$ is a strongly monoidal functor, satisfying the following:

$$\mathcal{Q}(1) := \mathbb{R} \tag{3}$$

$$\mathcal{Q}(a \cdot b) \cong \mathcal{Q}(b \cdot a) := \mathcal{Q}(a) \otimes \mathcal{Q}(b) \tag{4}$$

$$\mathcal{Q}(a \multimap b) \cong \mathcal{Q}(b \circ\!\!- a) := \mathcal{Q}(a) \Rightarrow \mathcal{Q}(b) \tag{5}$$

The tensor product in $FVect$ is symmetric, so (4) forgets the order and (5) collapses the two implications of $\mathcal{L}$ to just one. Also, *FVect* only has one evaluation and currying map as follows:

$$ev_{A,B} : A \otimes (A \Rightarrow B) \to B \qquad \Lambda(f) : C \to (A \Rightarrow B), \quad \text{for } f : (A \otimes C) \to B$$

The closed object $V \Rightarrow W$ is the set of all linear maps from V to W, made into a vector space in the standard way, via the isomorphism $V \Rightarrow W \cong V^* \otimes W \cong V \otimes W$. Hence, the closed structure of *FVect* determines its compact structure. Given this fact, the evaluation map of *FVect* is the same as the corresponding co-unit of the compact adjunction. Hence, for V, W respectively spanned by $\{\vec{v}_i\}_i, \{\vec{w}_j\}_j$, and a vector $\vec{v} \in V$, the morphism

$$V \otimes (V \Rightarrow W) \xrightarrow{ev_{V,W}} W$$

is as in the compact closed case, that is:

$$ev_{V,W}\left(\vec{v} \otimes \left(\sum_{ij} c_{ij}\, \vec{v}_i \otimes \vec{w}_j\right)\right) = \sum_{ij} c_{ij}\, \langle \vec{v} \mid \vec{v}_i \rangle\, \vec{w}_j$$

The quantisation of atomic types yields atomic vector spaces: for a noun phrase n we stipulate $\mathcal{Q}(n) := N$ and for a declarative sentence s we stipulate $\mathcal{Q}(s) := S$. The quantisation of compound types yields closed vector spaces. The quantisation of an intransitive verb, which has the type $n \multimap s$, is computed as follows:

$$\mathcal{Q}(n \multimap s) \quad \cong \quad \mathcal{Q}(n) \Rightarrow \mathcal{Q}(s) \quad := \quad N \Rightarrow S$$

The quantisation of an adjective, which has the type $n \circ\!\!- n$, is computed as follows:

$$\mathcal{Q}(n \circ\!\!- n) \quad \cong \quad \mathcal{Q}(n) \Rightarrow \mathcal{Q}(n) \quad := \quad N \Rightarrow N$$

The quantisation of a transitive verb, which has the type $(n \multimap s) \circ\!\!- n$, is computed as follows:

$$\mathcal{Q}\big((n \multimap s) \circ\!\!- n\big) \quad \cong \quad \mathcal{Q}(n) \Rightarrow \big(\mathcal{Q}(n) \Rightarrow \mathcal{Q}(s)\big) \quad := \quad N \Rightarrow (N \Rightarrow S)$$

Monoidal meaning vectors of strings of words are computed by applying the quantisation of their grammatical reduction map to the tensor products of the quantisations of their words. More precisely, we have:

Definition 6.2. The monoidal meaning vector of a string $\mathrm{w}_1\mathrm{w}_2 \cdots \mathrm{w}_n$ consisting of n words is:

$$\overrightarrow{\mathrm{w}_1\mathrm{w}_2 \cdots \mathrm{w}_n} := \mathcal{Q}(f)(\overrightarrow{\mathrm{w}_1} \otimes \overrightarrow{\mathrm{w}_2} \otimes \cdots \otimes \overrightarrow{\mathrm{w}_n})$$

where for t_i the grammatical type of the word w_i, the map f is the monoidal grammatical reduction map of the string, that is $t_1 \cdot t_2 \cdot \cdots \cdot t_n \xrightarrow{f} s$ and $\mathcal{Q}(f)$ is defined as follows:

$$\mathcal{Q}(t_1 \cdot t_2 \cdot \cdots \cdot t_n \xrightarrow{f} s) := \mathcal{Q}(t_1 \cdot t_2 \cdot \cdots \cdot t_n) \xrightarrow{\mathcal{Q}(f)} \mathcal{Q}(s)$$

For example, the $\mathcal{Q}(f)$ of an intransitive sentence with $f = n \cdot (n \multimap s) \xrightarrow{ev^l_{n,s}} s$, is computed as follows:

$$\begin{aligned}\mathcal{Q}\big(n \cdot (n \multimap s) \xrightarrow{ev^l_{n,s}} s\big) &\cong \mathcal{Q}\big(n \cdot (n \multimap s)\big) \xrightarrow{\mathcal{Q}(ev^l_{n,s})} Q(s)\\ &\cong N \otimes (N \Rightarrow S) \xrightarrow{ev_{N,S}} S\end{aligned}$$

The monoidal meaning of an intransitive sentence 'men kill' is as follows

$$\overrightarrow{\text{men kill}} := \mathcal{Q}(f)(\overrightarrow{\text{men}} \otimes \overrightarrow{\text{kill}})$$

Given that a vector $\vec{v}$ in a vector space V is represented by the morphism $I \xrightarrow{\vec{v}} V$, the details of this computation become as follows:

$$\begin{aligned}\overrightarrow{\text{men kill}} &\cong ev_{N,S}\big((I \xrightarrow{\overrightarrow{\text{men}}} N) \otimes \big(I \xrightarrow{\overrightarrow{\text{kill}}} (N \Rightarrow S)\big)\big)\\ &= ev_{N,S}\big(I \otimes I \xrightarrow{\overrightarrow{\text{men}} \otimes \overrightarrow{\text{kill}}} N \otimes (N \Rightarrow S)\big)\\ &\cong I \xrightarrow{\overrightarrow{\text{men}} \otimes \overrightarrow{\text{kill}}} N \otimes (N \Rightarrow S) \xrightarrow{ev_{N,S}} S\\ &= I \xrightarrow{ev_{N,S}(\overrightarrow{\text{men}} \otimes \overrightarrow{\text{kill}})} S\end{aligned}$$

This morphism picks a sentence vector from S; given that the monoidal and compact meanings of the verb 'kill' coincide and both are representable by $\sum_{ij} c_{ij} \vec{n}_i \otimes \vec{s}_j$, the above morphism picks the vector $\sum_{ij} c_{ij} \langle \overrightarrow{\text{men}} \mid \vec{n}_i \rangle \vec{s}_j$, which is the same vector as the compact meaning vector of 'men kill'.

