

Journal of Language Modelling

VOLUME 6 ISSUE 2
DECEMBER 2018

Editorials

A special issue on statistical and logical models of meaning 223
Glyn Morrill, Mehrnoosh Sadrzadeh

Articles

Graded hyponymy for compositional distributional semantics 225
Dea Bankova, Bob Coecke, Martha Lewis, Dan Marsden

A proof-theoretic approach
to scope ambiguity in compositional vector space models 261
Gijs Jasper Wijnholds

Combining logical and distributional methods
in type-logical grammars 287
Richard Moot

Static and dynamic vector semantics
for lambda calculus models of natural language 319
Mehrnoosh Sadrzadeh, Reinhard Muskens

Squibs and discussions

A note on movement in logical grammar 353
Glyn Morrill

External reviewers 2016–2018 365



JOURNAL OF
LANGUAGE MODELLING

ISSN 2299-8470 (electronic version)

ISSN 2299-856X (printed version)

<http://jlm.ipipan.waw.pl/>

MANAGING EDITOR

Adam Przepiórkowski IPI PAN

GUEST EDITORS OF THIS SPECIAL ISSUE

Glyn Morrill Universitat Politècnica de Catalunya

Mehrnoosh Sadrzadeh Queen Mary University of London

SECTION EDITORS

Elżbieta Hajnicz IPI PAN

Agnieszka Mykowiecka IPI PAN

Marcin Woliński IPI PAN

STATISTICS EDITOR

Łukasz Dębowski IPI PAN



Published by IPI PAN

Institute of Computer Science, Polish Academy of Sciences
ul. Jana Kazimierza 5, 01-248 Warszawa, Poland

Circulation: 100 + print on demand

Layout designed by Adam Twardoch.

Typeset in X_YL^AT_EX using the typefaces: *Playfair Display*
by Claus Eggers Sørensen, *Charis SIL* by SIL International,
JLM monogram by Łukasz Dziedzic.

*All content is licensed under
the Creative Commons Attribution 3.0 Unported License.*
<http://creativecommons.org/licenses/by/3.0/>



EDITORIAL BOARD

Steven Abney University of Michigan, USA

Ash Asudeh Carleton University, CANADA;
University of Oxford, UNITED KINGDOM

Chris Biemann Technische Universität Darmstadt, GERMANY

Igor Boguslavsky Technical University of Madrid, SPAIN;
Institute for Information Transmission Problems,
Russian Academy of Sciences, Moscow, RUSSIA

António Branco University of Lisbon, PORTUGAL

David Chiang University of Southern California, Los Angeles, USA

Greville Corbett University of Surrey, UNITED KINGDOM

Dan Cristea University of Iași, ROMANIA

Jan Daciuk Gdańsk University of Technology, POLAND

Mary Dalrymple University of Oxford, UNITED KINGDOM

Darja Fišer University of Ljubljana, SLOVENIA

Anette Frank Universität Heidelberg, GERMANY

Claire Gardent CNRS/LORIA, Nancy, FRANCE

Jonathan Ginzburg Université Paris-Diderot, FRANCE

Stefan Th. Gries University of California, Santa Barbara, USA

Heiki-Jaan Kaalep University of Tartu, ESTONIA

Laura Kallmeyer Heinrich-Heine-Universität Düsseldorf, GERMANY

Jong-Bok Kim Kyung Hee University, Seoul, KOREA

Kimmo Koskenniemi University of Helsinki, FINLAND

Jonas Kuhn Universität Stuttgart, GERMANY

Alessandro Lenci University of Pisa, ITALY

Ján Mačutek Comenius University in Bratislava, SLOVAKIA

Igor Mel'čuk University of Montreal, CANADA

Glyn Morrill Technical University of Catalonia, Barcelona, SPAIN

Stefan Müller Freie Universität Berlin, GERMANY

Mark-Jan Nederhof University of St Andrews, UNITED KINGDOM

Petya Osenova Sofia University, BULGARIA

David Pesetsky Massachusetts Institute of Technology, USA

Maciej Piasecki Wrocław University of Technology, POLAND

Christopher Potts Stanford University, USA

Louisa Sadler University of Essex, UNITED KINGDOM

Agata Savary Université François Rabelais Tours, FRANCE

Sabine Schulte im Walde Universität Stuttgart, GERMANY

Stuart M. Shieber Harvard University, USA

Mark Steedman University of Edinburgh, UNITED KINGDOM

Stan Szpakowicz School of Electrical Engineering
and Computer Science, University of Ottawa, CANADA

Shravan Vasishth Universität Potsdam, GERMANY

Zygmunt Vetulani Adam Mickiewicz University, Poznań, POLAND

Aline Villavicencio Federal University of Rio Grande do Sul,
Porto Alegre, BRAZIL

Veronika Vincze University of Szeged, HUNGARY

Yorick Wilks Florida Institute of Human and Machine Cognition, USA

Shuly Wintner University of Haifa, ISRAEL

Zdeněk Žabokrtský Charles University in Prague, CZECH REPUBLIC

A Special Issue on Statistical and Logical Models of Meaning

Glyn Morrill¹ and Mehrnoosh Sadrzadeh²

¹ Department of Computer Science, Universitat Politècnica de Catalunya

² School of Electronic Engineering and Computer Science,
Queen Mary University of London

In this special issue we collect five papers derived from the workshop on Statistical and Logical Models of Meaning (SaLMoM) celebrated 11th–15th July 2016 as part of the North American Summer School in Logic, Language and Information at Rutgers University. The workshop addressed fundamental problems of the interplay of syntax, traditional logical semantics, and contemporary distributional semantics. Speakers, in the order of presentation, were Mehrnoosh Sadrzadeh, Gemma Boleda, Laura Rimell, Glyn Morrill, Richard Moot, Jules Hedges, Mark Steedman, Gijs Wijnholds, Kyle Richardson, Dimitri Kartsaklis, Martha Lewis, Reinhard Muskens, and Nicholas Asher (<https://sites.google.com/site/statlogmeaning/workshop-program>).

Since the time of Richard Montague's semantic analysis of phenomena such as quantification around 1970, logical semantics based on lambda calculus and type theory has been a stable paradigm for the characterization of natural language truth conditions (via model theory) and, to a lesser extent, entailment (via proof theory), with the two in principle related by metalogical properties such as soundness and completeness. More recently, statistical semantics has used massive computational corpora to revindicate prior methods such as distributional semantics which models the semantics of a word by the words which locally cooccur with it. At the present time these distinct paradigms appear to represent approaches which are radically distinct, yet both valid. The central objective of this volume is to air options to contrast and possibly reconcile and bring together logical and statistical semantics such as distributional semantics.

Standardly, linguistics identifies dictionary semantics for linguistic study. But in practice language is used for reasoning without a clear distinction between such linguistic knowledge and world knowledge. A classical problem in real reasoning is that of establishing and combining the relevant roles of encyclopedic or world knowledge and linguistic semantic knowledge. The article by Bankova et al. analyses entailment in distributional semantics in relation to this issue.

The article by Wijnholds addresses the classical covert movement phenomenon of quantification in logical syntactic calculi with statistical semantics building on a categorical compositional distributional approach that uses bialgebras to model generalised quantifiers.

The article by Moot proposes weighted logical syntactic proof rules integrating distributional and logical semantics and suggests ways to compare proofs by applying vector similarity measures induced from their weights; the latter are mined from proof banks.

Lambda calculus logical semantic representation is a lingua franca of logical semantics. The article by Muskens and Sadrzadeh presents possible ways for a systematic development of distributional semantics for lambda calculus models of natural language. For this purpose they adopt also lambda grammar syntax, but inessentially, and in this way they furnish proposals which are widely applicable. Their approach allows them to use dynamic logic to reason about admittance of sentences by corpora using concepts similar to those in Heim's context logic.

The squib by Morrill discusses details of logical grammar in relation to the covert movement of relativization and the overt movement of quantification.

The order of papers and a brief description of them are as follows:

1. Bankova et al: entailment in distributional semantics using density matrices,
2. Gijs Wijnholds: scope ambiguity in compositional distributional semantics,
3. Richard Moot: weighted proof rules for distributional + logical semantics,
4. Reinhard Muskens and Mehrnoosh Sadrzadeh: vector semantics for lambda calculus and its logic,
5. Glyn Morrill: a note on movement in logical grammar.

Graded hyponymy for compositional distributional semantics

Dea Bankova¹, Bob Coecke¹, Martha Lewis², and Dan Marsden¹

¹ Quantum Group, University of Oxford

² ILLC, University of Amsterdam

ABSTRACT

The categorical compositional distributional model of natural language provides a conceptually motivated procedure to compute the meaning of a sentence, given its grammatical structure and the meanings of its words. This approach has outperformed other models in mainstream empirical language processing tasks, but lacks an effective model of lexical entailment. We address this shortcoming by exploiting the freedom in our abstract categorical framework to change our choice of semantic model. This allows us to describe hyponymy as a graded order on meanings, using models of partial information used in quantum computation. Quantum logic embeds in this graded order.

Keywords:
categorical
compositional
distributional
semantics,
computational
linguistics,
entailment,
density operator

1

INTRODUCTION

Finding a formalization of language in which the meaning of a sentence can be computed from the meaning of its parts has been a long-standing goal in formal and computational linguistics.

Distributional semantics represents individual word meanings as vectors in finite dimensional real vector spaces. On the other hand, symbolic accounts of meaning combine words via compositional rules to form phrases and sentences. These two approaches are in some sense orthogonal. Distributional schemes have no obvious compositional structure, whereas compositional models lack a canonical way of determining the meaning of individual words. In Coecke *et al.* (2010), the authors develop the categorical compositional distributional model of natural language semantics. This model exploits the

shared categorical structure of pregroup grammars and vector spaces to provide a compositional structure for distributional semantics. It has produced state-of-the-art results in measuring sentence similarity (Kartsaklis *et al.* 2012; Grefenstette and Sadrzadeh 2011), effectively describing aspects of the human understanding of sentences.

A satisfactory account of natural language should incorporate a suitable notion of lexical entailment. Until recently, categorical compositional distributional models of meaning have lacked this crucial feature. In order to address the entailment problem, we exploit the freedom inherent in our abstract categorical framework to change models. We move from a pure state setting to a category used to describe mixed states and partial knowledge in the semantics of categorical quantum mechanics. Meanings are now represented by density matrices rather than simple vectors. We use this extra flexibility to capture the concept of hyponymy, where one word may be seen as an instance of another. For example, *red* is a hyponym of *colour*. The hyponymy relation can be associated with a notion of logical entailment. Some entailment is crisp, for example: *dog* entails *animal*. However, we may also wish to permit entailments of differing strengths. For example, the concept *dog* gives high support to the concept *pet*, but does not completely entail it: some dogs are working dogs. The hyponymy relation we describe here can account for these phenomena. Some crisp entailment can be seen as encoding linguistic knowledge. The kind of entailment we are interested in here is, in general, about the properties that objects have in the world, rather than grammatically based entailment. In particular, we explicitly avoid downward-monotone contexts such as negation. We do, however, examine the hyponymy between an adjective-noun compound and the head noun. We should also be able to measure entailment strengths at the sentence level. For example, we require that *Cujo is a dog* crisply entails *Cujo is an animal*, but that the statement *Cujo is a dog* does not completely entail *Cujo is a pet*. Again, the relation we describe here will successfully describe this behaviour at the sentence level. Closely related to the current work are the ideas in Balkır (2014), Balkır *et al.* (2016), and Sadrzadeh *et al.* (2018). In this work, the authors develop a graded form of entailment based on von Neumann entropy and with links to the distributional inclusion hypotheses developed by Geffet and Dagan (2005). The authors

show how entailment at the word level carries through to entailment at the sentence level. However, this is done without taking account of the grading. In contrast, the measure that we develop here provides a lower bound for the entailment strength between sentences, based on the entailment strength between words. Some of the work presented here was developed in the first author's MSc thesis (Bankova 2015).

An obvious choice for a logic built upon vector spaces is quantum logic (Birkhoff and von Neumann 1936). Briefly, this logic represents propositions about quantum systems as projection operators on an appropriate Hilbert space. These projections form an orthomodular lattice where the distributive law fails in general. The logical structure is then inherited from the lattice structure in the usual way. In the current work, we propose an order that embeds the orthomodular lattice of projections, and so contains quantum logic. This order is based on the Löwner ordering with propositions represented by density matrices. When this ordering is applied to density matrices with the standard trace normalization, no propositions compare, and therefore the Löwner ordering is useless as applied to density operators. The trick we use is to develop an approximate entailment relationship which arises naturally from any commutative monoid. We introduce this in general terms and describe conditions under which this gives a graded measure of entailment. This grading becomes continuous with respect to noise. Our framework is flexible enough to subsume the Bayesian partial ordering of Coecke and Martin (2011) and provides it with a grading. A procedure is given for determining the hyponymy strength between *any* pair of phrases of the same overall grammatical type. The pair of phrases can have differing lengths. So, for example, we can compare 'blond men' to 'men', as these are both noun phrases. This is possible because within categorical compositional semantics, phrases of each type are reduced to one common space according to their type, and can be compared within that space. Furthermore, this notion is consistent with hyponymy at the word level, giving a lower bound on phrase hyponymy.

Density matrices have also been used in other areas of distributional semantics such as Kartsaklis (2015), Piedeleu (2014),

Piedeleu *et al.* (2015), and Blacoe *et al.* (2013). Quantum logic is used in (Widdows and Peters 2003) and Rijsbergen (2004).

Entailment is an important and thriving area of research within distributional semantics. The PASCAL Recognising Textual Entailment Challenge (Dagan *et al.* 2006) has attracted a large number of researchers in the area and generated a number of approaches. Previous lines of research on entailment for distributional semantics investigate the development of directed similarity measures which can characterize entailment (Weeds *et al.* 2004; Kotlerman *et al.* 2010; Lenci and Benotto 2012). Geffet and Dagan (2005) introduce a pair of *distributional inclusion hypotheses*, where if a word v entails another word w , then all the typical features of the word v will also occur with the word w . Conversely, if all the typical features of v also occur with w , v is expected to entail w . Clarke (2009) defines a vector lattice for word vectors, and a notion of graded entailment with the properties of a conditional probability. Rimell (2014) explores the limitations of the distributional inclusion hypothesis by examining the properties of those features that are not shared between words. An interesting approach in Kiela *et al.* (2015) is to incorporate other modes of input into the representation of a word. Measures of entailment are based on the dispersion of a word representation, together with a similarity measure. All of these look at entailment at the word level.

Attempts have also been made to incorporate entailment measures with elements of compositionality. Baroni *et al.* (2012) exploit the entailment relations between adjective-noun and noun pairs to train a classifier that can detect similar relations. They further develop a theory of entailment for quantifiers. The approach that we propose here has the characteristic that it can be applied to more types of phrases and sentences than just adjective-noun and noun-noun type phrases.

Another approach to compositional vector-based entailment is the use of deep neural networks to represent logical semantics, as in Bowman *et al.* (2015), for example. The drawback with the use of this sort of method is that the transparency of the compositional method is lost: the networks may indeed learn how to represent logical semantics but it is not clear how they do so. In contrast, the method we propose has a clear basis in formal semantics and links to quantum logic.

CATEGORICAL COMPOSITIONAL DISTRIBUTIONAL MEANING

Compositional and distributional accounts of meaning are unified in Coecke *et al.* (2010), constructing the meaning of sentences from the meanings of their component parts using their syntactic structure.

2.1

Pregroup grammars

In order to describe syntactic structure, we use Lambek's pregroup grammars (Lambek 1997). Within the standard categorical compositional distributional model, it is possible to move to other forms of categorial grammar, as argued in Coecke *et al.* (2013). This is due to the fact that the category of finite-dimensional vector spaces is particularly well-behaved, and so grammars with greater or lesser structure may be used. A pregroup $(P, \leq, \cdot, 1, (-)^l, (-)^r)$ is a partially ordered monoid $(P, \leq, \cdot, 1)$ where each element $p \in P$ has a left adjoint p^l and a right adjoint p^r , such that the following inequalities hold:

$$(1) \quad p^l \cdot p \leq 1 \leq p \cdot p^l \quad \text{and} \quad p \cdot p^r \leq 1 \leq p^r \cdot p$$

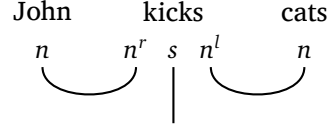
Intuitively, we think of the elements of a pregroup as linguistic types. The monoidal structure allows us to form composite types, and the partial order encodes type reduction. The important right and left adjoints then enable the introduction of types requiring further elements on either their left or right respectively.

The pregroup grammar $\text{Preg}_{\mathcal{B}}$ over an alphabet \mathcal{B} is freely constructed from the atomic types in \mathcal{B} . In what follows we use an alphabet $\mathcal{B} = \{n, s\}$. We use the type s to denote a declarative sentence and n to denote a noun. A transitive verb can then be denoted $n^r s n^l$. If a string of words and their types reduces to the type s , the sentence is judged grammatical. The sentence *John kicks cats* is typed $n (n^r s n^l) n$, and can be reduced to s as follows:

$$n (n^r s n^l) n \leq 1 \cdot s n^l n \leq 1 \cdot s \cdot 1 \leq s$$

This symbolic reduction can also be expressed graphically, as shown in Figure 1. In this diagrammatic notation, the elimination of types by means of the inequalities $n \cdot n^r \leq 1$ and $n^l \cdot n \leq 1$ is denoted by a 'cup'. The fact that the type s is retained is represented by a straight wire.

Figure 1:
A transitive sentence in the graphical calculus



2.2 Compositional distributional models

The symbolic account and distributional approaches are linked by the fact that they are both compact closed categories. This compatibility allows the compositional rules of the grammar to be applied in the vector space model. In this way, we can map syntactically well-formed strings of words into one shared meaning space.

A *compact closed category* is a monoidal category in which for each object A there are left and right dual objects A^l and A^r , and corresponding unit and counit morphisms $\eta^l : I \rightarrow A \otimes A^l$, $\eta^r : I \rightarrow A^r \otimes A$, $\epsilon^l : A^l \otimes A \rightarrow I$, $\epsilon^r : A \otimes A^r \rightarrow I$ such that the *snake equations* hold:

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

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

The underlying poset of a pregroup can be viewed as a compact closed category with the monoidal structure given by the pregroup monoid, and $\epsilon^l, \eta^l, \eta^r, \epsilon^r$ the unique morphisms witnessing the inequalities of (1).

Distributional vector space models live in the category **FHilb** of finite dimensional real Hilbert spaces and linear maps. **FHilb** is compact closed. Each object V is its own dual and the left and right unit and counit morphisms coincide. Given a fixed basis $\{|v_i\rangle\}_i$ of V , we define the unit by $\eta : \mathbb{R} \rightarrow V \otimes V :: 1 \mapsto \sum_i |v_i\rangle \otimes |v_i\rangle$ and counit by $\epsilon : V \otimes V \rightarrow \mathbb{R} :: \sum_{ij} c_{ij} |v_i\rangle \otimes |v_j\rangle \mapsto \sum_i c_{ii}$. Here, we use the physicists' bra-ket notation, for details see Nielsen and Chuang (2011).

2.3 Graphical calculus

The morphisms of compact closed categories can be expressed in a convenient graphical calculus (Kelly and Laplaza 1980) which we will exploit in the following sections. Objects are labelled wires, and morphisms are given as vertices with input and output wires. Composing morphisms consists of connecting input and output wires, and the tensor product is formed by juxtaposition, as shown in Figure 2.