For the transitive sentence, the grammatical reduction map is the composition of two maps as follows:

$$n \cdot \big((n \multimap s) \circ\!\!- n\big) \cdot n \xrightarrow{1_n \cdot ev^r_{n,n\multimap s}} n \cdot (n \multimap s) \xrightarrow{ev^l_{n,s}} s$$

Hence, f is the composition $ev^l_{n,s} \circ (1_n \cdot ev^r_{n,n\multimap s})$. The quantisation of this composition is the composition of quantisations, computed as follows:

$$\begin{aligned}\mathcal{Q}\big(ev^l_{n,s} \circ (1_n \cdot ev^r_{n,n\multimap s})\big) &\cong \mathcal{Q}\big(ev^l_{n,s}\big) \circ \mathcal{Q}\big(1_n \cdot ev^r_{n,n\multimap s}\big) \quad \text{by functoriality of } \mathcal{Q}\\ &\cong ev_{N,S} \circ \big(\mathcal{Q}(1_n) \otimes \mathcal{Q}\big(ev^r_{n,n\multimap s}\big)\big) \quad \text{by monoidality of } \mathcal{Q}\\ &\cong ev_{N,S} \circ (1_{\mathcal{Q}(n)} \otimes ev_{N,N\Rightarrow S}) \quad \text{by functoriality of } \mathcal{Q}\\ &\cong ev_{N,S} \circ (1_N \otimes ev_{N,N\Rightarrow S})\end{aligned}$$

And based on the above, $\mathcal{Q}(f)$ is computed as follows for a transitive sentence:

$$\mathcal{Q}\big(n \cdot \big((n \multimap s) \circ\!\!- n\big) \cdot n \xrightarrow{ev^l_{n,s} \circ (1_n \cdot ev^r_{n,n\multimap s})} s\big) \cong N \otimes \big(N \Rightarrow (N \Rightarrow S)\big) \otimes N \xrightarrow{ev_{N,S} \circ (1_N \otimes ev_{N,N\Rightarrow S})} S$$

The monoidal meaning vector of a transitive sentence such as 'men kill dogs' simplifies to the following:

$$\overrightarrow{\text{men kill dogs}} := \big(ev_{N,S} \circ (1_N \otimes ev_{N,N\Rightarrow S})\big)(\overrightarrow{men} \otimes \overrightarrow{kill} \otimes \overrightarrow{dogs})$$

Following a computation similar to that of the intransitive sentence, the above picks the sentence vector $\sum_{ijk} c_{ijk} \langle \overrightarrow{\text{men}} \mid \vec{n}_i \rangle \langle \vec{n}_k \mid \overrightarrow{\text{dogs}} \rangle \vec{n}_j$ from S; this vector is the same vector as the one obtained in the compact closed case.

For an adjective-noun phrase, f is $(n \circ\!\!- n) \cdot n \xrightarrow{ev^r_{n,n}} n$, hence $\mathcal{Q}(f)$ becomes as follows:

$$\mathcal{Q}\big((n \circ\!\!- n) \cdot n \xrightarrow{ev^r_{n,n}} n\big) \cong (N \Rightarrow N) \otimes N \xrightarrow{ev_{N,N}} N$$

whose concrete vector is the same as in the compact closed case, that is $\sum_{lm} c_{lm} \ \langle \overrightarrow{\text{noun}} \mid \vec{n}_l \rangle \ \vec{n}_m$.

For a sentence with an adjective-noun phrase, such as 'men kill cute dogs', $\mathcal{Q}(f)$ is computed as follows:

$$\begin{aligned}&\mathcal{Q}\big(n \cdot \big((n \multimap s) \circ\!\!- n\big) \cdot (n \circ\!\!- n) \cdot n \xrightarrow{ev^l_{n,s} \circ (1_n \cdot ev^r_{n,n\multimap s}) \circ (1_{n\cdot(n\multimap n)} \circ\!\!- s \cdot ev^r_{n,n})} s\big)\\ &\quad\cong N \otimes \big(N \Rightarrow (N \Rightarrow S)\big) \otimes (N \Rightarrow N) \otimes N \xrightarrow{ev_{N,S} \circ (1_N \otimes ev_{N,N\Rightarrow S}) \circ (1_{N\otimes(N\Rightarrow(N\Rightarrow S))} \otimes ev_{N,N})} S\end{aligned}$$

The computation of the monoidal meaning vector of the above is done by substituting the meaning vector of the adjective-noun phrase for the meaning of the object in the meaning vector of the transitive sentence. It provides us with the same meaning vector as in the compact closed case.

To compute the monoidal vector meaning of the sentence 'men do not kill cute dogs', we take 'do' to be the name of the identity morphism on $N \Rightarrow S$ and 'not' to be the name of $\overline{\text{not}}$ on the morphism $(N \Rightarrow S) \to (N \Rightarrow S)$. That is:

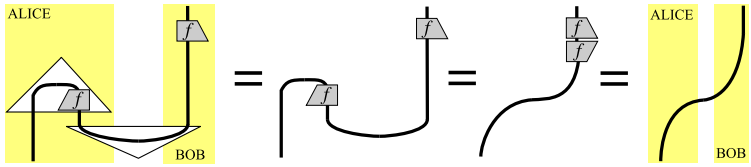


Fig. 2. Diagram of information flow in the teleportation protocol.

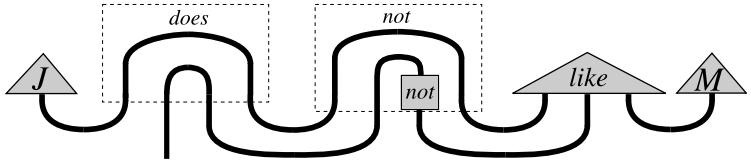


Fig. 3. Diagram of information flow in the negative transitive sentence.

$$\overrightarrow{\text{do}} := \ulcorner 1_{N\Rightarrow S} \urcorner : \mathbb{R} \to \big((N \Rightarrow S) \Rightarrow (N \Rightarrow S)\big)$$
$$\overrightarrow{\text{not}} := \ulcorner \overline{\text{not}} \urcorner : \mathbb{R} \to \big((N \Rightarrow S) \Rightarrow (N \Rightarrow S)\big)$$

The meaning vector of the sentence will then simplify to the following:

$$ev_{N,S}\big(\overrightarrow{men} \otimes \overline{\text{not}}\big(\overrightarrow{kill}(-, \overrightarrow{dogs})\big)\big)$$

So the monoidal meaning of a negative sentence is obtained by first applying the meaning of the verb to the object, then negating it, then applying the result to the subject. This procedure is different from the compact closed case, which was obtained by first applying the meaning of the verb to the meanings of subject and object, then negating it. The monoidal meaning results in the following vector:

$$\left\langle \overrightarrow{\text{men}} \mid \overline{\text{not}}\left(\sum_{ijk} c_{ijk}\, \vec{n}_i \langle \vec{n}_k \mid \overrightarrow{\text{dogs}} \rangle\, \vec{s}_j \right) \right\rangle$$

Whereas the compact meaning results in the following vector:

$$\overline{\text{not}}\left(\sum_{ijk} c_{ijk}\, \langle \overrightarrow{\text{men}} \mid \vec{n}_i \rangle \langle \vec{n}_k \mid \overrightarrow{\text{dogs}} \rangle\, \vec{s}_j \right)$$

These two vectors only become equivalent in special cases, for instance one in which the meaning of $\overline{\text{not}}$ can only act on S (and not on N). This happens in the truth theoretic case, in which N has many dimensions, S has only two dimensions, and $\overline{\text{not}}$ swaps the basis vectors, that is, it is the linear map corresponding the matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. In this case, $\overline{\text{not}}$ can be applied to any vector in S, but it is not defined for the vectors in N. If $\overline{\text{not}}$ is also defined for N, for instance it is a certain permutation of basis vectors, then in the computation of the monoidal meaning vector of the sentence, it can either apply to the basis elements of N, that is to $\vec{n}_i$, or to the basis elements of S, that is to $\vec{s}_j$. Only the latter will provide a result which is the same as the meaning vector of the sentence in the compact case.

7. Diagrammatic reasoning in monoidal bi-closed categories

One of the principal advantages of a DisCoCat is that it is equipped with the sound and complete diagrammatic calculus of compact closed categories [26]. We refer the reader for further details to previous work, but in a nutshell, this diagrammatic calculus depicts the information flows that happen among the words of a sentence and which produce a meaning for the full sentence. This information flow happens when certain objects cancel out via *yanking*; this procedure is depicted by pulling sequences of connected *cup* and *cap* structures, representing ϵ and η maps, and turning them into straight lines. If the information flow happens in stages, the process is depicted via equivalences between the resulting diagram of each stage. The equivalences translate into equations between categorical morphisms corresponding to each diagram and provide a symbolic proof of the claim that for instance, a sentence is grammatical. The yanking operations have also been useful to elegantly describe the flow of information in quantum protocols such as teleportation, as depicted in Fig. 2, from [2].

In a linguistic context, the yanking operations depict the flow of information between words, as depicted in Fig. 3, from [13,51].

The ability to compute with diagrams is not lost in the move from compact to monoidal bi-closed categories. A diagrammatic calculus for these categories has been developed by Baez and Stay [6]. Similar to the graphical calculus of compact

Fig. 4. Basic diagrammatic language constructs (1).

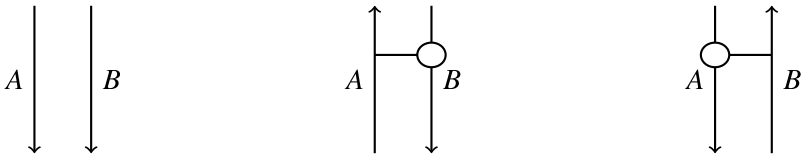


Fig. 5. Basic diagrammatic language constructs (2).

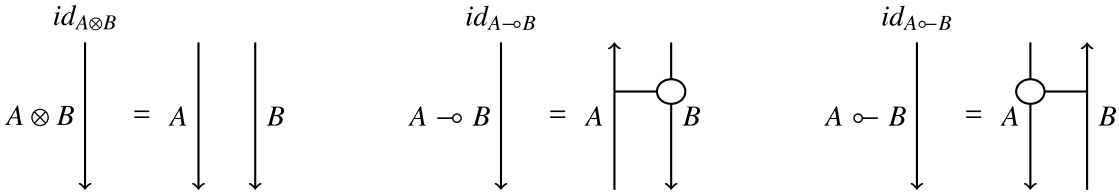


Fig. 6. Equivalent diagrammatic representations.

closed categories, the graphical calculus of monoidal bi-closed categories can be applied to both depict the grammatical reduction of Lambek monoids and the flow of information between meanings of the words in a sentence.

The basic constructs of the diagrammatic language of a monoidal bi-closed category are shown in Figs. 4 and 5. We read these diagrams from top to bottom. Objects of the category are depicted by arrows. For instance the left hand side arrow of Fig. 4 depicts object A. Morphisms of the category are depicted by 'blobs' with 'input' and 'output' arrows standing for their domain and codomain objects. The right hand side blob of Fig. 4 depicts morphism f which has domain A and codomain B. Since an identity morphism has the same domain and codomain, it is depicted in the same way as the object corresponding to it. The identity morphism id_A is denoted by the same diagram as that of object A, e.g. in the left hand side diagram of Fig. 4.

The tensor product of two objects corresponds to the side by side placement of their arrows, for instance the tensor product $A \otimes B$ of two objects A and B is depicted by putting the arrow depicting object A beside the arrow depicting object B, as shown in the left hand side diagram of Fig. 5. Right and left implications $A \multimap B$ and $A \circ\!\!- B$ are depicted by side by side placement and clasping of their arrows, pointing in opposite directions. For instance, the right implication $A \multimap B$ is depicted by putting the arrow corresponding to A pointing upwards with the arrow corresponding to B pointing downwards; with the arrow of B being clasped to the arrow of A. The intuition behind the directions of the arrows is that in a right implication, the antecedent has a negative role and the consequence has a positive role; this reversal of roles is the reason behind the classical logical equivalence $A \to B \cong \neg A \vee B$. The clasp is a restriction: it is there to make us treat the implication as one entity and to prevent us from treating each of the arrows separately. As a result, for example, we cannot apply a function to one arrow, e.g. A and not the other, e.g. B. The diagram corresponding to the left implication $A \circ\!\!- B$ is the opposite of that of the right implication. These two implications are depicted in the last two diagrams of Fig. 5.