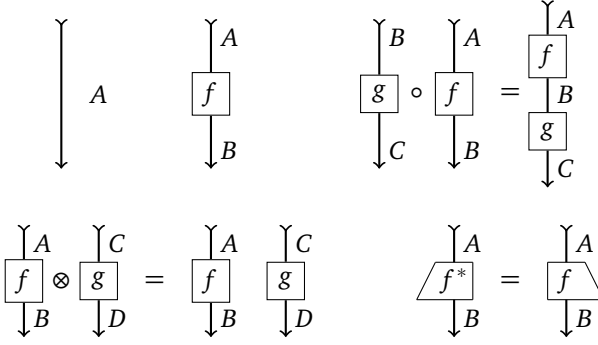


Figure 2:
Monoidal graphical calculus

By convention the wire for the monoidal unit is omitted. The morphisms ϵ and η can then be represented by ‘cups’ and ‘caps’ as shown in Figure 3. The snake equations can be seen as straightening wires, as shown in Figure 4.



Figure 3:
Compact structure graphically

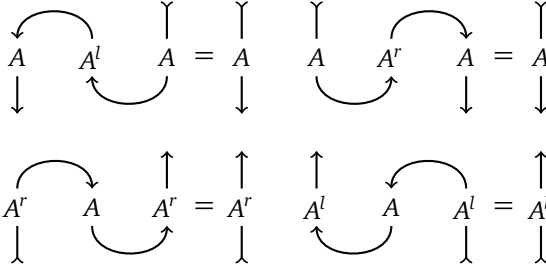


Figure 4:
The snake equations

2.4 Grammatical Reductions in Vector Spaces

Following Preller and Sadrzadeh (2011), reductions of the pregroup grammar may be mapped onto the category **FHilb** of finite dimensional Hilbert spaces and linear maps using an appropriate strong monoidal functor Q :

$$Q : \text{Preg} \rightarrow \text{FHilb}$$

Strong monoidal functors automatically preserve the compact closed structure. For our example $\text{Preg}_{\{n,s\}}$, we must map the noun and

sentence types to appropriate finite dimensional vector spaces:

$$Q(n) = N \quad Q(s) = S$$

Composite types are then constructed functorially using the corresponding structure in **FHilb**. Each morphism α in the pregroup is mapped to a linear map interpreting sentences of that grammatical type. Then, given word vectors $|w_i\rangle$ with types p_i , and a type reduction $\alpha : p_1, p_2, \dots, p_n \rightarrow s$, the meaning of the sentence $w_1 w_2 \dots w_n$ is given by:

$$|w_1 w_2 \dots w_n\rangle = Q(\alpha)(|w_1\rangle \otimes |w_2\rangle \otimes \dots \otimes |w_n\rangle)$$

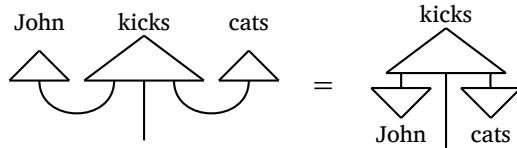
For example, as described in Section 2.1, transitive verbs have type $n^r s n^l$, and can, therefore, be represented in **FHilb** as a rank 3 space $N \otimes S \otimes N$. The transitive sentence *John kicks cats* has type $n(n^r s n^l)n$, which reduces to the sentence type via $\epsilon^r \otimes 1_s \otimes \epsilon^l$. So representing $|kicks\rangle$ by:

$$|kicks\rangle = \sum_{ijk} c_{ijk} |e_i\rangle \otimes |s_j\rangle \otimes |e_k\rangle$$

using the definitions of the counits in **FHilb** we then have:

$$\begin{aligned} |John\ kicks\ cats\rangle &= \epsilon_N \otimes 1_S \otimes \epsilon_N (|John\rangle \otimes |kicks\rangle \otimes |cats\rangle) \\ &= \sum_{ijk} c_{ijk} \langle John | e_i \rangle \otimes |s_j\rangle \otimes \langle e_k | cats \rangle \\ &= \sum_j \sum_{ik} c_{ijk} \langle John | e_i \rangle \langle e_k | cats \rangle |s_j\rangle \end{aligned}$$

Diagrammatically,



The category **FHilb** is actually a \dagger -compact closed category. A \dagger -compact closed category is a compact closed category with an additional *dagger functor* that is an identity-on-objects involution, satisfying natural coherence conditions. In the graphical calculus, the dagger operation “flips diagrams upside-down”. In the case of **FHilb**

the dagger sends a linear map to its adjoint, and this allows us to reason about inner products in a general categorical setting, so that meanings of sentences may be compared using the inner product to calculate the cosine distance between vector representations.

The abstract categorical framework we have introduced allows meanings to be interpreted not just in \mathbf{FHilb} , but in any \dagger -compact closed category. We will exploit this freedom when we move to density matrices. Detailed presentations of the ideas in this section are given in Coecke *et al.* (2010) and Preller and Sadrzadeh (2011) and an introduction to relevant category theory in Coecke and Paquette (2011).

3 DENSITY MATRICES IN CATEGORICAL COMPOSITIONAL DISTRIBUTIONAL SEMANTICS

3.1 Positive operators and density matrices

The methods outlined in Section 2 can be applied to the richer setting of density matrices. Density matrices are used in quantum mechanics to express uncertainty about the state of a system. For unit vector $|v\rangle$, the projection operator $|v\rangle\langle v|$ onto the subspace spanned by $|v\rangle$ is called a *pure state*. Pure states can be thought of as giving sharp, unambiguous information. In general, density matrices are given by a convex sum of pure states, describing a probabilistic mixture. States that are not pure are referred to as *mixed states*. Necessary and sufficient conditions for an operator ρ to encode such a mixture are:

- $\forall v \in V. \langle v | \rho | v \rangle \geq 0$,
- ρ is self-adjoint,¹
- ρ has trace 1.

Operators satisfying the first two axioms are called *positive operators*. The third axiom ensures that the operator represents a convex mixture of pure states. Relaxing this condition gives us different choices for normalization.

¹ As we are dealing with real-valued positive operators, this condition is necessary.

3.2

Representing words as positive matrices

Within standard distributional semantics, words are represented as vectors, where the values on specific dimensions correspond to some function of the frequency with which they co-occur with the words represented by the basis vectors. The vector space induced can be modified or reduced using singular value decomposition or other techniques, where the basis vectors no longer have specific meanings. In order to represent words as density matrices, we first observe that each word vector has a corresponding pure matrix:

$$|cat\rangle \mapsto |cat\rangle \langle cat|$$

Words which are more general can be built up by taking sums over pure matrices. We can think of the meaning of the word *pet* as represented by:

$$\begin{aligned} \llbracket pet \rrbracket &= p_d |dog\rangle \langle dog| + p_c |cat\rangle \langle cat| + p_t |tarantula\rangle \langle tarantula| + \dots \\ \text{where } \forall i. p_i &\geq 0 \quad \text{and} \quad \sum_i p_i = 1 \end{aligned}$$

In general, we consider the meaning of a word w to be given by a collection of unit vectors $\{|w_i\rangle\}_i$, where each $|w_i\rangle$ represents an instance of the concept expressed by the word. Each $|w_i\rangle$ is weighted by $p_i \in [0, 1]$, such that $\sum_i p_i = 1$. These describe the meaning of w as a weighted combination of exemplars. Then the density operator:

$$\llbracket w \rrbracket = \sum_i p_i |w_i\rangle \langle w_i|$$

represents the word w .

This is an extension of the distributional hypothesis. The coefficients p_i may be determined as a function of the frequency with which each word represented by a pure matrix co-occurs with the word represented by $\llbracket w \rrbracket$, for example.

3.3

The CPM construction

Applying Selinger's CPM construction (Selinger 2007) to **FHilb** produces a new \dagger -compact closed category in which the states are positive operators. This construction has previously been exploited in a linguistic setting in Kartsaklis (2015), Piedeleu et al. (2015), and Balkır et al. (2016).

Throughout this section \mathcal{C} denotes an arbitrary \dagger -compact closed category.

Definition 1 (Completely positive morphism). *A \mathcal{C} -morphism $\varphi : A^* \otimes A \rightarrow B^* \otimes B$ is said to be completely positive (Selinger 2007) if there exists $C \in \text{Ob}(\mathcal{C})$ and $k \in \mathcal{C}(C \otimes A, B)$, such that φ can be written in the form:*

$$(k_* \otimes k) \circ (1_{A^*} \otimes \eta_C \otimes 1_A)$$

Identity morphisms are completely positive, and completely positive morphisms are closed under composition in \mathcal{C} , leading to the following:

Definition 2. *If \mathcal{C} is a \dagger -compact closed category then $\text{CPM}(\mathcal{C})$ is a category with the same objects as \mathcal{C} and its morphisms are the completely positive morphisms.*

The \dagger -compact structure required for interpreting language in our setting lifts to $\text{CPM}(\mathcal{C})$:

Theorem 1. *$\text{CPM}(\mathcal{C})$ is also a \dagger -compact closed category. There is a functor:*

$$\begin{aligned} E : \mathcal{C} &\rightarrow \text{CPM}(\mathcal{C}) \\ k &\mapsto k_* \otimes k \end{aligned}$$

This functor preserves the \dagger -compact closed structure, and is faithful “up to a global phase” (Selinger 2007).


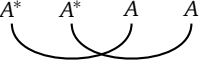

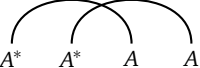
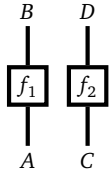
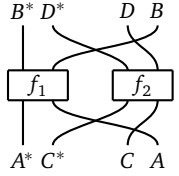
3.4 Diagrammatic calculus for $\text{CPM}(\mathcal{C})$

As $\text{CPM}(\mathcal{C})$ is also a \dagger -compact closed category, we can use the graphical calculus described in Section 2.3. By convention, the diagrammatic calculus for $\text{CPM}(\mathcal{C})$ is drawn using thick wires. The corresponding diagrams in \mathcal{C} are given in Table 1.

In the vector space model of meaning the transition between syntax and semantics was achieved by using a strong monoidal functor $Q : \text{Preg} \rightarrow \text{FHilb}$. Language can be assigned semantics in $\text{CPM}(\text{FHilb})$ in an entirely analogous way via a strong monoidal functor:

$$S : \text{Preg} \rightarrow \text{CPM}(\text{FHilb})$$

Table 1:
Table of diagrams in $\text{CPM}(\mathcal{C})$ and \mathcal{C}

$\text{CPM}(\mathcal{C})$	\mathcal{C}
$E(\epsilon) = \epsilon_* \otimes \epsilon$ 	$\epsilon : A^* \otimes A^* \otimes A \otimes A \rightarrow I$ 
$\epsilon : e_i\rangle \otimes e_j\rangle \otimes e_k\rangle \otimes e_l\rangle \mapsto \langle e_i e_k \rangle \langle e_j e_l \rangle$	
$E(\eta) = \eta_* \otimes \eta$ 	$\eta : I \rightarrow A \otimes A \otimes A^* \otimes A^*$ 
$\eta : 1 \mapsto \sum_{i,j} e_i\rangle \otimes e_j\rangle \otimes e_i\rangle \otimes e_j\rangle$	
	
$f_1 \otimes f_2 : A^* \otimes C^* \otimes C \otimes A \rightarrow B^* \otimes D^* \otimes D \otimes B$	

Definition 3. Let $w_1, w_2 \dots w_n$ be a string of words with corresponding grammatical types t_i in $\text{Preg}_{\mathcal{B}}$. Suppose that the type reduction is given by $t_1, \dots, t_n \xrightarrow{r} x$ for some $x \in \text{Ob}(\text{Preg}_{\mathcal{B}})$. Let $\llbracket w_i \rrbracket$ be the meaning of word w_i in $\text{CPM}(\text{FHilb})$, i.e. a state of the form $I \rightarrow S(t_i)$. Then the meaning of $w_1 w_2 \dots w_n$ is given by:

$$\llbracket w_1 w_2 \dots w_n \rrbracket = S(r)(\llbracket w_1 \rrbracket \otimes \dots \otimes \llbracket w_n \rrbracket)$$

We now have all the ingredients to derive sentence meanings in $\text{CPM}(\text{FHilb})$.

Example 1. We firstly show that the results from FHilb lift to $\text{CPM}(\text{FHilb})$. Let the noun space N be a real Hilbert space with basis vectors given by $\{|n_i\rangle\}_i$, where for some i , $|n_i\rangle = |\text{Clara}\rangle$ and for some j , $|n_j\rangle = |\text{beer}\rangle$. Let the sentence space be another space S with basis $\{|s_i\rangle\}_i$. The verb $|\text{likes}\rangle$ is given by:

$$|\text{likes}\rangle = \sum_{pqr} C_{pqr} |n_p\rangle \otimes |s_q\rangle \otimes |n_r\rangle$$

The density matrices for the nouns Clara and beer are in fact pure states given by:

$$\llbracket \text{Clara} \rrbracket = |n_i\rangle \langle n_i| \quad \text{and} \quad \llbracket \text{beer} \rrbracket = |n_j\rangle \langle n_j|$$

and similarly, $\llbracket \text{likes} \rrbracket$ in $\text{CPM}(\text{FHilb})$ is:

$$\llbracket \text{likes} \rrbracket = \sum_{pqr\,tuv} C_{pqr} C_{tuv} |n_p\rangle \langle n_t| \otimes |s_q\rangle \langle s_u| \otimes |n_r\rangle \langle n_v|$$

The meaning of the composite sentence is simply $(\varepsilon_N \otimes 1_S \otimes \varepsilon_N)$ applied to $(\llbracket \text{Clara} \rrbracket \otimes \llbracket \text{likes} \rrbracket \otimes \llbracket \text{beer} \rrbracket)$ as shown in Figure 5, with interpretation in FHilb shown in Figure 6.

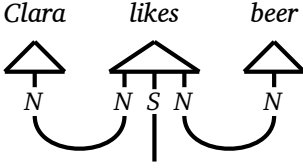


Figure 5:
A transitive sentence in $\text{CPM}(\mathcal{C})$

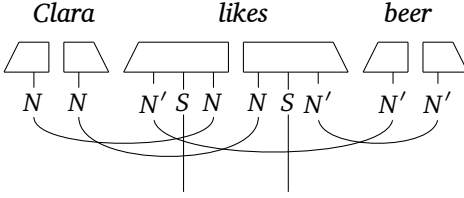


Figure 6:
A transitive sentence in \mathcal{C} with pure states

In terms of linear algebra, this corresponds to:

$$\begin{aligned} \llbracket \text{Clara likes beer} \rrbracket &= \varphi(\llbracket \text{Clara} \rrbracket \otimes \llbracket \text{likes} \rrbracket \otimes \llbracket \text{beer} \rrbracket) \\ &= \sum_{qu} C_{iqj} C_{iuj} |s_q\rangle \langle s_u| \end{aligned}$$

This is a pure state corresponding to the vector $\sum_q C_{iqj} |s_q\rangle$.

However, in $\text{CPM}(\text{FHilb})$ we can work with more than the pure states.

Example 2. Let the noun space N be a real Hilbert space with basis vectors given by $\{|n_i\rangle\}_i$. Let:

$$|\text{Annie}\rangle = \sum_i a_i |n_i\rangle, |\text{Betty}\rangle = \sum_i b_i |n_i\rangle, |\text{Clara}\rangle = \sum_i c_i |n_i\rangle$$

$$|beer\rangle = \sum_i d_i |n_i\rangle, \quad |wine\rangle = \sum_i e_i |n_i\rangle$$

and with the sentence space S , we define:

$$|likes\rangle = \sum_{pqr} C_{pqr} |n_p\rangle \otimes |s_q\rangle \otimes |n_r\rangle$$

$$|appreciates\rangle = \sum_{pqr} D_{pqr} |n_p\rangle \otimes |s_q\rangle \otimes |n_r\rangle$$

Then, we can set:

$$\llbracket the\ sisters \rrbracket = \frac{1}{3}(|Annie\rangle \langle Annie| + |Betty\rangle \langle Betty| + |Clara\rangle \langle Clara|)$$

$$\llbracket drinks \rrbracket = \frac{1}{2}(|beer\rangle \langle beer| + |wine\rangle \langle wine|)$$

$$\llbracket enjoy \rrbracket = \frac{1}{2}(|like\rangle \langle like| + |appreciate\rangle \langle appreciate|)$$

Then, the meaning of the sentence:

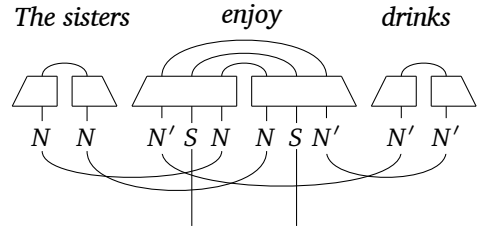
$$s = \text{The sisters enjoy drinks}$$

is given by:

$$\llbracket s \rrbracket = (\varepsilon_N \otimes 1_S \otimes \varepsilon_N)(\llbracket the\ sisters \rrbracket \otimes \llbracket enjoy \rrbracket \otimes \llbracket drinks \rrbracket)$$

Diagrammatically, this is shown in Figure 7.

Figure 7:
A transitive sentence in \mathcal{C} with impure states



The impurity is indicated by the fact that the pairs of states are connected by wires (Selinger 2007).

4 PREDICATES AND ENTAILMENT

If we consider a model of (non-deterministic) classical computation, a state of a set X is just a subset $\rho \subseteq X$. Similarly, a predicate is a subset $A \subseteq X$. We say that ρ satisfies A if:

$$\rho \subseteq A$$

which we write as $\rho \Vdash A$. Predicate A entails predicate B , written $A \models B$, if for every state ρ :

$$\rho \Vdash A \Rightarrow \rho \Vdash B$$

Clearly this is equivalent to requiring $A \subseteq B$.

4.1 *The Löwner order*

As our linguistic models derive from a quantum mechanical formalism, positive operators form a natural analogue for subsets as our predicates. This follows ideas in D'Hondt and Panangaden (2006) and earlier work in a probabilistic setting in Kozen (1983). Crucially, we can order positive operators (Löwner 1934).

Definition 4 (Löwner order). *For positive operators A and B , we define:*

$$A \subseteq B \iff B - A \text{ is positive}$$

If we consider this as an entailment relationship, we can follow our intuitions from the non-deterministic setting. Firstly, we introduce a suitable notion of satisfaction. For positive operator A and density matrix ρ , we define $\rho \Vdash A$ as the positive real number $\text{tr}(\rho A)$.

This generalizes satisfaction from a binary relation to a binary function into the positive reals. We then find that the Löwner order can equivalently be phrased in terms of satisfaction as follows:

Lemma 1 (D'Hondt and Panangaden 2006). *Let A and B be positive operators. $A \subseteq B$ if and only if for all density operators ρ :*

$$\rho \Vdash A \leq \rho \Vdash B$$

Linguistically, we can interpret this condition as saying that every noun, for example, satisfies predicate B at least as strongly as it satisfies predicate A .

4.2 *Quantum logic*

Quantum logic (Birkhoff and von Neumann 1936) views the projection operators on a Hilbert space as propositions about a quantum system. As the Löwner order restricts to the usual ordering on projection operators, we can embed quantum logic within the poset of projection operators, providing a direct link to existing theory.