There might be more than one way of representing an object or morphism in this diagrammatic calculus. If so, these different ways are considered to be 'equal' and their resulting representations are considered to be 'equivalent'. Fig. 6 shows three of the main Baez–Stay diagrammatic equivalences. They express the fact that one can either draw the left hand side or equivalently the right hand side diagram to represent their corresponding morphism. For instance, one can either draw an arrow labelled with $A \otimes B$ for the tensor object $A \otimes B$ (or the identity arrow on it, i.e. $id_{A\otimes B}$) or two side-by-side arrows, one labelled with A and the other with B. These equivalences are not the same as the equivalences of other diagrammatic calculi [54]. For instance, there is no connection between these and the coherence conditions of the category.

The equivalences for the left and right evaluations ev^l and ev^r are shown in Fig. 7. Each of the diagrams of each equivalence input two arrows and output one arrow. The left hand side diagrams do not depict any information flow: they just show which objects the evaluation morphism is being applied to. The internal structure of the evaluation morphism is being depicted inside the blob of the right hand side diagrams. These diagrams depict how the information flows between an object and the implication, leading to an output. The flow represents the fact that, for instance, for the tensor product

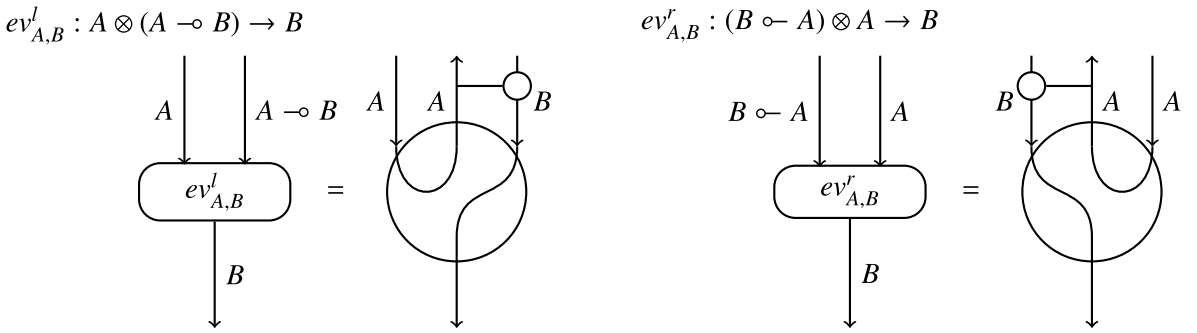


Fig. 7. Diagrams for left and right evaluation.

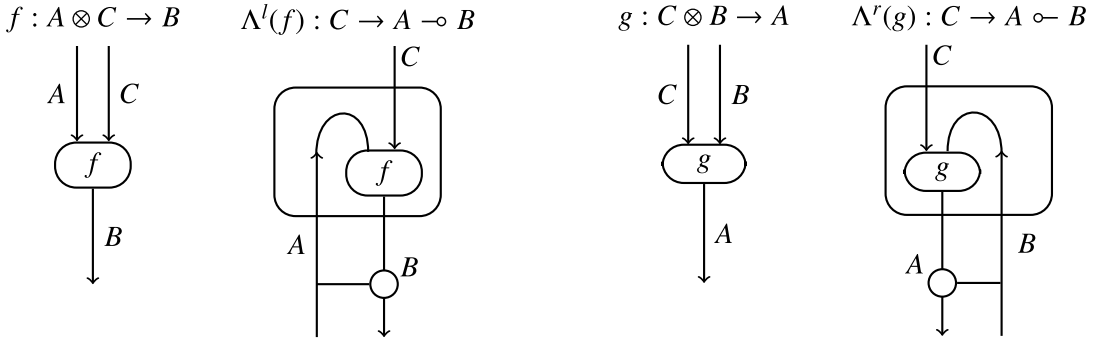


Fig. 8. Diagrams for left and right currying.

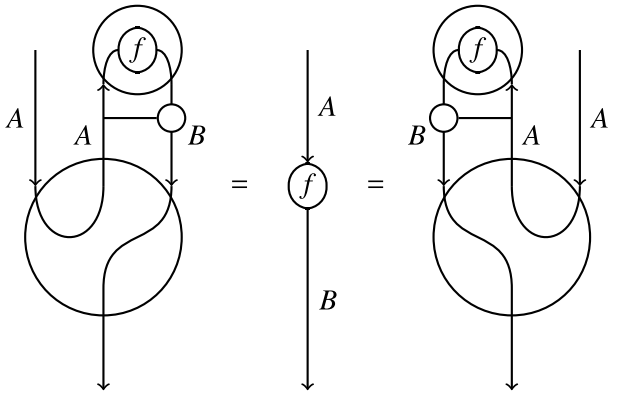


Fig. 9. Diagrams for left and right yanking.

$A \otimes (A \multimap B)$ to be evaluated and to produce the output B, information has to flow from object A to the antecedent of $A \multimap B$, and similarly for the left implication.

The diagrams for left and right currying are shown in Fig. 8. Here, the left hand side diagrams are the morphisms that are being curried and the right hand side diagrams depict the flow of information that happens in the process of currying. What happens outside the blob of the right hand side diagram has no flow and just lists the objects to which the currying morphism is being applied. The internal structure of the blobs depict the corresponding information flows. For instance, the diagram of $\Lambda^l(f)$ shows that in order to produce the morphism $C \to A \multimap B$, the information encoded in A and C have to flow to $f : (A \otimes C) \to B$. For this to happen, first A has to detach itself from the clasp $A \multimap B$, a process that can only happen inside a blob.

The diagrams for the equivalences corresponding to the evaluation of the names of morphisms, which lead to yanking, are shown in Fig. 9 (note that in these diagrams we have stretched the caps inside the outer blobs for reasons of space). They show how, if we first curry f to produce $A \multimap B$, then evaluate the result with A, the output will be the same as the output of applying f to A, that is B.

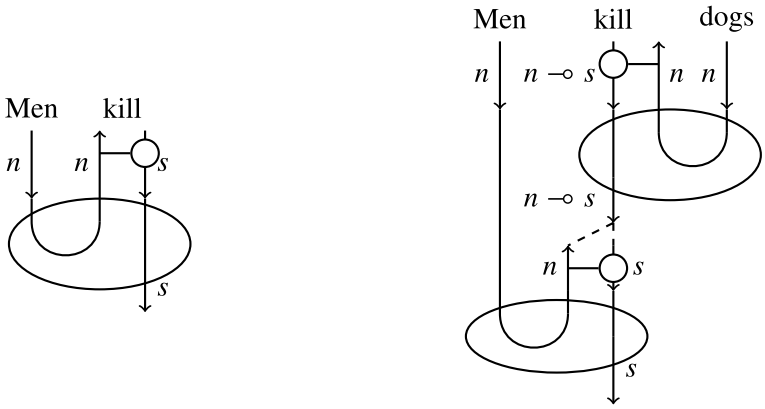


Fig. 10. Diagram for 'men kill' and 'men kill dogs'.

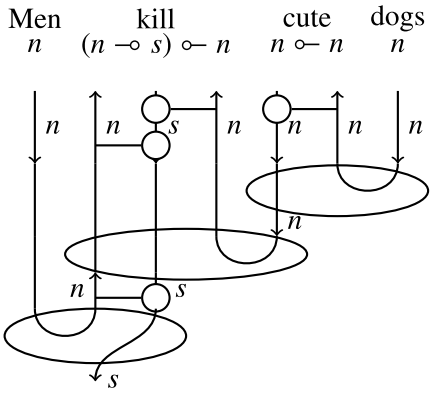


Fig. 11. Diagram for 'men kill cute dogs'.

8. Monoidal bi-closed diagrams and flow of meaning in language

The diagram of the intransitive sentence 'men kill' consists of just one evaluation: $ev^l_{n,s}$. It shows how information has to flow from the subject 'men' to the verb 'kill' to form the intransitive sentence 'men kill'. In other words, the meaning of an intransitive sentence is obtained by applying its verb to its subject. The diagram for this application is depicted in Fig. 10. For reasons of space, we use lowercase letters for both the grammatical types and their interpretation.

For the transitive sentence 'men kill dogs', we first depict the subtype $n \multimap s$ of the verb with a single arrow and do an $ev^r_{n\multimap s,n}$ with the object. Then we use the equivalence for the right implication and replace the resulting object $n \multimap s$ with its equivalent clasped diagram which has two arrows of types n and s respectively. The replacement procedure is marked with a dotted line. At this point, we do an $ev^l_{n,s}$ on the implication diagram $n \multimap s$ and the subject n. The full diagram in Fig. 10, shows that to obtain a meaning for a transitive sentence, information has to flow in two steps: first from the object 'dogs' to the verb 'kill' and then from the subject 'men' to the resulting verb phrase 'kill dogs'. Note that this information flow was done in one step in the compact closed setting, where the subject and object interacted with the verb simultaneously and in one step. Whereas in the monoidal case, we have to wait for the verb to interact with the object before it can interact with the subject.

The diagram for the meaning of 'men kill cute dogs' is obtained by three evaluations, as depicted in Fig. 11. Here, first the adjective acts on the object, then the verb acts on the resulting adjective-noun object to produce a verb phrase, which is then applied to the subject to produce a sentence. The corresponding diagrammatic computations are depicted in Fig. 12. The left hand side diagram shows the verb phrase part of this interaction, whereby, 'cute' is applied to 'dogs' and then input to 'kill' as its object. The result is then applied to the meaning of 'men', this third application is depicted in the right hand side diagram, where the dotted area stands for the general structure of the first and second applications and serves as a shorthand for the left hand side diagram. The arrow between the two diagrams shows that the bottom left side of the first diagram is being inserted into the right hand side and acted upon.

The words whose meaning vectors are *names* of linear maps are obtained by currying the corresponding linear map. For instance, the meaning vector of the auxiliary 'do' is the name of the identity and the meaning vector of the negation word 'not' is the name of a negation operator, as specified in Section 6.2. The diagrams corresponding to the meaning vector of

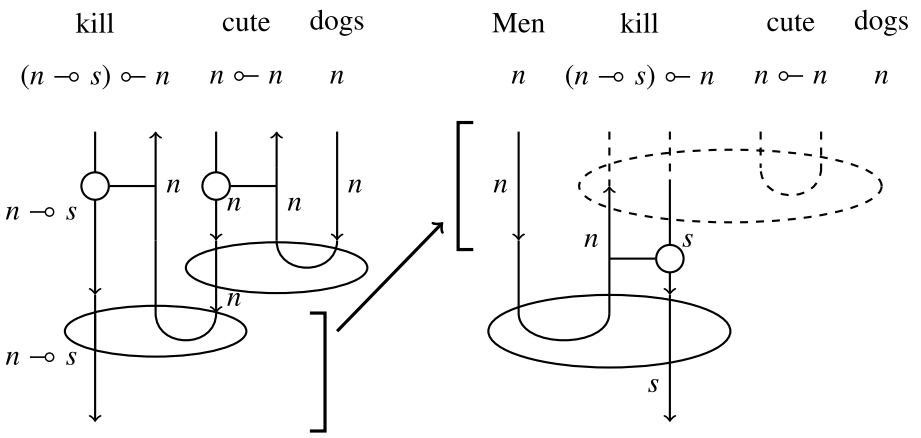


Fig. 12. Diagram for the meaning of 'men kill cute dogs'.

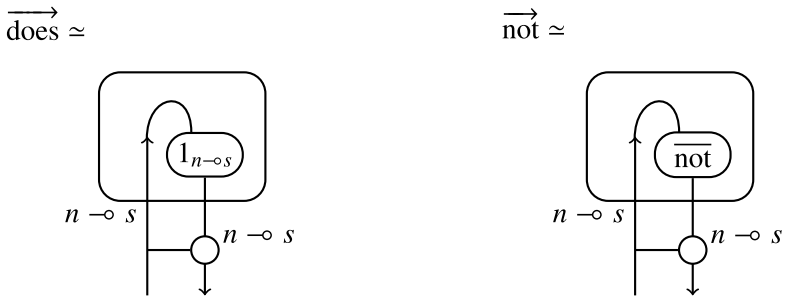


Fig. 13. Diagrams for 'does' and 'not'.

words that are names of certain morphisms are obtained by currying those morphisms. The diagrams of 'does' and 'not' are depicted in Fig. 13.