4.3 *A general setting for approximate entailment*

We can build an entailment preorder on any commutative monoid, viewing the underlying set as a collection of propositions. We then write $A \models B$ and say A entails B if there exists a proposition D such that $A + D = B$. If our commutative monoid is the powerset of some set X , with union the binary operation and unit the empty set, then we recover our non-deterministic computation example from the previous section. If, on the other hand, we take our commutative monoid to be the positive operators on some Hilbert space, with addition of operators and the zero operator as the monoid structure, we recover the Löwner ordering.

In linguistics, we may ask ourselves: does *dog* entail *pet*? Naïvely, the answer is clearly no, not every dog is a pet. This seems too crude for realistic applications though, most dogs are pets, and so we might say *dog* entails *pet* to some extent. This motivates our need for an approximate notion of entailment.

For proposition E , we say that A entails B to the extent E if:

$$A \models B + E$$

We think of E as a error term, for instance in our dogs and pets example, E adds back in dogs that are not pets. Expanding definitions, we find A entails B to extent E if there exists D such that:

$$(2) \quad A + D = B + E$$

From this more symmetrical formulation it is easy to see that for arbitrary propositions A, B , proposition A trivially entails B to extent A , as by commutativity:

$$A + B = B + A$$

It is therefore clear that the mere existence of a suitable error term is not sufficient for a weakened notion of entailment. If we restrict our attention to errors in a complete meet semilattice $\mathcal{E}_{A,B}$, we can take the lower bound on the E satisfying equation (2) as our canonical choice. Finally, if we wish to be able to compare entailment strengths globally, this can be achieved by choosing a partial order \mathcal{K} of “error sizes” and monotone functions:

$$\mathcal{E}_{A,B} \xrightarrow{\kappa_{A,B}} \mathcal{K}$$

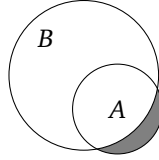
sending errors to their corresponding size.

For example, if A and B are positive operators, we take our complete lattice of error terms $\mathcal{E}_{A,B}$ to be all operators of the form $(1-k)A$ for $k \in [0, 1]$, ordered by the size of $1-k$. We then take k as the strength of the entailment, and refer to it as k -hyponymy.

In the case of finite sets A, B , we take $\mathcal{E}_{A,B} = \mathcal{P}(A)$, and take the size of the error terms as:

$$\frac{\text{cardinality of } E}{\text{cardinality of } A}$$

measuring “how much” of A we have to supplement B with, as indicated in the shaded region below:



In terms of conditional probability, the error size is then:

$$P(A \mid \neg B)$$

These general error terms are strictly more general than the k -hyponymy.

5 HYPONYMY IN CATEGORICAL COMPOSITIONAL DISTRIBUTIONAL SEMANTICS

Modelling hyponymy in the categorical compositional distributional semantics framework was first considered in Balkır (2014). She introduced an asymmetric similarity measure called *representativeness* on density matrices based on quantum relative entropy. This can be used to translate hyponym-hypernym relations to the level of positive transitive sentences. Our aim here will be to provide an alternative measure which relies only on the properties of density matrices and the fact that they are the states in $\text{CPM}(\text{FHilb})$. This will enable us to quantify the *strength* of the hyponymy relationship, described as k -hyponymy. The measure of hyponymy that we use has an advantage over the representativeness measure. Due to the way it combines with linear maps, we can give a quantitative measure to sentence-level entailment based on the entailment strengths between words, whereas representativeness is not shown to combine in this way.

5.1

Properties of hyponymy

Before proceeding with defining the concept of *k-hyponymy*, we give two properties of hyponymy that can be captured by our new measure.

- **Asymmetry.** If A is a hyponym of B, then usually, B is not a hyponym of A.
- **Pseudo-transitivity.** If X is a hyponym of Y and Y is a hyponym of Z, then X is a hyponym of Z. However, if the hyponymy is not perfect, then we get a weakened form of transitivity.

The measure of hyponymy that we described above and named *k-hyponymy* will be defined in terms of density matrices – the containers for word meanings. The idea is then to define a quantitative order on the density matrices, which is not a partial order, but does give us an indication of the asymmetric relationship between words.

5.2

Ordering positive matrices

A density matrix can be used to encode the precision that is needed when describing an action. In the sentence *I took my pet to the vet*, we do not know whether the pet is a dog, cat, tarantula, and so on. The sentence *I took my dog to the vet* is more specific. We then wish to develop an order on density matrices so that *dog*, as represented by $|dog\rangle\langle dog|$ is more specific than *pet* as represented by $[[pet]]$. This ordering may then be viewed as an entailment relation, and entailment between words can lift to the level of sentences, so that the sentence *I took my dog to the vet* entails the sentence *I took my pet to the vet*. Note that we do not require that the sentences have exactly the same structure. For example, we would like *I took my brown dog to the vet* to entail *I took my dog to the vet*, and we would expect this to happen because *brown dog* should entail *dog*.

We now define our notion of approximate entailment, following the discussions of Section 4.3:

Definition 5 (*k-hyponym*). We say that A is a *k-hyponym* of B for a given value of *k* in the range $(0, 1]$ and write $A \preceq_k B$ if:

$$0 \subseteq B - kA$$

Note that such a *k* need not be unique or even exist at all.

Definition 6 (k_{max} hyponym). k_{max} is the maximum value of $k \in (0, 1]$ for which we have $A \preceq_{k_{max}} B$.

In general, we are interested in the maximal value k_{max} for which k -hyponymy holds between two positive operators. This k_{max} value quantifies the strength of the entailment between the two operators.

In what follows, for operator A we write A^+ for the corresponding Moore-Penrose pseudo-inverse and $supp(A)$ for the support of A .

Lemma 2 (Balkır 2014). Let A, B be positive operators.

$$supp(A) \subseteq supp(B) \iff \exists k. k > 0 \text{ and } B - kA \geq 0$$

Lemma 3. For positive self-adjoint matrices A, B such that:

$$supp(A) \subseteq supp(B)$$

B^+A has non-negative eigenvalues.

We now develop an expression for the optimal k in terms of the matrices A and B .

Theorem 2. For positive self-adjoint matrices A, B such that:

$$supp(A) \subseteq supp(B)$$

the maximum k such that $B - kA \geq 0$ is given by $1/\lambda$ where λ is the maximum eigenvalue of B^+A .

Proof. We wish to find the maximum k for which

$$\forall |x\rangle \in \mathbb{R}^n. \langle x | (B - kA) | x \rangle \geq 0$$

Since $supp(A) \subseteq supp(B)$, such a k exists. We assume that for $k = 1$, there is at least one $|x\rangle$ such that $\langle x | (B - kA) | x \rangle \leq 0$, since otherwise we're done. For all $|x\rangle \in \mathbb{R}^n$, $\langle x | (B - kA) | x \rangle$ increases continuously as k decreases. We therefore decrease k until $\langle x | (B - kA) | x \rangle \geq 0$, and there will be at least one $|x_0\rangle$ at which $\langle x_0 | (B - kA) | x_0 \rangle = 0$. These points are minima so that the vector of partial derivatives $\nabla \langle x_0 | (B - k_0A) | x_0 \rangle = 2(B - k_0A) | x_0 \rangle = \vec{0}$ (requires B, A self-adjoint).

Therefore $B | x_0 \rangle = k_0 A | x_0 \rangle$, and so $1/k_0 B^+ B | x_0 \rangle = B^+ A | x_0 \rangle$. Since $B^+ B$ is a projector onto the support of B and $supp(A) \subseteq supp(B)$, we have:

$$1/k_0 | v_0 \rangle = B^+ A | v_0 \rangle$$

where $| v_0 \rangle = B^+ B | x_0 \rangle$, i.e., $1/k_0$ is an eigenvalue of $B^+ A$.

Now, B^+A has only non-negative eigenvalues, and in fact any pair of eigenvalue $1/k$ and eigenvector $|v\rangle$ will satisfy the condition $B|v\rangle = kA|v\rangle$. We now claim that to satisfy $\forall |x\rangle \in \mathbb{R}^n. \langle x|(B - kA)|x\rangle \geq 0$, we must choose k_0 equal to the reciprocal of the maximum eigenvalue λ_0 of B^+A . For a contradiction, take $\lambda_1 < \lambda_0$, so $1/\lambda_1 = k_1 > k_0 = 1/\lambda_0$. Then we require that $\forall |x\rangle \in \mathbb{R}^n. \langle x|(B - k_1A)|x\rangle \geq 0$, and in particular for $|v_0\rangle$. However:

$$\begin{aligned} \langle v_0|(B - k_1A)|v_0\rangle \geq 0 &\iff \langle v_0|B|v_0\rangle \geq k_1 \langle v_0|A|v_0\rangle \\ &\iff k_0 \langle v_0|A|v_0\rangle \geq k_1 \langle v_0|A|v_0\rangle \\ &\text{contradiction, since } k_0 < k_1 \end{aligned}$$

We therefore choose k_0 equal to $1/\lambda_0$ where λ_0 is the maximum eigenvalue of B^+A , and $\langle x|(B - k_0A)|x\rangle \geq 0$ is satisfied for all $|x\rangle \in \mathbb{R}^n$. \square

5.3 Properties of k -hyponymy

- Reflexivity: k -hyponymy is reflexive for $k = 1$.
- Symmetry: k -hyponymy is neither symmetric nor anti-symmetric.
- Transitivity: k -hyponymy satisfies a version of transitivity. Suppose $A \preceq_k B$ and $B \preceq_l C$. Then $A \preceq_{kl} C$, since:

$$B \subseteq kA \text{ and } C \subseteq lB \implies C \subseteq klA$$

by transitivity of the Löwner order.

For the maximal values $k_{\max}, l_{\max}, m_{\max}$ such that $A \preceq_{k_{\max}} B, B \preceq_{l_{\max}} C$ and $A \preceq_{m_{\max}} C$, we have the inequality $m_{\max} \geq k_{\max}l_{\max}$.

- Continuity: For $A \preceq_k B$, when there is a small perturbation to A , there is a correspondingly small decrease in the value of k . The perturbation must lie in the support of B , but can introduce off-diagonal elements.

Theorem 3. Given $A \preceq_k B$ and density operator ρ such that $\text{supp}(\rho) \subseteq \text{supp}(B)$, then for any $\varepsilon > 0$ we can choose a $\delta > 0$ such that:

$$A' = A + \delta\rho \implies A' \preceq_{k'} B \text{ and } |k - k'| < \varepsilon$$

Proof of Theorem 3. We wish to show that we can choose δ such that $|k - k'| < \varepsilon$. We use the notation $\lambda_{\max}(A)$ for the maximum eigenvalue of A . $A' = A + \delta\rho$ satisfies the condition of Theorem 2, that

$\text{supp}(A') \subseteq \text{supp}(B)$, since suppose $|x\rangle \notin \text{supp}(B)$. $\text{supp}(A) \subseteq \text{supp}(B)$, so $|x\rangle \notin \text{supp}(A)$ and $A|x\rangle = 0$. Similarly, $\rho|x\rangle = 0$. Therefore $(A + \rho)|x\rangle = A'|x\rangle = 0$, so $|x\rangle \notin \text{supp}(A')$.

By Theorem 2 we have:

$$k = \frac{1}{\lambda_{\max}(B+A)}, \quad \text{and} \quad k' = \frac{1}{\lambda_{\max}(B+A')}$$

$$(3) \quad k - k' = \frac{\lambda_{\max}(B^+A') - \lambda_{\max}(B^+A)}{\lambda_{\max}(B^+A')\lambda_{\max}(B^+A)}$$

We may treat the denominator of (3) as a constant. We expand the numerator and apply Weyl's inequalities (Weyl 1912). These inequalities apply only to Hermitian matrices, whereas we need to apply these to products of Hermitian matrices. Since B^+ , A , and ρ are all real-valued positive semidefinite, the products B^+A and $B^+\rho$ have the same eigenvalues as the Hermitian matrices $A^{\frac{1}{2}}B^+A^{\frac{1}{2}}$ and $\rho^{\frac{1}{2}}B^+\rho^{\frac{1}{2}}$. Now:

$$\begin{aligned} \lambda_{\max}(B^+A') - \lambda_{\max}(B^+A) &= \lambda_{\max}(B^+A + \delta B^+\rho) - \lambda_{\max}(B^+A) \\ &\leq \lambda_{\max}(B^+A) + \delta \lambda_{\max}(B^+\rho) - \lambda_{\max}(B^+A) \\ &= \delta \lambda_{\max}(B^+\rho) \leq \delta \lambda_{\max}(B^+)\lambda_{\max}(\rho) \leq \delta \lambda_{\max}(B^+) \end{aligned}$$

Therefore:

$$(4) \quad k - k' \leq \delta \frac{\lambda_{\max}(B^+)}{\lambda_{\max}(B^+A')\lambda_{\max}(B^+A)}$$

so that given ε , A , B , we can always choose a δ to make $k - k' \leq \varepsilon$. \square

5.4

Scaling

When comparing positive operators, in order to standardize the magnitudes resulting from calculations, it is natural to consider normalizing their trace so that we work with density operators. Unfortunately, this is a poor choice when working with the Löwner order as distinct pairs of density operators are never ordered with respect to each other, i.e., for density operators σ , τ , $\sigma \sqsubseteq \tau \Rightarrow \sigma = \tau$. Another option is to bound operators as having maximum eigenvalue 1, as suggested in D'Hondt and Panangaden (2006). With this ordering, the projection operators regain their usual ordering and we recover quantum logic as a suborder of our setting.

Our framework is flexible enough to support other normalization strategies. The optimal choice for linguistic applications is left to future empirical work. Other ideas are also possible. For example we can embed the Bayesian order (Coecke and Martin 2011) within our setting via a suitable transformation on positive operators as follows:

1. Diagonalize the operator, choosing a permutation of the basis vectors such that the diagonal elements are in descending order.
2. Let d_i denote the i^{th} diagonal element. We define the diagonal of a new diagonal matrix inductively as follows:

$$d'_0 = d_0 \quad d'_{i+1} = d'_i * d_{i+1}$$

3. Transform the new operator back to the original basis.

Further theoretical investigations of this type are left to future work.

5.5 *Representing the order in the ‘Bloch disc’*

The Bloch sphere, Bloch (1946), is a geometrical representation of quantum states. Very briefly, points on the sphere correspond to pure states, and states within the sphere to impure states. Since we consider matrices only over \mathbb{R}^2 , we disregard the complex phase which allows us to represent the pure states on a circle. A pure state $\cos(\theta/2)|0\rangle + \sin(\theta/2)|1\rangle$ is represented by the vector $(\sin(\theta), \cos(\theta))$ on the circle.

We can calculate the entailment factor k between any two points on the disc. Figure 8 shows contour maps of the entailment strengths for the state with Bloch vector $v = (\frac{3}{4}\sin(\pi/5), \frac{3}{4}\cos(\pi/5))$, using the maximum eigenvalue normalization.

6 RESULTS ON COMPOSITIONALITY

This section provides results and examples on how the notion of hyponymy we have proposed interacts with the compositionality outlined in Section 2. We firstly give an example showing that phrases of different lengths can be compared. We then give a theorem and example to show that our notion of hyponymy ‘lifts’ to the sentence level, and that the k -values are preserved in a very intuitive fashion.

6.1 *k-hyponymy in phrases of varying length*

We can calculate the extent to which any pair of sentences or phrases are hyponyms of each other. We go back to the simple example in

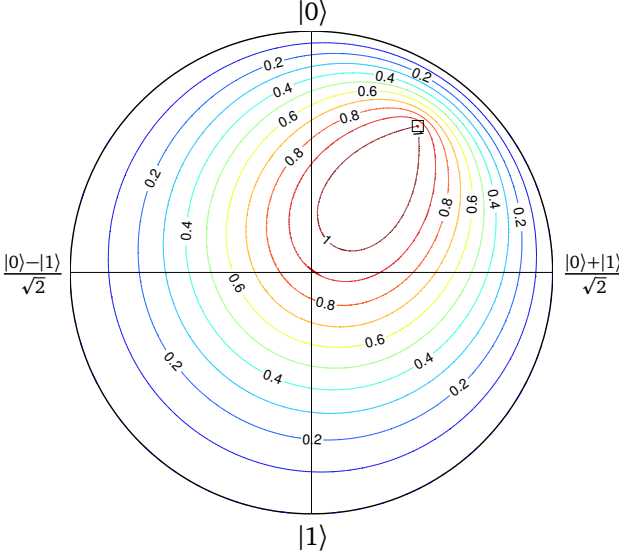


Figure 8:
Entailment strengths
in the Bloch disc for the state
with Bloch vector v

the introduction, comparing ‘blond men’ to ‘men’. Suppose our vector space has basis vectors $|blond\rangle$, $|brunette\rangle$, $|male\rangle$, $|female\rangle$. Then the word ‘men’ can be given by:

$$\llbracket men \rrbracket = \frac{1}{3}(|blond\rangle \langle blond| + |brunette\rangle \langle brunette| + |male\rangle \langle male|)$$

signifying that we are agnostic over all vectors with dimensions $|blond\rangle$, $|brunette\rangle$, $|male\rangle$.

The adjective ‘blond’ is viewed as an operator which takes nouns to blond nouns. This is given by the following:

$$\begin{aligned} \llbracket blond_{adj} \rrbracket &= (|blond\rangle \otimes |blond\rangle)(\langle blond| \otimes \langle blond|) \\ &+ (|blond\rangle \otimes |brunette\rangle)(\langle brunette| \otimes \langle blond|) \\ &+ \sum_{i,j \notin \{blond, brunette\}} (|i\rangle \otimes |i\rangle)(\langle j| \otimes \langle j|) \end{aligned}$$

Then

$$\begin{aligned} \llbracket blond men \rrbracket &= (1_{N \otimes N} \otimes \epsilon_{N \otimes N})(\llbracket blond_{adj} \rrbracket \otimes \llbracket men \rrbracket) \\ &= \frac{2}{3} |blond\rangle \langle blond| + \frac{1}{3} |male\rangle \langle male| \end{aligned}$$

Then if Carlos is described by the pure state

$$|Carlos\rangle = \frac{1}{\sqrt{2}}(|blond\rangle + |male\rangle)$$

we have

$$\llbracket \text{Carlos} \rrbracket = |\text{Carlos}\rangle \langle \text{Carlos}| \preceq_k \llbracket \text{blond men} \rrbracket$$

for $k = \frac{4}{9}$ by Theorem 2. For Janette described by the pure state $|\text{Janette}\rangle = \frac{1}{\sqrt{2}}(|\text{blond}\rangle + |\text{female}\rangle)$, we have

$$\llbracket \text{Janette} \rrbracket = |\text{Janette}\rangle \langle \text{Janette}| \preceq_k \llbracket \text{blond men} \rrbracket$$

for $k = 0$, since $\text{supp}(\llbracket \text{Janette} \rrbracket) \not\subseteq \text{supp}(\llbracket \text{blond men} \rrbracket)$.

An obvious line of enquiry here is to consider how to build this type of adjective operator computationally. One strategy might be to extend the linear regression approach from Baroni and Zamparelli (2010) and Grefenstette *et al.* (2013), having built representations of ‘noun’ and the noun phrase ‘blond noun’. Techniques for building density matrix representations of nouns are described in Sadrzadeh *et al.* (2018).

6.2

Sentence k -hyponymy

We can show that the application of k -hyponymy to various phrase types holds in the same way. In this section we provide a general proof for varying phrase types. We adopt the following conventions:

- A *positive phrase* is assumed to be a phrase in which individual words are upwardly monotone in the sense described by (Barwise and Cooper 1981; MacCartney and Manning 2007). This means that, for example, the phrase does not contain any negations, including words like *not*.
- The *length* of a phrase is the number of words in it, not counting definite and indefinite articles.