The meaning vector for 'men do not kill dogs' is obtained by substituting the diagrams of meanings of 'does' and 'not' into the diagram corresponding to the monoidal grammatical reduction of the sentence. The result is depicted in Fig. 14. Here, the diagram on the right is the simplified version of the diagram on the left, where 'kill' has been applied to 'dogs' and the result has been inputted to 'not'. This means that the result is being negated and the negation is then being applied to the meaning of 'men'. As mentioned in Section 6.2, this procedure is different from the compact closed case and this difference, which basically lies in the order of applications, is manifested in the corresponding diagram of each case.

If Lambek monoids are interpreted in vector spaces, as demonstrated in Section 6.2, they benefit from their extra compact structure, for instance symmetry of the tensor. Baez–Stay diagrams can easily be turned into compact string diagrams: by removing the clasp restrictions and popping the bubbles. There is, however, a problem: the resulting diagrams are not always the same as the compact diagrams that were originally drawn for sentences, as demonstrated in the above example. The problem stems from the restricted form of yanking in the monoidal case.

9. Concluding remarks and future work

We have shown that the DisCoCat framework [13,14] need not be committed to the grammatical formalism of Lambek pregroups [32], which are compact closed. Lambek monoids [30], which are monoidal closed, also allow a functorial passage to the category of finite dimensional vector spaces and the diagrammatic calculus of Baez and Stay [6] can be used to depict the information flow that happens within the sentences. This functorial passage is used in and referred to as *quantisation* in the context of Topological Quantum Field Theory (TQFT) [4,5,28]. As future work, we would like to (1) extend the results of this paper to richer type-logics, and (2) relate their diagrammatic calculi to Baez–Stay diagrams.

With regard to (1), we would like to develop distributional compositional models of meaning for richer grammars such as the Combinatorial Categorial Grammar [56] and Lambek–Grishin algebras [44,8]. These grammars have more expressive power and can formalise larger fragments of natural language, and in particular English; CCG has been applied to parse large corpora of real data. To extend our quantisation functor from Lambek monoids to CCG, the cross composition rules need to be given a categorical semantics. This would perhaps require restricting the application of these rules to subcategories of the full type-category, as in their general forms, they do not hold either in a compact or a monoidal closed category. To extend the quantisation functor to Lambek–Grishin algebras, we need to ask the functor to preserve the operations of the Grishin part of the algebra, as well as their interactions with the Lambek part, hence defining functors that have to

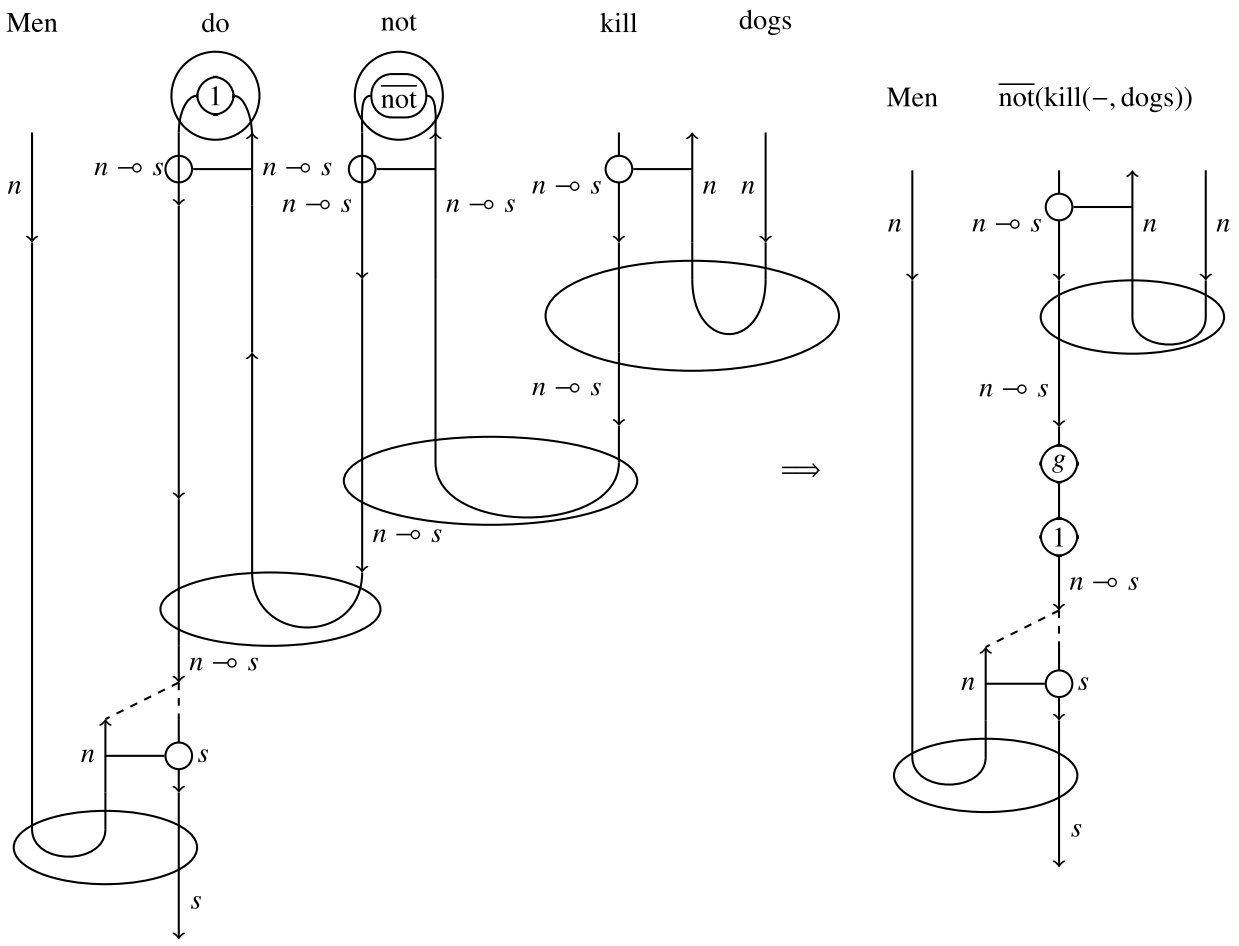


Fig. 14. Diagram for meaning of 'men do not kill dogs'.

satisfy more than the monoidal conditions; the categorical properties of these functors should be studied. Whether or not such extensions prove to be trivial remains to be seen, but it is certain that augmenting the applicability of this general compositional categorical formalism will greatly assist its acceptance by the linguistic community as a practical theoretical toolkit.