Theorem 4 (Sentence k -hyponymy). *Let Φ and Ψ be two positive phrases of the same length and grammatical structure, expressed in the same noun spaces N and sentence spaces S . Denote the words of Φ , in the order in which they appear, by A_1, \dots, A_n . Similarly, denote these in Ψ by B_1, \dots, B_n . Let their corresponding density matrices be denoted by $\llbracket A_1 \rrbracket, \dots, \llbracket A_n \rrbracket$ and $\llbracket B_1 \rrbracket, \dots, \llbracket B_n \rrbracket$ respectively. Suppose that $\llbracket A_i \rrbracket \preceq_{k_i} \llbracket B_i \rrbracket$ for $i \in \{1, \dots, n\}$ and some $k_i \in (0, 1]$. Finally, let φ be the sentence meaning map for both Φ and Ψ , such that $\varphi(\Phi)$ is the meaning of Φ and $\varphi(\Psi)$ is the meaning of Ψ . Then:*

$$\varphi(\Phi) \preceq_{k_1 \dots k_n} \varphi(\Psi)$$

so $k_1 \cdots k_n$ provides a lower bound on the extent to which $\varphi(\Phi)$ entails $\varphi(\Psi)$.

Proof of Theorem 4. First of all, we have $\llbracket A_i \rrbracket \preceq_{k_i} \llbracket B_i \rrbracket$ for $i \in \{1, \dots, n\}$. This means that for each i , we have positive matrices ρ_i and non-negative reals k_i such that $\llbracket B_i \rrbracket = k_i \llbracket A_i \rrbracket + \rho_i$. Now consider the meanings of the two sentences. We have:

$$\begin{aligned} \varphi(\Phi) &= \phi(\llbracket A_1 \rrbracket \otimes \dots \otimes \llbracket A_n \rrbracket) \\ \varphi(\Psi) &= \varphi(\llbracket B_1 \rrbracket \otimes \dots \otimes \llbracket B_n \rrbracket) \\ &= \varphi((k_1 \llbracket A_1 \rrbracket + \rho_1) \otimes \dots \otimes (k_n \llbracket A_n \rrbracket + \rho_n)) \\ &= (k_1 \cdots k_n) \varphi(\llbracket A_1 \rrbracket \otimes \dots \otimes \llbracket A_n \rrbracket) + \varphi(P) \end{aligned}$$

where P consists of a sum of tensor products of positive matrices, namely:

$$P = \sum_{S \subset \{1, \dots, n\}} \bigotimes_{i=1}^n \sigma_i$$

where:

$$(5) \quad \sigma_i = \begin{cases} k_i \llbracket A_i \rrbracket & \text{if } i \in S \\ \rho_i & \text{if } i \notin S \end{cases}$$

Then we have:

$$\varphi(\Psi) - (k_1 \cdots k_n) \varphi(\Phi) = \varphi(P) \geq 0$$

since P is a sum of tensor products of positive matrices, and φ is a completely positive map. Therefore:

$$\varphi(\Phi) \preceq_{k_1 \cdots k_n} \varphi(\Psi)$$

as required. □

Intuitively, this means that if (some of) the words of a sentence Φ are k -hyponyms of (some of) the words of sentence Ψ , then this hyponymy is translated into sentence hyponymy. Upward-monotonicity is important here, in particular as introduced by some implicit quantifiers. It might be objected that *dogs bark* should not imply *pets bark*. If the implicit quantification is universal, then this is true, however

the universal quantifier is downward monotone in the first argument, and therefore does not conform to the convention concerning positive phrases. If the implicit quantification is existential, then *some dogs bark* does entail *some pets bark*, and the problem is averted. Discussion of the behaviour of quantifiers and other word types is given in, for example, Barwise and Cooper (1981) or MacCartney and Manning (2007).

The quantity $k_1 \cdots k_n$ is not necessarily maximal, and indeed usually is not. As we only have a lower bound, zero entailment strength between a pair of components does not imply zero entailment strength between entire sentences.

Corollary 1. *Consider two sentences:*

$$\Phi = \bigotimes_i \llbracket A_i \rrbracket \quad \Psi = \bigotimes_i \llbracket B_i \rrbracket$$

such that for each $i \in \{1, \dots, n\}$ we have $\llbracket A_i \rrbracket \sqsubseteq \llbracket B_i \rrbracket$, i.e. there is strict entailment in each component. Then there is strict entailment between the sentences $\varphi(\Phi)$ and $\varphi(\Psi)$.

Proof of Corollary 1. Since $k_i = 1$ for each $i = \{1, \dots, n\}$,

$$\begin{aligned} \varphi(\Phi) \preceq_{k_1 \dots k_n} \varphi(\Psi) &\implies \varphi(\Phi) \preceq_1 \varphi(\Psi) \\ &\implies \varphi(\Phi) \leq \varphi(\Psi) \end{aligned}$$

□

We consider a concrete example. Suppose we have a noun space N with basis $\{|e_i\rangle\}_i$, and sentence space S with basis $\{|x_j\rangle\}_j$. We consider the verbs *nibble*, *scoff* and the nouns *cake*, *chocolate*:



where these nouns and verbs are pure states. The more general *eat* and *sweets* are given by:

$$\text{eat} = \frac{1}{2} \left(\text{nibble} + \text{scoff} \right)$$

Compositional graded hyponymy

$$\begin{array}{c} \text{sweets} \\ \triangleup \\ \uparrow \end{array} = \frac{1}{2} \left(\begin{array}{c} \text{cake} \\ \triangleup \\ \uparrow \end{array} + \begin{array}{c} \text{chocolate} \\ \triangleup \\ \uparrow \end{array} \right)$$

Then

$$\begin{array}{c} \text{scoff} \\ \triangleup \\ \uparrow \end{array} \preccurlyeq_{1/2} \begin{array}{c} \text{eat} \\ \triangleup \\ \uparrow \end{array} \quad \text{and} \quad \begin{array}{c} \text{cake} \\ \triangleup \\ \uparrow \end{array} \preccurlyeq_{1/2} \begin{array}{c} \text{sweets} \\ \triangleup \\ \uparrow \end{array}$$

We consider the sentences:

$$\begin{array}{c} s_1 \\ \triangleup \\ \uparrow \end{array} = \begin{array}{c} \text{John} \quad \text{scoffs} \quad \text{cake} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array}, \quad \begin{array}{c} s_2 \\ \triangleup \\ \uparrow \end{array} = \begin{array}{c} \text{John} \quad \text{eats} \quad \text{sweets} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array}$$

and as per Theorem 4, we will show that $\llbracket s_1 \rrbracket \preccurlyeq_{kl} \llbracket s_2 \rrbracket$ where $kl = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. Expanding $\llbracket s_2 \rrbracket$ we obtain:

$$\begin{array}{c} s_2 \\ \triangleup \\ \uparrow \end{array} = \frac{1}{4} \left(\begin{array}{c} \text{John} \quad \text{scoffs} \quad \text{cake} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} + \begin{array}{c} \text{John} \quad \text{scoffs} \quad \text{choc} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} \right. \\ \left. + \begin{array}{c} \text{John} \quad \text{nibbles} \quad \text{cake} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} + \begin{array}{c} \text{John} \quad \text{nibbles} \quad \text{choc} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} \right)$$

Therefore:

$$\begin{array}{c} s_2 \\ \triangleup \\ \uparrow \end{array} - \frac{1}{4} \begin{array}{c} s_1 \\ \triangleup \\ \uparrow \end{array} = \frac{1}{4} \left(\begin{array}{c} \text{John} \quad \text{scoffs} \quad \text{choc} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} + \begin{array}{c} \text{John} \quad \text{nibbles} \quad \text{cake} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} \right. \\ \left. + \begin{array}{c} \text{John} \quad \text{nibbles} \quad \text{choc} \\ \triangleup \quad \triangleup \quad \triangleup \\ \uparrow \quad \uparrow \quad \uparrow \end{array} \right)$$

We can see that $\llbracket s_2 \rrbracket - \frac{1}{4}\llbracket s_1 \rrbracket$ is positive by positivity of the individual elements and the fact that positivity is preserved under addition and tensor product. Therefore $\llbracket s_1 \rrbracket \preceq_{kl} \llbracket s_2 \rrbracket$ as required.

7

A TOY EXPERIMENT

To investigate the effectiveness of the model we perform a toy experiment using a simplified version of the model. We use the dataset introduced in Balkır *et al.* (2016). This dataset consists of pairs of simple sentences annotated by humans as to whether the first sentence entails the second. Example pairs are:

recommend development \models suggest improvement
progress reduce \models development replace

The first sentence is rated highly by humans for entailment, whereas the second has lower ratings. The sentences are either noun-verb or verb-noun, and they are of the same type within the pairs.

We use simplified models of composition which we detail as follows. The first model is a baseline, where we use only the verb to predict the entailment between the two sentences. For the second and third models, we use the notion of a Frobenius algebra. As described in Kartsaklis *et al.* (2012), we can ‘lift’ lower-order vectors and tensors to higher-order ones. This means that we can obtain a representation for the verb by lifting a density matrix representation. This has the important aspect that the dimensionality needed to represent the word is greatly reduced. In the category $\mathbf{CPM}(\mathbf{FHilb})$, there are two Frobenius algebras we can use. The first equates to a pointwise multiplication of the noun and the verb, and the second is expressed by

$$\rho(s) = \rho(n)^{1/2} \rho(v) \rho(n)^{1/2}$$

where $\rho(s)$, $\rho(n)$, and $\rho(v)$ indicate density matrices for the sentence, noun, and verb respectively.

The last model we examine is an additive model. In general, addition of two positive operators will not be a morphism in $\mathbf{CPM}(\mathbf{FHilb})$. However, in the particular case where the operators are density matrices, we can design a morphism that will implement addition. We give this morphism diagrammatically in Figure 9.

Compositional graded hyponymy

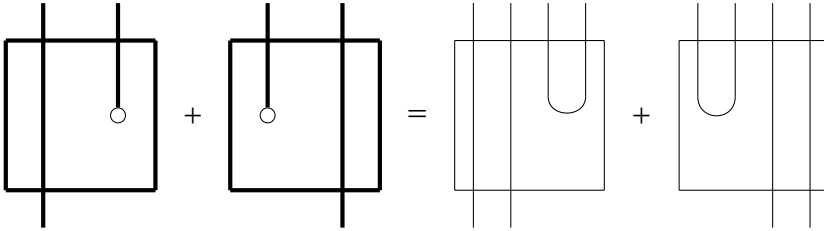


Figure 9:
Morphism
implementing
addition of
density matrices

To build density matrices for the nouns and verbs, we firstly collect a set of hyponyms for each word. To do this, we use WordNet (Miller 1995) via the Natural Language ToolKit (nltk) package in Python (Bird *et al.* 2009). We traverse the WordNet graph below each word to a depth of 8, and collect lemma names of every hyponym encountered. We then use GloVe vectors (Pennington *et al.* 2014) to build representations of each word as follows. Firstly, note that in fact the majority of the hyponyms encountered in WordNet were not present in the off-the-shelf GloVe dataset. Approximately 47,000 hyponyms were found across all words in the sentence pairs, of which approximately 10,000 were in the GloVe dataset. To build the density matrix representations for each word, we simply summed the density matrices corresponding to each GloVe vector for each hyponym of the word, and normalised. We added in some small random values along the diagonal, uniformly distributed over $[0, 10^{-3})$ and renormalised. This step is used to ensure that there is some minimal amount of entailment between every word. After creating sentence vectors from the composition of noun and verb vectors, we calculated the entailment using the result from Theorem 2. We ran the experiments over 50, 100, 200, and 300 dimension vectors. We judged the results by computing Spearman’s ρ between the generated results and the mean of the human judgements. The best results were obtained with 50 dimensional vectors which we report in Table 2.

Model	ρ	p
Verb-only	0.268	> 0.25
Frobenius mult.	0.508	> 0.05
Frobenius n.c.	0.436	> 0.05
Additive	0.643	> 0.001
Inter-annotator	0.66	–

Table 2:
Results in the sentence entailment task

All the compositional models beat the verb-only baseline. The highest scoring model was the additive model, achieving close to inter-annotator agreement. Note that the sentences were extremely simple, and so it would be good to see how the commutative additive model fares when presented with more complex sentences. The best results from Balkir *et al.* (2016) were $\rho = 0.66$ for a vector-based model using the Spearman's ρ metric and our results are comparable. These vectors were built using part-of-speech information which our model did not use, so there is scope for improvement in that direction.

8

CONCLUSION

Integrating a logical framework with compositional distributional semantics is an important step in improving this model of language. By moving to the setting of density matrices, we have described a graded measure of hyponymy that may be used to describe the extent of hyponymy between two words represented within this enriched framework. This approach extends uniformly to provide hyponymy strengths between two phrases of the same type. That type can be any part of speech for which entailment makes sense, such as a noun phrase, verb phrase, or sentence. This includes pairs of phrases with differing numbers of words. We have also shown how a lower bound on hyponymy strength of phrases of the same structure can be calculated from their components.

Whilst we have given a means for modelling hyponymy in a compositional manner, and provided results on how hyponymy strengths compose, the task of integrating logical and distributional semantics is extremely wide-ranging. We mention here a number of areas to which we can start to contribute.

As mentioned in the introduction, some forms of crisp entailment are based in grammatical structure. So, for example, some adjectives interact with nouns to narrow down concepts, as in our example of ‘blond men’, and we therefore have that ‘blond men’ is a hyponym of ‘men’. Other adjectives should not operate in this way, such as *former* in *former president*. This phenomenon is related to the notion of downward monotone contexts and the inclusion of negative words like *not*, or negative prefixes. At present, our model cannot effectively account for downward-monotone phenomena. In order to do so, additional

structure, such as some form of involution, must be added to begin to model these phenomena.

The area of grammatical kinds of entailment also includes phenomena such as verb-phrase ellipsis. The framework developed here is all within the category of pregroups, and in order to be able to model more complex grammatical phenomena, we may need to move to other grammar categories. This has started to be developed in Kartsaklis *et al.* (2016) and we may therefore be able to use these methods within our current model.

The area of quantification is an important one. Hedges and Sadrzadeh (2016) have started to develop a theory of quantification within this framework, and so this is an area in which extension could be possible.

Another line of inquiry is to examine transitivity behaves. In some cases entailment can strengthen. We had that *dog* entails *pet* to a certain extent, and that *pet* entails *mammal* to a certain extent, but that *dog* completely entails *mammal*.

Our framework supports different methods of scaling the positive operators representing propositions. Empirical work will be required to establish the most appropriate method in linguistic applications.

ACKNOWLEDGEMENTS

Bob Coecke, Martha Lewis, and Dan Marsden gratefully acknowledge funding from AFOSR grant Algorithmic and Logical Aspects when Composing Meanings. Martha Lewis gratefully acknowledges funding from NWO Veni grant Metaphorical Meanings for Artificial Agents.

REFERENCES

- Esma BALKIR (2014), *Using Density Matrices in a Compositional Distributional Model of Meaning*, Master's thesis, University of Oxford, <http://www.cs.ox.ac.uk/people/bob.coecke/Esma.pdf>.
- Esma BALKIR, Mehrnoosh SADRZADEH, and Bob COECKE (2016), Distributional Sentence Entailment Using Density Matrices, in Mohammad T. HAJIAGHAYI and Mohammad R. MOUSAVI, editors, *Topics in Theoretical Computer Science*, volume 9541 of *Lecture Notes in Computer Science*, pp. 1–22, Springer, Cham, https://doi.org/10.1007/978-3-319-28678-5_1.

- Dea BANKOVA (2015), *Comparing Meaning in Language and Cognition: P-Hyponymy, Concept Combination, Asymmetric Similarity*, Master's thesis, University of Oxford,
<http://www.cs.ox.ac.uk/people/bob.coecke/Dea.pdf>.
- Marco BARONI, Raffaella BERNARDI, Ngoc-Quynh DO, and Chung-chieh SHAN (2012), Entailment above the word level in distributional semantics, in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 23–32, Association for Computational Linguistics,
<http://aclweb.org/anthology/E12-1004>.
- Marco BARONI and Roberto ZAMPARELLI (2010), Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space, in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1183–1193, Association for Computational Linguistics, <http://aclweb.org/anthology/D10-1115>.
- Jon BARWISE and Robin COOPER (1981), Generalized Quantifiers and Natural Language, *Linguistics and Philosophy*, 4:159–219.
- Steven BIRD, Ewan KLEIN, and Edward LOPER (2009), *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, Inc.
- Garrett BIRKHOFF and John VON NEUMANN (1936), The Logic of Quantum Mechanics, *Annals of Mathematics*, 37(4):823–843, ISSN 0003486X,
<http://www.jstor.org/stable/1968621>.
- William BLACOE, Elham KASHEFI, and Mirella LAPATA (2013), A Quantum-Theoretic Approach to Distributional Semantics, in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 847–857, Association for Computational Linguistics,
<http://aclweb.org/anthology/N13-1105>.
- Felix BLOCH (1946), Nuclear Induction, *Phys. Rev.*, 70:460–474,
doi:10.1103/PhysRev.70.460,
<https://link.aps.org/doi/10.1103/PhysRev.70.460>.
- Samuel R. BOWMAN, Christopher POTTS, and Christopher D. MANNING (2015), Recursive Neural Networks Can Learn Logical Semantics, in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 12–21, Association for Computational Linguistics,
doi:10.18653/v1/W15-4002, <http://aclweb.org/anthology/W15-4002>.
- Daoud CLARKE (2009), Context-theoretic Semantics for Natural Language: An Overview, in *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics, GEMS '09*, pp. 112–119, Association for Computational Linguistics, Stroudsburg, PA, USA,
<http://dl.acm.org/citation.cfm?id=1705415.1705430>.

Bob COECKE, Edward GREFENSTETTE, and Mehrnoosh SADRZADEH (2013), Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus, *Annals of Pure and Applied Logic*, 164(11):1079 – 1100, ISSN 0168-0072, <https://doi.org/10.1016/j.apal.2013.05.009>, special issue on Seventh Workshop on Games for Logic and Programming Languages (GaLoP VII).

Bob COECKE and Keye MARTIN (2011), A partial order on classical and quantum states, in *New Structures for Physics*, pp. 593–683, Springer.

Bob COECKE and Éric Oliver PAQUETTE (2011), Categories for the practising physicist, in *New Structures for Physics*, pp. 173–286, Springer, https://doi.org/10.1007/978-3-642-12821-9_3.

Bob COECKE, Mehrnoosh SADRZADEH, and Stephen J CLARK (2010), Mathematical Foundations for a Compositional Distributional Model of Meaning, *Linguistic Analysis*, 36(1):345–384.

Ido DAGAN, Oren GLICKMAN, and Bernardo MAGNINI (2006), The PASCAL Recognising Textual Entailment Challenge, in Joaquin QUIÑONERO-CANDELA, Ido DAGAN, Bernardo MAGNINI, and Florence D’ALCHÉ BUC, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, pp. 177–190, Springer, Berlin, Heidelberg, https://doi.org/10.1007/11736790_9.

Ellie D’HONDT and Prakash PANANGADEN (2006), Quantum Weakest Preconditions, *Mathematical Structures in Computer Science*, 16(3):429–451, <https://doi.org/10.1017/S0960129506005251>.

Maayan GEFFET and Ido DAGAN (2005), The Distributional Inclusion Hypotheses and Lexical Entailment, in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pp. 107–114, Association for Computational Linguistics, <http://aclweb.org/anthology/P05-1014>.