As for (2), note that the general structure of Baez–Stay diagrams are the same as parse trees. This is not surprising given the Curry–Howard isomorphism between monoidal categories and lambda calculus [57]. Whereas parse trees are purely syntactical and only depict the grammatical structure of sentences, Baez–Stay diagrams have extra information in their nodes (blobs) about the flow of meaning and the semantic structure of phrases. The straight lines and the cups allow for a flow between the nodes they are connecting; this encodes the order and details of function applications. The clasps, however, stop this flow from happening, hence making it explicit which applications cannot happen. One can identify the clasps with lines and remove the cups to obtain a parse tree without this information. We have demonstrated this procedure for an example sentence in Fig. 15; here, VP stands for an intransitive, transitive, or ditransitive verb phrases, and NP stands for a noun phrase.

The extra information in the blobs of Baez–Stay diagrams makes them more like *proof nets*. A proof net is a diagrammatic notation used to depict the grammatical structures of sentences and their lambda calculus meanings. Proof nets unify proofs, hence solve the spurious ambiguity problem and have been extensively studied by the linear logic [17] and linguistics communities [53,46,42]. They contain semantic information about sentences, represented by *traverse instructions* that describe how to form the lambda term corresponding to a grammatical reduction. These lambda terms can also be directly read from the sequent calculus proofs corresponding to the proof nets. An abstract form of proof nets, referred to as *proof structures*, encode semantic information about sentences to start with. These come equipped with certain *contraction rules* and can be rewritten to parse trees using these rules. The rewrites model the computations that provide us with the final meaning of the sentence. A formal connection between Baez–Stay diagrams and proof nets or structures constitutes future work. As one of the referees suggested, one possibility would be to develop a lambda calculus for Baez–Stay diagrams in the style of *glue semantics* [16].

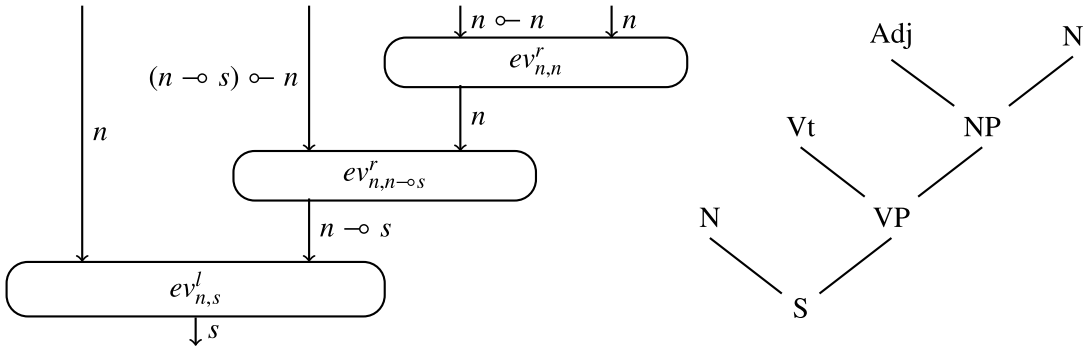


Fig. 15. Baez–Stay diagrams and parse trees.

Acknowledgements

We would like to thank Michael Moortgat, Glyn Morrill, Samson Abramsky, and Mike Stay for fruitful and clarifying discussions.

References

[1] S. Abramsky, Temperley–Lieb algebra: from knot theory to logic and computation via quantum mechanics, in: Mathematics of Quantum Computation and Quantum Technology, in: Chapman & Hall/CRC Appl. Math. Nonlinear Sci. Ser., Chapman and Hall/CRC, Boca Raton, FL, 2008, pp. 515–558.
[2] S. Abramsky, B. Coecke, A categorical semantics of quantum protocols, in: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Science Press, 2004, pp. 415–425, arXiv:quant-ph/0402130.
[3] S. Abramsky, R. Duncan, A categorical quantum logic, Math. Structures Comput. Sci. 16 (3) (2006) 469–489.
[4] M. Atiyah, Topological quantum field theories, Publ. Math. Inst. Hautes Études Sci. 68 (1989) 175–186.
[5] J. Baez, J. Dolan, Higher-dimensional algebra and topological quantum field theory, J. Math. Phys. 36 (1995) 6073–6105.
[6] J. Baez, M. Stay, Physics, topology, logic, and computation: A Rosetta Stone, in: B. Coecke (Ed.), New Structures in Physics, in: Lecture Notes in Phys., vol. 813, Springer, 2011.
[7] Y. Bar-Hillel, A quasiarithmetical notation for syntactic description, Language 29 (1953) 47–58.
[8] R. Bernardi, M. Moortgat, Continuation semantics for the Lambek–Grishin calculus, Inform. and Comput. 208 (2010) 397–416.
[9] W. Buszkowski, Lambek grammars based on pregroups, in: Proceedings of 4th International Conference on Logical Aspects of Computational Linguistics, in: Lecture Notes in Comput. Sci., vol. 2099, 2001, pp. 95–109.
[10] N. Chomsky, Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.
[11] S. Clark, S. Pulman, Combining symbolic and distributional models of meaning, in: Proceedings of AAAI Spring Symposium on Quantum Interaction, AAAI Press, 2007, pp. 52–55.
[12] B. Coecke, Kindergarten quantum mechanics, in: Proceedings of Quantum Theory: Reconsiderations of the Foundations III, AIP Press, 2005, pp. 81–98.
[13] B. Coecke, M. Sadrzadeh, S. Clark, A compositional distributional model of meaning, in: Proceedings of the Second Quantum Interaction Symposium, College Press, 2008, pp. 133–140.
[14] B. Coecke, M. Sadrzadeh, S. Clark, Mathematical foundations for a compositional distributional model of meaning, Linguist. Anal. 36 (1–4) (2010) 345–384.
[15] J.R. Curran, From distributional to semantic similarity, PhD thesis, University of Edinburgh, 2004.
[16] M. Dalrymple, Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach, MIT Press, 1999.
[17] P. de Groote, C. Reotrè, On the semantic readings of proof nets, in: Proceedings of 2nd Formal Grammar Conference, FoLLI, 1996, pp. 57–70.
[18] J.R. Firth, A Synopsis of Linguistic Theory 1930–1955, Stud. Linguist. Anal., vol. 1952-59, The Philological Society, Oxford, 1957, pp. 1–32.
[19] G. Frege, Uber sinn und bedeutung, Z. Philos. Philos. Kritik 100 (1) (1892) 25–50.
[20] G. Grefenstette, Explorations in Automatic Thesaurus Discovery, Kluwer, 1994.
[21] E. Grefenstette, M. Sadrzadeh, Experimental support for a categorical compositional distributional model of meaning, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2011, pp. 1394–1404.
[22] E. Grefenstette, M. Sadrzadeh, Experimenting with transitive verbs in a DisCoCat, in: Proceedings of the EMNLP Workshop on Geometric Models of Natural Language Semantics, 2011.
[23] J. Hintikka, G. Sandu, Game-theoretical semantics, in: Handbook of Logic and Language, MIT Press, 1997, pp. 361–410.
[24] J.J. Jiang, D.W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, in: Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X), 1997.
[25] A. Joyal, R. Street, The geometry of tensor calculus I, Adv. Math. 88 (1991) 55–112.
[26] M. Kelly, M.L. Laplaza, Coherence for compact closed categories, J. Pure Appl. Algebra 19 (1980) 193–213.
[27] A. Kislak-Malinowska, Pregroups with modalities, in: Shuly Wintner (Ed.), Proceedings of Formal Grammar Conference, FoLLI, 2006.
[28] J. Kock, Frobenius Algebras and 2D Topological Quantum Field Theories, Cambridge University Press, 2003.
[29] F. Lamarche, C. Retoré, Proof nets for the Lambek calculus—an overview, in: Proceedings of the Third Roma Workshop. Proofs and Linguistic Categories, 1996, pp. 241–262.
[30] J. Lambek, The mathematics of sentence structure, Amer. Math. Monthly 65 (1958) 154–170.
[31] J. Lambek, On some connections between logic and category theory, Studia Logica 48 (1989) 269–278.
[32] J. Lambek, Type grammar revisited, in: Proceedings of 2nd International Conference on Logical Aspects of Computational Linguistics, in: Lecture Notes in Comput. Sci., vol. 1582, 1999, pp. 1–27.
[33] J. Lambek, Iterated Galois connections in arithmetics and linguistics, in: Galois Connections and Applications, in: Math. Appl., vol. 565, 2004, pp. 389–397.
[34] J. Lambek, From Word to Sentence, Polimetrica, 2008.