Edward GREFENSTETTE, Georgiana DINU, Yi ZHANG, Mehrnoosh SADRZADEH, and Marco BARONI (2013), Multi-Step Regression Learning for Compositional Distributional Semantics, in *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pp. 131–142, Association for Computational Linguistics, <http://aclweb.org/anthology/W13-0112>.

Edward GREFENSTETTE and Mehrnoosh SADRZADEH (2011), Experimental Support for a Categorical Compositional Distributional Model of Meaning, in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1394–1404, Association for Computational Linguistics, <http://aclweb.org/anthology/D11-1129>.

Jules HEDGES and Mehrnoosh SADRZADEH (2016), A Generalised Quantifier Theory of Natural Language in Categorical Compositional Distributional

Semantics with Bialgebras, *CoRR*, abs/1602.01635,
<http://arxiv.org/abs/1602.01635>.

Dimitri KARTSAKLIS (2015), *Compositional Distributional Semantics with Compact Closed Categories and Frobenius Algebras*, Ph.D. thesis, University of Oxford,
<https://arxiv.org/abs/1505.00138>.

Dimitri KARTSAKLIS, Matthew PURVER, and Mehrnoosh SADRZADEH (2016), Verb Phrase Ellipsis using Frobenius Algebras in Categorical Compositional Distributional Semantics, in *DSALT Workshop, European Summer School on Logic, Language and Information*, <https://www.eecs.qmul.ac.uk/~mpurver/papers/kartsaklis-et-al16dsalt.pdf>.

Dimitri KARTSAKLIS, Mehrnoosh SADRZADEH, and Stephen PULMAN (2012), A Unified Sentence Space for Categorical Distributional-Compositional Semantics: Theory and Experiments, in *Proceedings of COLING 2012: Posters*, pp. 549–558, The COLING 2012 Organizing Committee,
<http://aclweb.org/anthology/C12-2054>.

Graham M. KELLY and Miguel L. LAPLAZA (1980), Coherence for compact closed categories, *Journal of Pure and Applied Algebra*, 19:193 – 213, ISSN 0022-4049, [https://doi.org/10.1016/0022-4049\(80\)90101-2](https://doi.org/10.1016/0022-4049(80)90101-2).

Douwe KIELA, Laura RIMELL, Ivan VULIĆ, and Stephen CLARK (2015), Exploiting Image Generality for Lexical Entailment Detection, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 119–124, Association for Computational Linguistics,
[doi:10.3115/v1/P15-2020](https://doi.org/10.3115/v1/P15-2020), <http://aclweb.org/anthology/P15-2020>.

Lili KOTLERMAN, Ido DAGAN, Idan SZPEKTOR, and Maayan ZHITOMIRSKY-GEFFET (2010), Directional distributional similarity for lexical inference, *Natural Language Engineering*, 16(4):359–389,
<https://doi.org/10.1017/S1351324910000124>.

Dexter KOZEN (1983), A Probabilistic PDL, in David S. JOHNSON, Ronald FAGIN, Michael L. FREDMAN, David HAREL, Richard M. KARP, Nancy A. LYNCH, Christos H. PAPADIMITRIOU, Ronald L. RIVEST, Walter L. RUZZO, and Joel I. SEIFERAS, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pp. 291–297, ACM, <https://doi.org/10.1145/800061.808758>.

Joachim LAMBEK (1997), Type Grammar Revisited, in Alain LECOMTE, François LAMARCHE, and Guy PERRIER, editors, *Logical Aspects of Computational Linguistics, Second International Conference, LACL '97, Nancy, France, September 22-24, 1997, Selected Papers*, volume 1582 of *Lecture Notes in Computer Science*, pp. 1–27, Springer, ISBN 3-540-65751-7,
https://doi.org/10.1007/3-540-48975-4_1.

Alessandro LENCI and Giulia BENOTTO (2012), Identifying hypernyms in distributional semantic spaces, in **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pp. 75–79, Association for Computational Linguistics, <http://aclweb.org/anthology/S12-1012>.

Karl LÖWNER (1934), Über monotone Matrixfunktionen, *Mathematische Zeitschrift*, 38(1):177–216.

Bill MACCARTNEY and Christopher D. MANNING (2007), Natural Logic for Textual Inference, in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, RTE '07*, pp. 193–200, Association for Computational Linguistics, Stroudsburg, PA, USA, <http://dl.acm.org/citation.cfm?id=1654536.1654575>.

George A. MILLER (1995), WordNet: A Lexical Database for English, *Communications of the ACM*, 38(11):39–41, ISSN 0001-0782, doi:10.1145/219717.219748, <http://doi.acm.org/10.1145/219717.219748>.

Michael A. NIELSEN and Isaac L. CHUANG (2011), *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, New York, NY, USA, 10th edition, ISBN 1107002176, 9781107002173.

Jeffrey PENNINGTON, Richard SOCHER, and Christopher MANNING (2014), Glove: Global Vectors for Word Representation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Association for Computational Linguistics, doi:10.3115/v1/D14-1162, <http://aclweb.org/anthology/D14-1162>.

Robin PIEDELEU (2014), *Ambiguity in Categorical Models of Meaning*, Master's thesis, University of Oxford, <http://www.cs.ox.ac.uk/people/bob.coecke/Robin.pdf>.

Robin PIEDELEU, Dimitri KARTSAKLIS, Bob COECKE, and Mehrnoosh SADRZADEH (2015), Open System Categorical Quantum Semantics in Natural Language Processing, in Lawrence S. MOSS and Pawel SOBOCIŃSKI, editors, *6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24–26, 2015, Nijmegen, The Netherlands*, volume 35 of *LIPICs*, pp. 270–289, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, ISBN 978-3-939897-84-2, <https://doi.org/10.4230/LIPICs.CALCO.2015.270>.

Anne PRELLER and Mehrnoosh SADRZADEH (2011), Bell States and Negative Sentences in the Distributed Model of Meaning, *Electronic Notes in Theoretical Computer Science*, 270(2):141 – 153, ISSN 1571-0661, <https://doi.org/10.1016/j.entcs.2011.01.028>, proceedings of the 6th International Workshop on Quantum Physics and Logic (QPL 2009).

C. J. van RIJSBERGEN (2004), *The Geometry of Information Retrieval*, Cambridge University Press, New York, NY, USA, ISBN 0521838053.

Laura RIMELL (2014), Distributional Lexical Entailment by Topic Coherence, in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 511–519, Association for Computational Linguistics, doi:10.3115/v1/E14-1054, <http://aclweb.org/anthology/E14-1054>.

Mehrnoosh SADRZADEH, Dimitri KARTSAKLIS, and Esma BALKIR (2018), Sentence entailment in compositional distributional semantics, *Annals of Mathematics and Artificial Intelligence*, 82(4):189–218, <https://doi.org/10.1007/s10472-017-9570-x>.

Peter SELINGER (2007), Dagger Compact Closed Categories and Completely Positive Maps: (Extended Abstract), *Electronic Notes in Theoretical Computer Science*, 170:139 – 163, ISSN 1571-0661, <https://doi.org/10.1016/j.entcs.2006.12.018>, proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005).

Julie WEEDS, David WEIR, and Diana MCCARTHY (2004), Characterising Measures of Lexical Distributional Similarity, in *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, <http://aclweb.org/anthology/C04-1146>.

Hermann WEYL (1912), Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung), *Mathematische Annalen*, 71(4):441–479.

Dominic WIDDOWS and Stanley PETERS (2003), Word vectors and quantum logic: Experiments with negation and disjunction, in *Proceedings of Mathematics of Language 8*, pp. 141–154.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



A proof-theoretic approach to scope ambiguity in compositional vector space models

Gijs Jasper Wijnholds

School of Electronic Engineering and Computer Science,
Queen Mary University of London

ABSTRACT

We investigate the extent to which compositional vector space models can be used to account for scope ambiguity in quantified sentences (of the form *Every man loves some woman*). Such sentences containing two quantifiers introduce two readings, a direct scope reading and an inverse scope reading. This ambiguity has been treated in a vector space model using bialgebras by Hedges and Sadrzadeh (2016) and Sadrzadeh (2016), though without an explanation of the mechanism by which the ambiguity arises. We combine a polarised focussed sequent calculus for the non-associative Lambek calculus **NL**, as described in Moortgat and Moot (2011), with the vector-based approach to quantifier scope ambiguity. In particular, we establish a procedure for obtaining a vector space model for quantifier scope ambiguity in a derivational way.

Keywords: proof theory, scope ambiguity, compositional vector space models, bialgebra

1

INTRODUCTION

There is a long standing tradition in formal semantics on compositionality: in order to separate the meaning of basic elements (lexical semantics) from the construction of higher-level meaning (derivational semantics) one assigns a homomorphism from a *syntactic algebra* to a *semantic algebra*. Having been rigorously formalised by Montague in his seminal papers (Montague 1970, 1973), these ideas have been made concrete in the field of grammar, where syntactic types are

mapped onto semantic types so that any derivation gives rise to a *meaning recipe*. Traditionally, meaning is taken to be a linear lambda term that evaluates to a truth value.

Ongoing research on distributional semantics, based on the idea that word meaning is defined relative to a word's context, has revealed an appealing way to incorporate type-logical grammar into distributional models (Coecke *et al.* 2010). This approach, also known as the DisCoCat approach (Distributional Compositional Categorical models), treats compositionality in the Montagovian style as a functorial passage from syntactic types and proofs to vectors and linear maps. Given that this line of research is still in its early phase, there is much to be done to formalise details of the model, give accounts for semantic phenomena, and evaluate the effectiveness of the chosen approach.

Though traditional categorial syntax and semantics go hand in hand, some aspects of the set-theoretic formal semantics get lost in the switch to a vector space model of meaning. First, the interpretation of constants that one can appeal to in formal semantics are not directly available in a vector-based setting; a logical word like “not” can be computed in the formal setting by taking set complement, but negating a vector or matrix is not trivial.¹ Similarly, for coordinators like “and” and “or” the standard set intersection and union are not available in a vectorial setting. One could replace intersection by vector multiplication and union by vector summation, but in the presence of concrete distributional vectors it is not clear that such operations indeed perform well in an experimental setting. Second, the DisCoCat approach assumes a tight categorial correspondence between a syntactic formalism and the concrete vector semantics: when we want to stay in the realm of finite dimensional vector spaces, we are dealing with a compact closed category; to model a categorial grammar as a category, one needs to fully explicate its proof-theoretic logical and structural rules, an exposition that is not trivially available for any categorial system.² Another issue with this categorial treatment is

¹ Although there is work on simulating negation in a tensor-based setting (Grefenstette 2013), it is not clear what negation really means in a distributional setting. For instance, an alternative view is to treat distributional negation as conversational (Kruszewski *et al.* 2016).

² For instance, the composition and type-raising combinators one finds in Combinatorial Categorical Grammar (Steedman 2000) don't easily translate into

that a simple vector-based model does not have the non-linearity that some models would assume. As an example, allowing a non-linearity in lexical lambda terms or as a syntactic mechanism means the copying of material which is not possible with all vectors. We discuss this issue in more detail in the rest of the paper.

Some of the above issues have been addressed in recent work by Sadrzadeh *et al.* (2013), Hedges and Sadrzadeh (2016), Sadrzadeh (2016), giving accounts of subject/object relativisation, generalised quantifiers, and quantifier scope. In Sadrzadeh *et al.* (2013), the meaning of pronoun relative clauses is explained by using Frobenius algebras in the lexicon, and assigning different pregroup grammar types to the subject relative pronoun “who” and the object relative pronoun “whom”. Two different derivations then naturally arise, giving an intersectional meaning to subject relative clauses of the type “Men who like Mary”, and object relative clauses such as “Men whom Mary likes”. Such an approach does not lend itself to certain Germanic languages where the ambiguity has to be derivational: in Dutch, the subject relative and object relative interpretations above share the surface form “Mannen die Marie mogen”.³ To deal with this issue without specifying lexical alternatives, i.e. different possible typings of the relative pronoun “die”, Moortgat and Wijnholds (2017) provide a derivational account that results in the same intersective vector space meaning as the ones of Sadrzadeh *et al.* (2013).

An element that lacks in the results obtained so far on quantifier scope ambiguity is a detailed discussion of the derivational process, giving rise to ambiguities. Quantifier scope ambiguity as opposed to pronoun relativisation is more pressing as the former exists in English and does not come from the lexicon, but rather from different ways of reading the same surface form. The account of Hedges and Sadrzadeh (2016) explores the use of bialgebras to represent quantifiers, using context free grammars as the syntactic engine; its follow up (Sadrzadeh 2016) discusses scope ambiguity but assumes the ambiguity to be given before detailing the direct scope and inverse scope readings of phrases of the shape “Every man loves some woman”. In

a standard category, and the Displacement Calculus of Morrill *et al.* (2011) subsumes its structural rules in the rules of the system (Valentín 2014).

³“Die” can mean “who”, “whom”, “that”, “mogen” is an inflection of “like”.

order to explain how the ambiguity comes about, we need to detail the syntactic process, and integrate it with a vector-based semantics.

Our goal in this paper, then, is to pave the way to fully explain compositionality in vector space models of meaning while also taking into account the desirable mechanisms of e.g. Frobenius algebras and bialgebras. The contribution of this paper is to show how we can represent quantifier scope ambiguity in a derivational manner, fully determined by the syntactic process combined with a suitable lexical semantics.

We will make use of a polarised non-associative Lambek calculus, and use focussing as a technique to gain control over the space of sequent derivations. A *continuation-passing-style* translation from syntactic types into semantic objects then gives rise to the expected reading for quantifier scope ambiguity. This technique has been worked out by Moortgat and Moot (2011) (following Bernardi and Moortgat 2010 and Bastenhof 2012), but has not, until now, been put in the context of vector space models.

This paper is structured as follows: in Section 2, we briefly discuss quantifier scope ambiguity and its apparent non-linearity. Next, in Section 3 we define the basic, compositional DisCoCat model. We proceed to review quantifier scope ambiguity in vector space models in Section 4, and show in Section 5 how we can derive quantifier scope ambiguity in a compositional way using a polarised focussed sequent calculus that is interpreted in a vector space model. We conclude in Section 6 by explaining how our results can be further expanded and we introduce some potential new areas of investigation.

2 QUANTIFIER SCOPE AMBIGUITY

There seems to be an intrinsic non-linearity associated with quantifiers. Consider the word “all” in a phrase “all men sleep”. One way of modelling the universal quantification in the phrase is to let “all” refer to an operation that decides whether the set of “men” is a subset of those entities that are sleeping, i.e. if “men” refers to some set A , and “sleep” to some set B , then “all men sleep” computes whether $A \subseteq B$. This can be given an alternative definition:

$$[all](A)(B) = \begin{cases} 1 & \text{if } A = A \cap B \\ 0 & \text{otherwise} \end{cases}$$

When one tries to give this interpretation in terms of a λ -term, the usual approach is to model both “men” and “sleep” as a *characteristic function* of a set of entities, where “all” is given a non-linear λ -term:

$$\llbracket all \rrbracket = \lambda P. \lambda Q. (\forall (\lambda x. (P\ x) \rightarrow (Q\ x)))$$

This λ -term will effectively decide whether $A \subseteq B$, or alternatively whether $A = A \cap B$. Both the modellings sacrifice linearity in a sense: where the first, relational interpretation needs to use A as an operand to the intersection operation and as an argument to decide equality, the second interpretation has to *copy* the variable x to decide whether everything in the universe satisfying the property P also satisfies Q . We argue that this required non-linearity that is introduced by allowing non-linear λ -terms to be inserted through the lexicon, is exactly the same kind of non-linearity that is introduced to vector space models by means of bialgebra operations. It has been argued before that modelling quantification in vector space models forces one to use non-linear maps (Grefenstette 2013). However, this issue has been partially resolved by Hedges and Sadrzadeh (2016) by admitting a powerset structure to the basis vectors of the model. The then obtained bialgebra operations are linear in the algebraic sense, but non-linear in terms of typing information. That is, they allow for copying a resource X into a resource $X \otimes X$ and deleting a resource in the opposite direction. That this kind of operation would jeopardize a Lambek-style grammar formalism is immediate as the bialgebra operations would correspond to contraction and expansion, respectively. Our argument will proceed by claiming that a continuation-passing-style translation that allows for lexical insertion of non-linear λ -terms can instead be interpreted by means of the bialgebra operations of Hedges and Sadrzadeh (2016).

3 COMPOSITIONAL DISTRIBUTIONAL SEMANTICS

Compositional distributional semantics in a categorical setting takes a mathematically rigorous approach to compositionality. Much like traditional Montagovian semantics, there is a syntactic algebra involved that provides grammaticality by means of a proof system, in this case it can be either a *pregroup grammar* or the *Lambek calculus* (Lambek 1958, 1997). The *semantic algebra* is, in the basic setup, the category of

finite dimensional vector spaces, denoted **FVect**: content words are assigned a vector that represents its position in the space of word meanings, obtained through some method of co-occurrence extraction on a corpus. Whenever a sequence of words, annotated with their *syntactic types*, leads to a derivation that proves grammaticality, the proof term associated with that derivation provides a linear map on the vectors associated with basic words which, after evaluation, gives us the *phrase meaning* of that sequence of words.

3.1 Lambek grammars

We make the model sketched above concrete by giving the relevant definitions. These are based on work by Wijnholds (2014) in combination with the work of Coecke *et al.* (2013).

Definition 1 (Lambek types). *Given a set T of basic types, the set of Lambek types $F(T)$ is the smallest set such that:*

1. *If $p \in T$ then $p \in F(T)$,*
2. *If $A, B \in F(T)$ then $A \otimes B, A \backslash B, B / A \in F(T)$.*

We proceed to define a Lambek calculus in terms of a labelled deductive system, i.e. we use the notation of an inference system to show how proofs are derived:

Definition 2 (Non-associative Lambek calculus). *The (non-associative, non-unitary) Lambek calculus **NL** over T is given by the types in $F(T)$ and the proofs generated by the following (labelled) inference system:*

$$\begin{array}{c}
 \frac{}{1_A : A \rightarrow A} Ax \qquad \frac{f : A \rightarrow B \quad g : B \rightarrow C}{g \circ f : A \rightarrow C} T \\
 \\
 \frac{f : A \otimes B \rightarrow C}{\triangleright f : A \rightarrow C/B} R1 \qquad \frac{f : A \otimes B \rightarrow C}{\triangleleft f : B \rightarrow A \backslash C} R2 \\
 \\
 \frac{g : A \rightarrow C/B}{\triangleright^{-1} g : A \otimes B \rightarrow C} R1^{-1} \qquad \frac{g : B \rightarrow A \backslash C}{\triangleleft^{-1} g : A \otimes B \rightarrow C} R2^{-1}
 \end{array}$$

One can show that monotonicity laws for each of the connectives are derived rules of inference:

$$\frac{f : A \rightarrow C \quad g : B \rightarrow D}{f \otimes g : A \otimes B \rightarrow C \otimes D} M_{\otimes}$$

$$\frac{f : A \rightarrow C \quad g : B \rightarrow D}{g/f : B/C \rightarrow D/A} M_/_$$

$$\frac{f : A \rightarrow C \quad g : B \rightarrow D}{f \setminus g : C \setminus B \rightarrow A \setminus D} M_\setminus$$

where we have:

$$f \otimes g := \triangleright^{-1} ((\triangleright \triangleleft^{-1} ((\triangleleft 1_{C \otimes D}) \circ g)) \circ f)$$

$$g/f := \triangleright (g \circ (\triangleleft^{-1} ((\triangleleft \triangleright^{-1} 1_{B \setminus C}) \circ f)))$$

$$f \setminus g := \triangleleft (g \circ (\triangleright^{-1} ((\triangleright \triangleleft^{-1} 1_{C \setminus B}) \circ f)))$$

Leaving aside the issue of global associativity and its desirability from a linguistic perspective, we note how it can be added using two additional axioms:

$$\overline{a_{A,B,C} : (A \otimes B) \otimes C \rightarrow A \otimes (B \otimes C)} \text{ Ass}$$

$$\overline{a_{A,B,C}^{-1} : A \otimes (B \otimes C) \rightarrow (A \otimes B) \otimes C} \text{ Ass}^{-1}$$

The *categorical* version of the Lambek calculus can be obtained by imposing the relevant standard equivalences on proofs, amongst others stipulating that composing with the identity proof is a vacuous operation, and that all two-way inference rules are isomorphisms. For more detail we refer the reader to Wijnholds (2014).