[35] J. Lambek, C. Casadio (Eds.), Computational Algebraic Approaches to Natural Language, Polimetrica, Milan, 2006.
[36] T.K. Landauer, S.T. Dumais, A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge, Psychol. Rev. 104 (2) (1997) 211–240.
[37] A. Lecomte, M. Quatrini, Ludics and its applications to natural language semantics, in: Proceedings of 16th International Workshop on Logic, Language, Information and Computation (WoLLIC), in: Lecture Notes in Comput. Sci., vol. 5514, Springer, 2009, pp. 242–255.
[38] J.A. Levin, J.A. Moore, Dialogue-games: metacommunication structures for natural language interaction, Cogn. Sci. 1 (4) (1977) 395–420.
[39] P. Lorenzen, K. Lorenz, Dialogische Logik, Wissenschaftliche Buchgesellschaft, 1978.
[40] J. Mitchell, M. Lapata, Vector-based models of semantic composition, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, 2008, pp. 236–244.
[41] R. Montague, English as a formal language, in: Formal Philosophy, 1974, pp. 188–221.
[42] M. Moortgat, Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus, Foris Publications, 1988.
[43] M. Moortgat, Multimodal linguistic inference, J. Log. Lang. Inf. 5 (3/4) (1996) 349–385.
[44] M. Moortgat, Symmetric categorial grammar, J. Philos. Logic 38 (2009) 681–710.
[45] R. Moot, Proof nets for linguistic analysis, PhD thesis, Utrecht Institute of Linguistics, 2002.
[46] R. Moot, Q. Puite, Proof nets for the multimodal Lambek calculus, Studia Logica 71 (2002) 415–442.
[47] R. Penrose, Applications of negative dimensional tensors, in: Combinatorial Mathematics and Its Applications, Academic Press, 1971, pp. 221–244.
[48] M. Pentus, Lambek grammars are context free, in: Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 429–433.
[49] A. Preller, Towards discourse representation via pregroup grammars, J. Log. Lang. Inf. (2007).
[50] A. Preller, J. Lambek, Free compact 2-categories, Math. Structures Comput. Sci. 17 (2) (2007) 309–340.
[51] A. Preller, M. Sadrzadeh, Bell states and negative sentences in the distributed model of meaning, in: P. Selinger, B. Coecke, P. Panangaden (Eds.), Proceedings of the 6th QPL Workshop on Quantum Physics and Logic, in: Electron. Notes Theor. Comput. Sci., University of Oxford, 2010.
[52] S.G. Pulman, Conversational games, belief revision and Bayesian networks, in: J. Landsbergen, et al. (Eds.), CLIN VII: Proceedings of 7th Computational Linguistics in the Netherlands Meeting, Nov. 1996, 1997, pp. 1–25.
[53] D. Roorda, Proof nets for Lambek calculus, J. Logic Comput. 2 (1992) 211–231.
[54] P. Selinger, A survey of graphical languages for monoidal categories, in: B. Coecke (Ed.), New Structures for Physics, in: Lecture Notes in Phys., Springer, 2010, pp. 275–337.
[55] P. Smolensky, G. Legendre, The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar, vol. I: Cognitive Architecture, MIT Press, 2005;
P. Smolensky, G. Legendre, The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar, vol. II: Linguistic and Philosophical Implications, MIT Press, 2005.
[56] M. Steedman, The Syntactic Process, MIT Press, 2001.
[57] M.E. Szabo, Algebra of Proofs, Stud. Logic Found. Math., vol. 88, North-Holland, 1978.
[58] O. Valentin, G. Morrill, M. Fadda, The displacement calculus, J. Log. Lang. Inf. 20 (2011) 1–48.
[59] L. Wittgenstein, Philosophical Investigations, Blackwell, 1953.