In order to make grammaticality judgments to sequences of words, we need a *lexicon* assigning types to words over an alphabet. For the sake of completeness we define the lexicon as a relation, but in the remainder of this paper we will freely abuse notation and treat the lexicon as if it were a function.

Definition 3 (Lexicon). *Let Σ be a finite, non-empty set of words (an alphabet). A lexicon over Σ is a relation $\delta \subseteq \Sigma \times F(T)$.*

Definition 4 (Lambek grammar). *Given a set of basic types T , a Lambek grammar over T is a triple (Σ, δ, S) where Σ is an alphabet, δ is a lexicon over T , and $S \in F(T)$ is a distinguished goal type.*

Definition 5 (Grammaticality). *Given a Lambek grammar (Σ, δ, S) over T , we say that a sequence of words $w_1 \dots w_n$ over Σ is grammatical iff there is a merged sequence $W_1 \otimes W_2 \dots \otimes W_n$ (where for each i we have $w_i \delta W_i$), and there exists a proof of $W_1 \otimes W_2 \dots \otimes W_n \rightarrow S$ in the Lambek calculus.*

The presented definitions so far give a procedure to obtain a proof of sentencehood for a sequence of words. Moreover, there might be several proofs of the same sequence of words. This may be desirable (in cases of derivational ambiguity) or not (in the case proof-theoretic redundancy, e.g. the successive to and fro use of two-way rules). In the categorical variant of the Lambek calculus, we can simply take the proofs of sentencehood of a sequence to be the hom-set of morphisms $\text{Hom}(W_1 \otimes W_2 \dots \otimes W_n, S)$. This produces fewer proofs as unnecessary ambiguity of the proof system is brought down by categorical equations. The structure of the (non-associative) Lambek calculus **NL** is that of a *biclosed magmatic category*.⁴

3.2 Finite dimensional vector space models

Lambek grammars are easily interpretable in vector space semantics, as vector spaces enjoy compact closure – a weaker variant of the bi-closure of the Lambek calculus. We define the category **FVect** and show that it enjoys compact closure:

Definition 6 (Compact Closure). *A compact closed category is a monoidal category \mathcal{C} with dual objects A^l, A^r for every object A in \mathcal{C} and additional morphisms:*

$$\begin{aligned} A^l \otimes A &\xrightarrow{\epsilon_A^l} I \xrightarrow{\eta_A^l} A \otimes A^l \\ A \otimes A^r &\xrightarrow{\epsilon_A^r} I \xrightarrow{\eta_A^r} A^r \otimes A \end{aligned}$$

*that satisfy the yanking properties:*⁵

$$\begin{aligned} (id_A \otimes \epsilon_A^l) \circ (\eta_A^l \otimes id_A) &= id_A & (\epsilon_A^r \otimes id_A) \circ (id_A \otimes \eta_A^r) &= id_A \\ (\eta_A^l \otimes id_{A^l}) \circ (id_{A^l} \otimes \eta_A^l) &= id_{A^l} & (id_{A^r} \otimes \eta_A^r) \circ (\eta_A^r \otimes id_{A^r}) &= id_{A^r} \end{aligned}$$

In the category of finite dimensional vector spaces **FVect** we have that the dual space A^* is isomorphic to A when we fix a basis (which is the case for concrete models). The ϵ and η maps, now reduced to just two maps, are given by:

$$\begin{aligned} \epsilon &:= \sum_{ij} c_{ij} (v_i \otimes v_j) \mapsto \sum_{ij} c_{ij} \langle v_i \mid v_j \rangle \\ \eta &:= 1 \mapsto \sum_i (v_i \otimes v_i) \end{aligned}$$

⁴ A magmatic category is a weaker version of a monoidal category: the tensor has no unit and is not necessarily associative. See Wijnholds (2017).

⁵ Note that we left out the hidden associativity morphism.

In concrete vector models, we will have vectors learnt for content words. For instance, the noun phrases “John” and “Mary” can be interpreted as vectors $\vec{n}_1, \vec{n}_3 \in \mathbf{N}$, respectively. This means that they are essentially single points in a vector space. Setting the sentence space to be the real numbers, a transitive verb such as “loves” would live in the vector space $\mathbf{N} \otimes \mathbb{R} \otimes \mathbf{N}$, and would carry information about the degree with which individuals love one another. In vector terms:

$$\sum_{ij} c_{ij} (\vec{n}_i \otimes 1 \otimes \vec{n}_j)$$

The c_{ij} is the respective degree for any pair of individuals i, j . The meaning of the phrase “John loves Mary” should then reduce by taking the inner product of the noun phrases with the verbs to give:

$$\sum_{ij} c_{ij} \langle \vec{n}_1 | \vec{n}_i \rangle \langle \vec{n}_j | \vec{n}_3 \rangle$$

In the next section, we show how to relate derivations in a Lambek grammar to concrete computations in a vector space model.

3.3 Interpretation

Given that the compact closedness of **FVect** instantiates the closure of the Lambek calculus, we can easily interpret proofs in a Lambek grammar in a vector space model by passing from words and their lexical types to vectors in a homomorphically obtained vector space. Any proof of grammaticality will be interpreted through the η and ϵ maps:

Definition 7 (Interpretation). *Let (Σ, δ, S) be a Lambek grammar over T . An interpretation is a pair of maps $I_0 : F(T) \rightarrow \mathbf{FVect}$, $I_1 : \Sigma \rightarrow \delta(\Sigma)$, where $\delta(\Sigma)$ is the relational image of δ , such that I_0 respects typing and I_1 respects lexical type assignment. That is,*

$$I_0(A \otimes B) = I_0(A \setminus B) = I_0(A/B) = I_0(A) \otimes I_0(B)$$

and

$$I_1(w) = \vec{v} \quad \text{iff} \quad w \delta W \text{ and } \vec{v} \in I_0(W)$$

An interpretation map sends words to vectors that respect the syntactic types associated with those words. We need to give a vectorial interpretation of proofs as well, in order to know how to compute meanings

of a tuple of vectors. The identity proof and transitivity of proofs carries over to the identity map on vector spaces and the composition of linear maps. The remaining rules of residuation are interpreted as shown below:

$$\frac{f' : I_0(A) \otimes I_0(B) \rightarrow I_0(C)}{(f' \otimes id_{I_0(B)}) \circ (id_{I_0(A)} \otimes \eta_{I_0(B)}) : I_0(A) \rightarrow I_0(C) \otimes I_0(B)} R1$$

$$\frac{f' : I_0(A) \otimes I_0(B) \rightarrow I_0(C)}{(id_{I_0(A)} \otimes f') \circ (\eta_{I_0(A)} \otimes id_{I_0(B)}) : I_0(B) \rightarrow I_0(A) \otimes I_0(C)} R2$$

$$\frac{g' : I_0(A) \rightarrow I_0(C) \otimes I_0(B)}{(id_{I_0(C)} \otimes \epsilon_{I_0(B)}) \circ (g' \otimes id_{I_0(B)}) : I_0(A) \otimes I_0(B) \rightarrow I_0(C)} R1^{-1}$$

$$\frac{g' : I_0(B) \rightarrow I_0(A) \otimes I_0(C)}{(\epsilon_{I_0(A)} \otimes id_{I_0(C)}) \circ (id_{I_0(A)} \otimes g') : I_0(A) \otimes I_0(B) \rightarrow I_0(C)} R2^{-1}$$

It is a nice puzzle for the reader to verify that by the yanking equations, we preserve isomorphicity of residuation, for example one can show that the interpretation of $\triangleright^{-1} \triangleright f$ is equal to the interpretation of f .

3.4

Illustration

Recall that we have vectors for “John”, “Mary” and “loves” and we have an intended meaning of the phrase “John loves Mary”. We take a Lambek grammar over the set of basic types $\{np, s\}$, where np will be interpreted as \mathbb{N} and s will be mapped to \mathbb{R} . We define a lexicon as follows:

w	$\delta(w)$	$I_1(w)$	$I_0(\delta(w))$
“John”	np	n_1	\mathbf{N}
“Mary”	np	n_3	\mathbf{N}
“loves”	$(np \backslash s)/np$	$\sum_{ij} c_{ij}(\vec{n}_i \otimes 1 \otimes \vec{n}_j)$	$\mathbf{N} \otimes \mathbb{R} \otimes \mathbf{N}$

Given that $\triangleleft^{-1} \triangleright^{-1} (1_{(np \backslash s)/np})$ proves the grammaticality of “John loves Mary”, the associated meaning computation will be:

$$\begin{aligned}
 & (\epsilon_N \otimes id_{\mathbb{R}}) \circ (id_N \otimes ((id_{N \otimes \mathbb{R}} \otimes \epsilon_N) \circ (id_{N \otimes \mathbb{R} \otimes N} \otimes id_N))) \\
 & (\vec{n}_1 \otimes \sum_{ij} c_{ij}(\vec{n}_i \otimes 1 \otimes \vec{n}_j) \otimes \vec{n}_3) \\
 & = (\epsilon_N \otimes id_{\mathbb{R}}) \circ (id_{N \otimes N \otimes \mathbb{R}} \otimes \epsilon_N)(\vec{n}_1 \otimes \sum_{ij} c_{ij}(\vec{n}_i \otimes 1 \otimes \vec{n}_j) \otimes \vec{n}_3) \\
 & = (\epsilon_N \otimes id_{\mathbb{R}})(\vec{n}_1 \otimes \sum_{ij} c_{ij}(\vec{n}_i \otimes \langle \vec{n}_j \mid \vec{n}_3 \rangle)) \\
 & = \sum_{ij} c_{ij} \langle \vec{n}_1 \mid \vec{n}_i \rangle \langle \vec{n}_j \mid \vec{n}_3 \rangle \\
 & = c_{13}
 \end{aligned}$$

This result is exactly the intended meaning we wanted to obtain. Note that the result of the computation relies on the fact that the content words in the vector space model are taken to be the basis vectors, hence they are orthogonal. The result c_{13} indicates the distributional strength of John loving Mary in a corpus that the vectors have been learnt from. Until now, we have neglected discussion about function words: logical words, relative pronouns, and quantifiers are not intuitively represented well by co-occurrence data. The logical word “and” may occur with many different words, but that statistic does not tell us much about the meaning of the word. So, although all the basic operations from a Lambek grammar are directly interpretable in vector space models, more advanced semantic phenomena lack an explanation in the simple models.

4 QUANTIFIER SCOPE AMBIGUITY IN VECTOR SPACE MODELS

In this section we review the use of bialgebras in vector space models as exhibited by Hedges and Sadrzadeh (2016) and Sadrzadeh (2016) and show how the two scope readings can be obtained. The treatment of quantifiers in vector space models relies on the use of powersets: as long as we can know of our vector space that its basis vectors are given by the powerset of some set A , we can perform additional operations on the vector space.

Definition 8 (Bialgebra). *Given a symmetric monoidal category $(\mathcal{C}, \otimes, I, \sigma)$, a bialgebra on an object X in \mathcal{C} is a tuple of maps:*

$$\begin{aligned}
 X & \xrightarrow{\delta_X} X \otimes X \xrightarrow{\mu_X} X \\
 X & \xrightarrow{\iota_X} I \xrightarrow{\zeta_X} X
 \end{aligned}$$

that satisfy the conditions of a monoid for (X, μ, ζ) and a comonoid for (X, δ, ι) and furthermore satisfy the bialgebra axioms:

$$\begin{aligned}\iota \circ \mu &= \iota \otimes \iota \\ \delta \circ \zeta &= \zeta \otimes \zeta \\ \iota \circ \zeta &= id_I \\ \delta \circ \mu &= (\mu \otimes \mu) \circ (id_X \otimes \sigma \otimes id_X) \circ (\delta \otimes \delta)\end{aligned}$$

The last of the four equations tells us that in a bialgebra, the order of copying and merging is irrelevant given that we can switch copies by means of the symmetry of the category. What is interesting to note is that any powerset $P(U)$ bears a bialgebra structure if we consider the Cartesian product to be the tensor and the singleton set $\{\star\}$ as the identity object. What follows is that any vector space over a powerset, denoted $V_{P(U)}$, carries a bialgebra structure. Both bialgebras are given below:

$$\begin{array}{ccc} A & \xleftrightarrow{\delta} & A \times A \\ A \times B & \xleftrightarrow{\mu} & A \cap B \\ A & \xleftrightarrow{\iota} & \{\star\} \\ \{\star\} & \xleftrightarrow{\zeta} & U \end{array} \qquad \begin{array}{ccc} |A\rangle & \xrightarrow{\delta} & |A\rangle \otimes |A\rangle \\ |A\rangle \otimes |B\rangle & \xrightarrow{\mu} & |A \cap B\rangle \\ |A\rangle & \xrightarrow{\iota} & 1 \\ 1 & \xrightarrow{\zeta} & |U\rangle \end{array}$$

The existence of a bialgebra on powerset vector spaces allows for a neat treatment of quantification. Given that nouns and noun phrases are represented as vectors on a powerset, universal quantification and existential quantification are treated as:

$$\begin{aligned} |A\rangle & \xrightarrow{\llbracket all \rrbracket} \sum_{A \subseteq B \subseteq U} |B\rangle \\ |A\rangle & \xrightarrow{\llbracket some \rrbracket} \sum_{\substack{B \text{ s.t.} \\ A \cap B \neq \emptyset}} |B\rangle \end{aligned}$$

To get a feel for how the meaning of a quantified sentence should be computed according to Hedges and Sadrzadeh (2016), we show the example of “all men sleep”, which gets assigned the meaning:

$$\begin{aligned} & \epsilon_{V_{P(U)}} \circ (\llbracket all \rrbracket \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}})(| \llbracket men \rrbracket \rangle \otimes | \llbracket sleep \rrbracket \rangle) \\ &= \epsilon_{V_{P(U)}} \circ (\llbracket all \rrbracket \otimes \mu_{V_{P(U)}})(| \llbracket men \rrbracket \rangle \otimes | \llbracket men \rrbracket \rangle \otimes | \llbracket sleep \rrbracket \rangle) \\ &= \epsilon_{V_{P(U)}} \left(\sum_{\llbracket men \rrbracket \subseteq B \subseteq U} |B\rangle \otimes | \llbracket men \rrbracket \cap \llbracket sleep \rrbracket \rangle \right) \\ &= \sum_{\llbracket men \rrbracket \subseteq B \subseteq U} \langle B | \llbracket men \rrbracket \cap \llbracket sleep \rrbracket \rangle \\ &= \langle \llbracket men \rrbracket | \llbracket men \rrbracket \cap \llbracket sleep \rrbracket \rangle \end{aligned}$$

Although this approach works for statements with a single quantifier, it fails to deliver both readings for a doubly quantified statement such as “every student likes some teacher” as the computations for the subject and object quantifiers will be independent of each other. Hence, both readings will collapse to the same meaning. This lack of explanatory power of the model is amended in a subsequent paper (Sadrzadeh 2016), where the implicit quantified variable is passed on to the computation of the second quantifier. A transitive verb such as “likes” is modelled as an element:

$$\llbracket \text{likes} \rrbracket = \sum_{ij} c_{ij}(|A_i\rangle \otimes |A_j\rangle)$$

in $V_{P(U)} \otimes V_{P(U)}$, and we can model the forward image of an element in U as:

$$\llbracket \text{likes}_a \rrbracket = \sum_{ij} w_{ij}(\{a\} | A_i\rangle | A_j\rangle)$$

The backward image is computed similarly by taking the inner product of \vec{v}_a with \vec{v}_j . This construction now allows for both readings of “every student likes some teacher”, though there is no procedure given to obtain these readings through a syntactic process.

5 QUANTIFIER SCOPE AMBIGUITY USING FOCUSSING AND POLARISATION

Focussing is a proof-theoretic technique stemming from the work of (Andreoli 2001) that aims to eliminate redundancy from regular sequent systems. Focussed proof search proceeds by distinguishing those formulas that enjoy invertible introduction rules (*asynchronous formulas*), and those that do not (*synchronous formulas*). Asynchronous formulas are decomposed in a backward chaining proof search until there is no more decomposition possible. Then, one of the synchronous formulas is selected to be put in focus, after which the process of decomposition continues. This implies that now only the number of synchronous formulas determines the number of distinct proofs. This approach has been applied to the Lambek-Grishin calculus, a symmetric extension of the Lambek calculus, by Bernardi and Moortgat (2010), and is worked out in more detail by Moortgat and Moot (2011).

In order to obtain a compositional Montagovian semantics from a display style presentation of focussed proofs for the Lambek-Grishin calculus, Bastenhof (2012) applies a polarisation technique, whereby formulas are assigned either positive or negative polarity. Atomic formulas are assigned an arbitrary polarity; the choice of this *bias* affects the set of proofs obtained. The polarity also influences semantics: under the continuation semantics of Bernardi and Moortgat (2010), a negative formula will be *negated* in its interpretation. Though the focussing and polarisation approaches are described by Bernardi and Moortgat (2010) and Bastenhof (2012), respectively, here we follow the focussed sequent presentation of Moortgat and Moot (2011).

We start by defining polarity of types:

Definition 9 (Polarity). *Given a set of basic types T , a polarity assignment on types is a map $pol : F(T) \rightarrow \{-, +\}$ that assigns to the types in T an arbitrary polarity but fixes the polarity for complex types:*

$$\begin{aligned} pol(A \otimes B) &= + \\ pol(A \setminus B) &= - \\ pol(B / A) &= - \end{aligned}$$

Given a Lambek grammar G over a set T , grammaticality is defined similarly to Definition 5, where the set of proofs is given by the underlying proof system. The only difference is that the final sequent should have the consequent formula in focus. Any proof is encoded by its abstract label, according to the abstract sequent system defined in Figure 1.

5.1

CPS translation

The translation of types and proofs given by Moortgat and Moot (2011) into a target semantic algebra is a two-step process:

$$\begin{array}{ccccc} \text{source} & \xrightarrow{I} & \text{continuation} & \xrightarrow{J} & \text{target} \\ \mathbf{NL}_{\otimes, \setminus, /} & & \text{semantics} & & \mathbf{FVect} \\ & & \mathbf{LP}_{\otimes, \perp} & & \end{array}$$

Instead of considering a proof to be a simple transformation of values (the assumptions) to a value (the conclusion), we consider a proof to be a *continuation*, a function that awaits an evaluation context to compute a final value. The intermediate semantics is the Lambek calculus

Focused types are positive

$$\frac{}{Ax(A, x) \mid x : A \Rightarrow \boxed{A}}^{Ax}$$

$$\frac{M \mid X[x : A] \Rightarrow Y}{\leftarrow (M, x, A) \mid X[\boxed{A}] \Rightarrow Y} \leftarrow$$

$$\frac{M \mid X \Rightarrow \boxed{A}}{\rightarrow (M, \alpha) \mid X \Rightarrow \alpha : A} \rightarrow$$

$$\frac{M \mid X[\boxed{A}] \Rightarrow Z \quad N \mid Y \Rightarrow \boxed{B}}{/L(M, N) \mid X[\boxed{A/B}] \bullet Y \Rightarrow Z} /L$$

$$\frac{M \mid X[x : A \bullet y : B] \Rightarrow Y}{\otimes L(M, x, y, z) \mid X[z : A \otimes B] \Rightarrow Y} \otimes L$$

$$\frac{M \mid Y \Rightarrow \boxed{B} \quad N \mid X[\boxed{A}] \Rightarrow Z}{\backslash L(M, N) \mid X[Y \bullet \boxed{B/A}] \Rightarrow Z} \backslash L$$

Focused types are negative

$$\frac{}{CoAx(A, \alpha) \mid \boxed{A} \Rightarrow \alpha : A}^{CoAx}$$

$$\frac{M \mid X[\boxed{A}] \Rightarrow Y}{\leftarrow (M, x) \mid X[x : A] \Rightarrow Y} \leftarrow$$

$$\frac{M \mid X \Rightarrow \alpha : A}{\rightarrow (M, \alpha) \mid X \Rightarrow \boxed{A}} \rightarrow$$

$$\frac{M \mid X \bullet x : B \Rightarrow \alpha : A}{/R(M, x, \alpha, \beta) \mid X \Rightarrow \beta : A/B} /R$$

$$\frac{M \mid X \Rightarrow \boxed{A} \quad N \mid Y \Rightarrow \boxed{B}}{\otimes R(M, N) \mid X \bullet Y \Rightarrow \boxed{A \otimes B}} \otimes R$$

$$\frac{M \mid x : B \bullet X \Rightarrow \alpha : A}{\backslash R(M, x, \alpha, \beta) \mid X \Rightarrow \beta : B \backslash A} \backslash R$$

 Figure 1:
Focussed
labelled sequent
system for **NL**

with permutation and negation, $\mathbf{LP}_{\otimes, \perp}$, a system that only uses a product operation but introduces a negation. Furthermore, permutation of resources is allowed to compensate for the lack of directionality without the $/, \backslash$ connectives. We will define a direct mapping from source to target, to skip the administrative details of the intermediate semantics.

In order to replicate the effect of the negation in $\mathbf{LP}_{\otimes, \perp}$, we use vector spaces over sets; given some type A , we define its interpretation to be a vector space over a set. In this way, we enjoy the bialgebras defined over those vector spaces. First, a type W is mapped to some set A , using the Cartesian product and powerset operations. Then, the final interpretation of a type will be the vector space over the given set, V_A . We get the intended tensor products on spaces due to the fact that $V_{A \times B} \cong V_A \otimes V_B$.

Definition 10 (Type interpretation). *Given a set of basic types T and a basic interpretation map $I_0 : T \rightarrow \mathbf{Set}$, the type interpretation is a map $I_1 : F(T) \rightarrow \mathbf{Set}$ defined as follows:*

1. *For basic types $p \in T$ we have:*

$$I_1(p) = \begin{cases} I_0(p) & \text{if } \text{pol}(p) = + \\ P(I_0(p)) & \text{if } \text{pol}(p) = - \end{cases}$$

2. *For complex types, the interpretation depends both on the polarity of subtypes and the connective involved:*

$A \ B$	$I_1(A \otimes B)$	$I_1(A \setminus B)$	$I_1(B / A)$
$- \ -$	$P(I_1(A)) \times P(I_1(B))$	$P(I_1(A)) \times I_1(B)$	$I_1(B) \times P(I_1(A))$
$- \ +$	$P(I_1(A)) \times I_1(B)$	$P(I_1(A)) \times P(I_1(B))$	$I_1(B) \times I_1(A)$
$+ \ -$	$I_1(A) \times P(I_1(B))$	$I_1(A) \times I_1(B)$	$P(I_1(B)) \times P(I_1(A))$
$+ \ +$	$I_1(A) \times I_1(B)$	$I_1(A) \times P(I_1(B))$	$P(I_1(B)) \times I_1(A)$

3. *We stipulate that for any type A , its interpretation $I_1(A)$ is lifted to the vector space spanned by its elements, that is we define the final interpretation $I_2 : F(T) \rightarrow \mathbf{FVect}$ as $I_2(W) = V_{I_1(W)}$.*

Definition 11 (Word interpretation). *Given a Lambek grammar (Σ, δ, S) over a set of basic types T and an interpretation map $I_2 : F(T) \rightarrow I_2(\delta(\Sigma))$, where $\delta(\Sigma)$ is the relational image of Σ under the lexicon, and $I_2(\delta(\Sigma))$ is the image under interpretation (i.e. vector spaces), the word interpretation is a map I_3 that respects the following:*

$$\begin{aligned} I_3(w) \in I_2(W) & \quad \text{iff} \quad w \delta W \text{ and } \text{pol}(W) = + \\ I_3(w) \in I_2(W) \rightarrow \mathbb{R} & \quad \text{iff} \quad w \delta W \text{ and } \text{pol}(W) = - \end{aligned}$$

That is, words with a positive type are translated as vectors, while words with a negative type are translated as linear maps.

As an example, if we define the associated vector space of the type np to be U and n to be $P(U)$, then the interpretation of a noun like “student” will be a constant $I_3(\text{“student”}) \in V_{P(U)}$, whereas a word like “all” that is typed np/n will be a linear map:

$$I_3(\text{“all”}) \in V_{P(U)} \otimes V_{P(U)} \rightarrow \mathbb{R}$$

We proceed to define how we interpret proof terms. The intuitive idea is that a proof term is translated into a linear map which will subsequently be applied to the word interpretations of its antecedents.

Though the proof system builds up terms with potentially unbound variables, we require for grammaticality (see above) that the conclusion formula be in focus; this means that the only unbound variables in the proof term are those of the antecedent formula, which will be substituted by word interpretations.

Definition 12 (Proof term interpretation). *Given a proof in the focussed sequent calculus for \mathbf{NL} , there is a proof term that encodes the proof. We define the interpretation of a proof by giving the translation of proof terms into linear maps:*

$$\begin{array}{ll}
 Ax(A, x) & \xRightarrow{I_4} x \in I_3(A) \\
 CoAx(A, \alpha) & \xRightarrow{I_4} \alpha \in I_3(A) \\
 \multimap (M, x, A) & \xRightarrow{I_4} |\{x \in I_3(A) | I_4(M) \neq 0\}| \\
 \multimap (M, x) & \xRightarrow{I_4} x(I_4(M)) \\
 \rightarrow (M, \alpha) & \xRightarrow{I_4} \alpha(I_4(M)) \\
 \rightarrow (M, \alpha) & \xRightarrow{I_4} \alpha \mapsto I_4(M) \\
 /L(M, N) & \xRightarrow{I_4} I_4(M) \otimes I_4(N) \\
 /R(M, x, \alpha, \beta) & \xRightarrow{I_4} I_4(M)[\beta \rightarrow \alpha \otimes x] \\
 \otimes L(M, x, y, z) & \xRightarrow{I_4} I_4(M)[z \rightarrow x \otimes y] \\
 \otimes R(M, N) & \xRightarrow{I_4} I_4(M) \otimes I_4(N) \\
 \backslash L(M, N) & \xRightarrow{I_4} I_4(M) \otimes I_4(N) \\
 \backslash R(M, x, \alpha, \beta) & \xRightarrow{I_4} I_4(M)[\beta \rightarrow x \otimes \alpha]
 \end{array}$$

Finally, as the interpretation is a continuation-passing-style translation, we will end up with a map that needs an evaluation context before finishing computation. So, given that a proof gives a linear map, we apply it to the identity map, and we instantiate the unbound variables with the relevant *word interpretations*.

5.2 Deriving quantifier scope ambiguity

Quantifier scope ambiguity as exemplified by the phrase “Every student likes some teacher”, is already shown to be obtainable using the two-step translation process of Bernardi and Moortgat (2010) in a Lambek-Grishin grammar, and in a Lambek grammar (Moortgat and Moot 2011). Here, we alter the latter example given to translate into

the vector space model as employed by Hedges and Sadrzadeh (2016) and Sadrzadeh (2016) to show that both readings (narrow/wide and wide/narrow) can be obtained and give exactly the kind of meaning we would expect from a vector space model. This means that we can obtain the intended meaning in a *derivational* way. What is more, given that we have both a grammar available and we have learned concrete vectors, the process can potentially be fully automated. Each word has to be associated with a syntactic type, and we have to give a word interpretation mapping the words to a vector or linear map. We assume a set of basic types $\{np, n, s\}$ where s is the distinguished goal type. Polarity assignment is handled by stipulating that np and n are positive, and s is negative. Basic types np and n are interpreted as U and $P(U)$, respectively, and s gets translated to \mathbb{R} . The syntactic types and the word interpretation are given by the following table:

w	$\delta(w)$	$[w]$
every	np/n	$\epsilon_{V_{P(U)}} \circ (\llbracket \text{all} \rrbracket \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \circ \sigma$
student	n	$\llbracket \text{student} \rrbracket$
likes	$(np \setminus s)/np$	$\vec{a} \otimes f \otimes \vec{b} \mapsto f(\llbracket (\text{likes}_b)_a \rrbracket)$
some	np/n	$\epsilon_{V_{P(U)}} \circ (\llbracket \text{some} \rrbracket \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \circ \sigma$
teacher	n	$\llbracket \text{teacher} \rrbracket$

As a reminder, we also note the vectorial interpretation of lexical constants in the word interpretation:

$$\begin{aligned}
 \llbracket \text{all} \rrbracket(|A\rangle) &= \sum_{A \subseteq B \subseteq U} |B\rangle \\
 \llbracket \text{some} \rrbracket(|A\rangle) &= \sum_{\substack{B \subseteq U \text{ s.t.} \\ A \cap B \neq \emptyset}} |B\rangle \\
 \llbracket \text{student} \rrbracket &= |A\rangle \text{ for some } A \subseteq U \\
 \llbracket \text{teacher} \rrbracket &= |B\rangle \text{ for some } B \subseteq U \\
 \llbracket \text{likes} \rrbracket &= \sum_{ij} c_{ij}(|A_i\rangle \otimes 1 \otimes |A_j\rangle) \text{ for each } A_x \subseteq U
 \end{aligned}$$

The two proofs that we get from the focussed sequent calculus are displayed in Figures 2 and 3 (without labelling).

$$\begin{array}{c}
 \frac{a : np \Rightarrow \boxed{np} \quad Ax \quad \boxed{s} \Rightarrow \alpha : s \quad CoAx}{a : np \bullet \boxed{np \backslash s} \Rightarrow \alpha : s} \backslash L \quad \frac{b : np \Rightarrow \boxed{np} \quad Ax}{b : np \bullet \boxed{np \backslash s} \Rightarrow \alpha : s} /L \\
 \frac{a : np \bullet ((np \backslash s) / np) \bullet b : np \Rightarrow \alpha : s}{a : np \bullet (z : (np \backslash s) / np \bullet b : np) \Rightarrow \alpha : s} \leftarrow \\
 \frac{a : np \bullet (z : (np \backslash s) / np \bullet \boxed{np}) \Rightarrow \alpha : s}{a : np \bullet (z : (np \backslash s) / np \bullet \boxed{np / n} \bullet w : n) \Rightarrow \alpha : s} \leftarrow \quad \frac{\text{teacher} \quad Ax}{w : n \Rightarrow \boxed{n}} /L \\
 \frac{a : np \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s}{a : np \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s} \leftarrow \quad \frac{\text{student} \quad Ax}{y : n \Rightarrow \boxed{n}} /L \\
 \frac{(\boxed{np / n} \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s}{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s} \leftarrow \\
 \frac{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s}{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \boxed{s}} \rightarrow \\
 \text{every} \quad \text{student} \quad \text{likes} \quad \text{some} \quad \text{teacher}
 \end{array}$$

Figure 2: Proof for wide over narrow scope

$$\begin{array}{c}
 \frac{b : np \Rightarrow \boxed{np} \quad Ax \quad \boxed{s} \Rightarrow \alpha : s \quad CoAx}{b : np \bullet \boxed{np \backslash s} \Rightarrow \alpha : s} \backslash L \quad \frac{a : np \Rightarrow \boxed{np} \quad Ax}{a : np \bullet \boxed{np \backslash s} \Rightarrow \alpha : s} /L \\
 \frac{b : np \bullet ((np \backslash s) / np) \bullet a : np \Rightarrow \alpha : s}{b : np \bullet (z : (np \backslash s) / np \bullet a : np) \Rightarrow \alpha : s} \leftarrow \\
 \frac{b : np \bullet (z : (np \backslash s) / np \bullet a : np) \Rightarrow \alpha : s}{\boxed{np} \bullet (z : (np \backslash s) / np \bullet a : np) \Rightarrow \alpha : s} \leftarrow \quad \frac{\text{student} \quad Ax}{y : n \Rightarrow \boxed{n}} /L \\
 \frac{(\boxed{np / n} \bullet y : n) \bullet (z : (np \backslash s) / np \bullet a : np) \Rightarrow \alpha : s}{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet a : np) \Rightarrow \alpha : s} \leftarrow \\
 \frac{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet \boxed{np}) \Rightarrow \alpha : s}{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet \boxed{np / n} \bullet w : n) \Rightarrow \alpha : s} \leftarrow \quad \frac{\text{teacher} \quad Ax}{w : n \Rightarrow \boxed{n}} /L \\
 \frac{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s}{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s} \leftarrow \\
 \frac{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \alpha : s}{(x : np / n \bullet y : n) \bullet (z : (np \backslash s) / np \bullet (u : np / n \bullet w : n)) \Rightarrow \boxed{s}} \rightarrow \\
 \text{every} \quad \text{student} \quad \text{likes} \quad \text{some} \quad \text{teacher}
 \end{array}$$

Figure 3: A proof for narrow over wide scope

If we take the proof term for the first proof and translate this into a vectorial map we get:

$$(1a) \alpha \mapsto x (\{a \in U \mid u (\{b \mid z (a \otimes \alpha \otimes b) \neq 0\} \otimes w) \neq 0\} \otimes y)$$

For the second proof term, we get a slightly different map:

$$(2a) \alpha \mapsto u (\{a \in U \mid x (\{b \mid z (b \otimes \alpha \otimes a) \neq 0\} \otimes y) \neq 0\} \otimes w)$$

The unfolded maps are quite intimidating so the complete computation is taken up in the Appendix. Here we just note that the two maps reduce to the readings shown below:

$$(1b) \langle \llbracket \text{student} \rrbracket \mid \llbracket \text{student} \rrbracket \cap \{a \in U \mid \sum_{\substack{B \subseteq U \text{ s.t.} \\ \llbracket \text{teacher} \rrbracket \cap B \neq \emptyset}} \langle B \mid \llbracket \text{teacher} \rrbracket \cap C \rangle \rangle \rangle$$

$$\text{where } C = \{b \in U \mid \llbracket (\text{likes}_b)_a \rrbracket \neq 0\}$$

$$(2b) \sum_{\substack{B \subseteq U \text{ s.t.} \\ \llbracket \text{teacher} \rrbracket \cap B \neq \emptyset}} \langle B \mid \llbracket \text{teacher} \rrbracket \cap \{a \in U \mid \langle \llbracket \text{student} \rrbracket \mid \llbracket \text{student} \rrbracket \cap D \rangle \neq 0\} \rangle$$

$$\text{where } D = \{b \in U \mid \llbracket (\text{likes}_a)_b \rrbracket \neq 0\}$$

We can see that these interpretations will give different results depending on the instantiation of the vectors. In fact, these interpretations correspond to the result of Sadrzadeh (2016). This effectively shows that quantifier scope ambiguity can be achieved in vector space models by the use of appropriate proof-theoretic notions.

6

CONCLUDING REMARKS

In this paper, we elaborated on quantifier scope ambiguity in compositional distributional models of meaning. In particular, the approach of Moortgat and Moot (2011) using a continuation-passing-style translation for a polarised and focussed proof system for the Lambek calculus was combined with the approach to generalised quantifiers of Hedges and Sadrzadeh (2016). The result is fully derivational and provides a fully worked out compositional way to obtain two readings for phrases of the type “Every student likes some teacher”, thereby resolving the issue of manually assigning appropriate meaning vectors to such phrases.

Although we illustrate this with examples of two generalised quantifiers in a sentence, the approach works for a single quantifier, and since the applied strategy exploits the combinatorial choices of the proof system (focus on the first quantifier and then on the second one, or vice versa) we expect the approach to generalise to more quantifiers, though the possibility of overgeneration needs to be investigated.

As for experimental validation, since the writing of this paper, it has been recognised that using a powerset construction in vector spaces, to be able to make use of bialgebras, may not be very feasible in practical models: having a powerset as a basis may lead to an exponential blowup in vector space size, and could potentially give sparsity issues. One approach to deal with this could be to use fuzzy quantification (Zadeh 1983), which has already been explored by Dostal and Sadrzadeh (2016).

Another interesting avenue is to work out how several phenomena involving the copying of linguistic material can be analysed in a compositional distributional model. Coordination and pronoun relativisation have been given an account using Frobenius algebras over vector spaces (Kartsaklis 2016; Sadrzadeh *et al.* 2013), where the Frobenius operations allow one to express elementwise multiplication on arbitrary tensors. In future work we hope to analyse ellipsis, a phenomenon for which it can be argued that copying has to be part of the syntactic process. Rules of controlled copying, then, can be interpreted using the Frobenius or bialgebra operations. A first step has already been taken by Kartsaklis *et al.* (2016), and we wish to approach the problem from the type-logical perspective.

7

ACKNOWLEDGMENTS

I am grateful for a range of insightful discussions with Michael Moortgat on the focussing for the Lambek calculus, and various perspectives on (non-)linearity. Furthermore, I would like to thank Mehrnoosh Sadrzadeh and Dimitri Kartsaklis for the many short and long discussions on Frobenius algebras and bialgebras. Finally, I am grateful for technical comments from Paulo Oliva, and the anonymous reviewers of JLM. I was supported by a Queen Mary Principal Studentship during the writing of this paper. All remaining errors are my own.

APPENDIX

$$(1a) \alpha \mapsto x \ (\{a \in U \mid u \ (\{b \in U \mid z \ (a \otimes \alpha \otimes b) \neq 0\} \otimes w) \neq 0\} \otimes y)$$

Which, after lexical insertion gives

$$\alpha \mapsto [\text{every}] \ (\{a \in U \mid [\text{some}] \ (\{b \in U \mid [\text{likes}] \ (a \otimes \alpha \otimes b) \neq 0\} \otimes [\text{teacher}] \neq 0\} \otimes [\text{student}])$$

Unfolding the definition and inserting the identity map gives

$$\begin{aligned} & \epsilon_{V_{P(U)}} \circ ([\text{all}] \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \\ & \quad \circ \sigma \left(\{a \in U \mid \epsilon_{V_{P(U)}} \circ ([\text{some}] \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \circ \sigma \left(\{b \in U \mid [[\text{likes}_b]_a] \neq 0\} \otimes [[\text{teacher}]] \neq 0\} \right) \otimes [[\text{student}]] \right) \\ &= \epsilon_{V_{P(U)}} \circ ([\text{all}] \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \circ \sigma \left(\{a \in U \mid \sum_{\substack{B \subseteq U \text{ s.t.} \\ [[\text{teacher}]] \cap B \neq \emptyset}} \langle B \mid [[\text{teacher}]] \cap \{b \in U \mid [[\text{likes}_b]_a] \neq 0\} \rangle \otimes [[\text{student}]] \right) \\ &= \sum_{[[\text{student}]] \subseteq C \subseteq U} \langle C \mid [[\text{student}]] \cap \{a \in U \mid \sum_{\substack{B \subseteq U \text{ s.t.} \\ [[\text{teacher}]] \cap B \neq \emptyset}} \langle B \mid [[\text{teacher}]] \cap \{b \in U \mid [[\text{likes}_b]_a] \neq 0\} \rangle \rangle \rangle \end{aligned}$$

$$(2a) \alpha \mapsto u \mid (\{a \in U \mid x \mid (\{b \in U \mid z \mid (b \otimes \alpha \otimes a) \neq 0\} \otimes y) \neq 0\} \otimes w)$$

Which, after lexical insertion gives

$$\alpha \mapsto [\text{some}] \mid (\{a \in U \mid [\text{every}] \mid (\{b \in U \mid [\text{likes}] \mid (b \otimes \alpha \otimes a) \neq 0\} \otimes [\text{student}] \neq 0\} \otimes [\text{teacher}])$$

Unfolding the definition and inserting the identity map gives

$$\begin{aligned} & \epsilon_{V_{P(U)}} \circ ([\text{some}] \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \\ & \circ \sigma \left(\left\{ a \in U \mid \epsilon_{V_{P(U)}} \circ ([\text{all}] \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \circ \sigma \left(\{b \in U \mid [[(\text{likes})_b] \neq 0\} \otimes [\text{student}]] \neq 0 \right\} \right\} \otimes [\text{teacher}] \right) \\ & = \epsilon_{V_{P(U)}} \circ ([\text{some}] \otimes \mu_{V_{P(U)}}) \circ (\delta_{V_{P(U)}} \otimes id_{V_{P(U)}}) \circ \sigma \left(\left\{ a \in U \mid \sum_{[\text{student}] \subseteq B \subseteq U} \langle B \mid [\text{student}] \cap \{b \in U \mid [[(\text{likes})_b] \neq 0\} \neq 0 \rangle \right\} \otimes [\text{teacher}] \right) \\ & = \sum_{\substack{C \subseteq U \text{ s.t.} \\ [[\text{teacher}]] \cap C \neq \emptyset}} \langle C \mid [\text{teacher}] \rangle \cap \{a \in U \mid \sum_{[\text{student}] \subseteq B \subseteq U} \langle B \mid [\text{student}] \cap \{b \in U \mid [[(\text{likes})_b] \neq 0\} \neq 0 \rangle \neq 0 \rangle \end{aligned}$$

REFERENCES

- Jean-Marc ANDREOLI (2001), Focussing and proof construction, *Annals of Pure and Applied Logic*, 107(1):131–163,
doi:[https://doi.org/10.1016/S0168-0072\(00\)00032-4](https://doi.org/10.1016/S0168-0072(00)00032-4).
- Arno BASTENHOF (2012), Polarized Montagovian semantics for the Lambek-Grishin calculus, in Philippe DE GROOTE and Mark-Jan NEDERHOF, editors, *15th and 16th International Conference on Formal Grammar*, volume 7395, pp. 1–16, Springer, Springer-Verlag Berlin Heidelberg,
doi:<http://dx.doi.org/10.1007/978-3-642-32024-8>.
- Raffaella BERNARDI and Michael MOORTGAT (2010), Continuation semantics for the Lambek–Grishin calculus, *Information and Computation*, 208(5):397–416,
doi:<https://doi.org/10.1016/j.ic.2009.11.005>.
- Bob COECKE, Edward GREFENSTETTE, and Mehrnoosh SADRZADEH (2013), Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus, *Annals of Pure and Applied Logic*, 164(11):1079–1100,
doi:<https://doi.org/10.1016/j.apal.2013.05.009>.
- Bob COECKE, Mehrnoosh SADRZADEH, and Stephen CLARK (2010), Mathematical foundations for a compositional distributional model of meaning, *arXiv preprint arXiv:1003.4394*, <https://arxiv.org/pdf/1003.4394>.
- Matej DOSTAL and Mehrnoosh SADRZADEH (2016), Many valued generalised quantifiers for natural language in the DisCoCat model, Technical report, Czech Technical University Prague and Queen Mary University of London,
<https://qmul.ac.uk/xmlui/bitstream/handle/123456789/17382/DisCoCat%20Maodel%20Paper%20M.Sadrzadeh.pdf>.
- Edward GREFENSTETTE (2013), Towards a formal distributional semantics: simulating logical calculi with tensors, in *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pp. 1–10, Association for Computational Linguistics, <http://aclweb.org/anthology/S13-1001>.
- Jules HEDGES and Mehrnoosh SADRZADEH (2016), A generalised quantifier theory of natural language in categorical compositional distributional semantics with bialgebras, *arXiv preprint arXiv:1602.01635*,
<https://arxiv.org/pdf/1602.01635>.
- Dimitri KARTSAKLIS (2016), Coordination in categorical compositional distributional semantics, *arXiv preprint arXiv:1606.01515*,
<https://arxiv.org/pdf/1606.01515>.
- Dimitri KARTSAKLIS, Matthew PURVER, and Mehrnoosh SADRZADEH (2016), Verb phrase ellipsis using Frobenius algebras in categorical compositional distributional semantics, *DSALT Workshop, European Summer School on Logic, Language and Information*, <https://pdfs.semanticscholar.org/6c56/137ffb008ee5f94a482e0c74e494d7f7bc04.pdf>.

- Germán KRUSZEWSKI, Denis PAPERNO, Raffaella BERNARDI, and Marco BARONI (2016), There is no logical negation here, but there are alternatives: Modeling conversational negation with distributional semantics, *Computational Linguistics*, 42(4):637–660, doi:https://doi.org/10.1162/COLI_a_00262.
- Joachim LAMBEK (1958), The mathematics of sentence structure, *The American Mathematical Monthly*, 65(3):154–170, doi:<https://doi.org/10.1080/00029890.1958.11989160>.
- Joachim LAMBEK (1997), Type grammar revisited, in *International Conference on Logical Aspects of Computational Linguistics*, pp. 1–27, Springer, doi:https://doi.org/10.1007/3-540-48975-4_1.
- Richard MONTAGUE (1970), English as a formal language, *Linguaggi nella Società e nella Tecnica*.
- Richard MONTAGUE (1973), The proper treatment of quantification in ordinary English, in *Approaches to Natural Language*, pp. 221–242, Springer, doi:https://doi.org/10.1007/978-94-010-2506-5_10.
- Michael MOORTGAT and Richard MOOT (2011), Proof nets for the Lambek-Grishin calculus, *arXiv preprint arXiv:1112.6384*, <https://arxiv.org/pdf/1112.6384>.
- Michael MOORTGAT and Gijs WIJNHOLDS (2017), Lexical and derivational meaning in vector-based models of relativisation, *Proceedings of the 21st Amsterdam Colloquium*, pp. 55–64, <https://semanticsarchive.net/Archive/jZiM2FhZ/AC2017-Proceedings.pdf>.
- Glyn MORRILL, Oriol VALENTÍN, and Mario FADDA (2011), The displacement calculus, *Journal of Logic, Language and Information*, 20(1):1–48, doi:<https://doi.org/10.1007/s10849-010-9129-2>.
- Mehrnoosh SADRZADEH (2016), Quantifier scope in categorical compositional distributional semantics, *arXiv preprint arXiv:1608.01404*, <https://arxiv.org/pdf/1608.01404>.
- Mehrnoosh SADRZADEH, Stephen CLARK, and Bob COECKE (2013), The Frobenius anatomy of word meanings I: subject and object relative pronouns, *Journal of Logic and Computation*, 23(6):1293–1317, doi:<https://doi.org/10.1093/logcom/ext044>.
- Mark STEEDMAN (2000), *The Syntactic Process*, MIT Press.
- Oriol VALENTÍN (2014), The hidden structural rules of the discontinuous Lambek calculus, in *Categories and Types in Logic, Language, and Physics*, pp. 402–420, Springer, doi:https://doi.org/10.1007/978-3-642-54789-8_23.
- Gijs WIJNHOLDS (2014), *Categorical foundations for extended compositional distributional models of meaning*, Master’s thesis, Universiteit van Amsterdam, <https://www.illc.uva.nl/Research/Publications/Reports/reportlist/MoL-2014-22.text.pdf>.

Gijs Jasper WIJNHOLDS (2017), Coherent diagrammatic reasoning in compositional distributional semantics, in *International Workshop on Logic, Language, Information, and Computation*, pp. 371–386, Springer, doi:https://doi.org/10.1007/978-3-662-55386-2_27.

Lotfi A. ZADEH (1983), A computational approach to fuzzy quantifiers in natural languages, *Computers & Mathematics with Applications*, 9(1):149–184, ISSN 0898-1221, doi:[http://dx.doi.org/10.1016/0898-1221\(83\)90013-5](http://dx.doi.org/10.1016/0898-1221(83)90013-5).

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



Combining logical and distributional methods in type-logical grammars

Richard Moot

LIRMM, Montpellier University, CNRS

ABSTRACT

We propose a low-level way of combining distributional and logical ideas into a single formal system. This will be an instantiation of a more general system, adding weights to proof rules. These weights will not measure some sort of “confidence the proof is valid”, but rather act as a way to *prefer* some proofs over others, where preference can mean “easier to process (for humans)” or “more coherent (combining words that make sense together)”. The resulting system of weighted theorem proving can be implemented either as a best-first proof search strategy or as a polynomial-time approximation of proof search for NP-complete parsing problems.

Keywords:
type-logical
grammar,
Lambek calculus,
theorem proving

1

INTRODUCTION

Type-logical grammars (and formal semantics in general) are agnostic about the meaning of atomic terms, such as those corresponding to nouns and verbs (though not about the meaning corresponding to words with logical content such as “not”, “and”, “all”, “which”). Another way to see this is that in standard formal semantics, entailment only holds under strict identity of predicates. As a consequence, practical use of the output of a system computing such formal semantics depends to a large extent on the available world knowledge (Bos and Markert 2005), possibly stated in the form of additional axioms or meaning postulates, stating that “pub” and “bar” (in one meaning of the word) are synonyms, and that “good” and “bad” are antonyms, i.e. “bad” entails “not good” and inversely.

In contrast to formal semantics in the tradition of Montague, distributional or vector-based semantics take semantic similarity, as measured by word cooccurrences, as their basic notion. Systems using only semantic similarity are agnostic about argument structure and agnostic about the meaning of words with logical content. Given a vector of a sequence of words, it is not a priori clear how to combine these into the meaning of a phrase. In other words, vector space models are not compositional by nature, although many ways of computing vector space semantics for texts exist, and even the simplest models (adding or averaging all vectors for the words in a larger text) can perform well on a number of tasks (see Mitchell and Lapata 2010; Pham 2016, for discussion).

Whereas compositional formal semantics, unless augmented by specific lexical meanings or meaning postulates, concludes that “good” and “bad” are unrelated unary predicates, vector semantics concludes that “good” and “bad” are very similar. Other semantically similar words are “animal” and “veterinarian”, and “sweater” and “warm”. The absence of argument structure (who does what to whom) from the vectors makes “animal” and “veterinarian” similar: even though many sentences contain both words, these tend to be sentences where the veterinarian treats or examines the animal, or where the owner takes an animal to the veterinarian.

There appears to be some complementarity to these two approaches: formal semantics takes compositionality as its basic principle but has little to say about the meaning of individual predicates; vector semantics takes the similarity of predicates as its basic concept but then has little to say about compositionality and the words with logical content.

In this paper, we will look at a low-level way of combining distributional and logical ideas into a single formal system. This will be an instantiation of a more general system, adding weights to proof rules. These weights will not measure some sort of “confidence the proof is valid”, but rather act as a way to *prefer* some proofs over others, where preference can mean “easier to process (for humans)” or “more coherent (combining words that make sense together)”. The resulting system of weighted theorem proving can be implemented either as a best-first proof search strategy or as a polynomial-time approximation of proof search for NP-complete parsing problems.

TYPE-LOGICAL GRAMMAR AND FORMAL SEMANTICS

Definition 2.1 (Type-logical grammar) *Given a logic \mathcal{L} with formulas \mathcal{F} , a type-logical grammar over \mathcal{L} is a tuple $\langle \Sigma, \text{Lex}, \text{goal}, \text{yield}, h \rangle$, where*

1. Σ is a set of words (the vocabulary of the language),
2. the lexicon Lex , is a function from $w \in \Sigma$ to a (non-empty) subset of \mathcal{F} ,
3. goal , the set of goal formulas is a (non-empty) subset of \mathcal{F} ,
4. yield is a function from antecedents of \mathcal{L} to sequences of formulas,
5. h is a homomorphism from proofs in \mathcal{L} to proofs in multiplicative intuitionistic linear logic representing their “deep structure”.

Informally, a sentence is grammatical whenever the lexicon assigns each word in the sentence a formula, and these formulas produce a derivable statement in the logic. More formally, we say a sentence w_1, \dots, w_n is *grammatical* if for all i , $w_i \in \Sigma$ (each word is in the vocabulary) and there is an $A_i \in \text{Lex}(w_i)$ (we choose, for each word, one of the formulas assigned to it by the lexicon), there is a structure Γ with $\text{yield}(\Gamma) = A_1, \dots, A_n$, and there is a $C \in \text{goal}$ such that the statement $\Gamma \vdash C$ is a theorem of the logic \mathcal{L} . A sentence is *ungrammatical* otherwise.

Many authors choose the set $\{s\}$ for *goal* (that is, the only valid goal category is s , for *sentence*). However, for more elaborate grammars, we may be interested not only in declarative sentences, but also in yes-no questions, *wh* questions, imperatives, etc., and it seems reasonable to allow such sentences to have a different type of meaning from declarative sentences.

For the Lambek calculus (Lambek 1958), the logic is \mathbf{L} , the yield function is the identity function (since antecedents Γ of \mathbf{L} are already sequences of formulas), and h translates the Lambek calculus slashes “/” and “\” to the multiplicative linear logic implication “ \multimap ” (and the product “ \bullet ” to multiplicative conjunction “ \otimes ”).

For multimodal type-logical grammars (Moortgat 1997), sequents are of the form $\Gamma \vdash C$ where the antecedent Γ is a labelled tree with unary and binary branches and with formulas as its leaves. The yield

Figure 1:
Proof rules and
corresponding
lambda term
operations

$$\begin{array}{c}
 \frac{A/B : M^{U \rightarrow T} \quad B : N^U}{A : (MN)^T} /E \qquad \frac{B : N^U \quad B \backslash A : M^{U \rightarrow T}}{A : (MN)^T} \backslash E \\
 \\
 \frac{\dots [B : x^U]_i \quad \vdots \quad A : M^T}{A/B : (\lambda x.M)^{U \rightarrow T}} /I_i \qquad \frac{[B : x^U]_i \dots \quad \vdots \quad A : M^T}{B \backslash A : (\lambda x.M)^{U \rightarrow T}} \backslash I_i
 \end{array}$$

function is simply the left-to-right sequence of formulas occurring as its leaves (i.e. we use the standard definition of the yield of a tree).

2.1

The Lambek calculus

To make this more concrete, we'll instantiate the general type-logical grammar framework to Lambek's Syntactic Calculus, L (Lambek 1958). Formulas of the Lambek calculus are inductively defined from a set of atomic formulas, including *np* (noun phrase), *n* (common noun), *s* (sentence) and *pp* (prepositional phrase). A formula in the Lambek calculus is:

- an atomic formula,
- if *A* and *B* are formulas, then *A/B* (pronounced “*A* over *B*”, it looks for a *B* formula to its right to produce an *A*), *B \ A* (pronounced “*B* under *A*”, it looks for a *B* formula to its left to produce an *A*) are formulas.¹

Figure 1 shows the natural deduction proof rules for the Lambek calculus (and the associated lambda term assignments).

The elimination rule for “/”, labeled “/E” states that if we have a proof with conclusion *A/B* which is assigned term *M* (of type $U \rightarrow T$) and a proof with conclusion *B* which is assigned term *N* (of type *U*), then we can combine these two proofs to form a proof of *A* which is assigned lambda-term (*M N*). The order of the premisses is important: *B* must occur adjacent to and to the right of *A/B*. The elimination rule for \backslash is left-right symmetric, with *B* occurring to the immediate left of *B \ A*.

¹ To keep the discussion simple, we do not present the natural deduction proof rules for the product $A \bullet B$, representing the concatenation of *A* and *B*.

The introduction rule, labeled “/I”, states that if we have a proof of A with lambda-term M of some type T , which we have derived while using a hypothesis B , which is assigned a variable x of type U and which is the rightmost undischarged hypotheses of this proof, then we can discharge this B hypothesis to derive A/B of type $U \rightarrow T$ with term $\lambda x.M$. The discharged hypothesis is co-indexed with the rule, using an index i unique to the proof (for the Lambek calculus without product, this index is strictly speaking superfluous, since the leftmost and rightmost undischarged hypotheses are uniquely determined for each subproof). The introduction rule for $\backslash I$ is again left-right symmetric, requiring B to be the leftmost undischarged hypothesis.

We will write $A_1, \dots, A_n \vdash C$ for a proof with undischarged hypotheses A_1, \dots, A_n (in the given order) and conclusion C .

As an example, the following Lambek calculus proof shows that “moons which Galileo discovered” is a noun n . To make the proof more readable, the lexical entries have been indicated as the conclusions of a rule *Lex* with the word occurring above it and the corresponding formula assigned by the lexicon below it (we will add the lambda terms later).

$$\begin{array}{c}
 \text{moons} \quad \text{which} \quad \text{Galileo} \quad \text{discovered} \\
 \frac{n}{\text{moons}} \text{Lex} \quad \frac{(n \backslash n) / (s / np)}{\text{which}} \text{Lex} \quad \frac{np}{\text{Galileo}} \text{Lex} \quad \frac{(np \backslash s) / np \quad [np]_1}{\text{discovered}} \text{Lex} \\
 \frac{n \quad n \backslash n}{n} \backslash E \quad \frac{s \quad s / np}{s / np} / I_1 \quad \frac{np \backslash s}{np \backslash s} \backslash E \quad \frac{[np]_1}{[np]_1} / E
 \end{array}$$

We can read off the lexical assignments from the undischarged leaves of the proof above. So “discovered” is a transitive verb, looking for a noun phrase (np , its object) to its right, then for a noun phrase (its subject) to its left to form a sentence s . The relativiser “which” looks for a complex formula s / np (that is a sentence missing a noun phrase in its rightmost position) to its right and for a noun to its left. The hypothetical np corresponds to a trace in mainstream syntactic theory. A weakness of the Lambek calculus is that this analysis does not extend to only slightly more complicated examples such as “moons which Galileo discovered in 1610”, where the hypothetical noun phrase no

longer occurs in a peripheral position. Many variants and extensions of the Lambek calculus have been developed with the goal of solving this and other problems (see, for example Moortgat 1997; Morrill 2011).

Given the lexicon, the phrase “moons which Galileo discovered” is a noun n iff the following holds.

$$n, (n \backslash n) / (s / np), np, (np \backslash s) / np \vdash n$$

The proof above shows this statement holds. Even though it is easy to verify this proof is correct by inspecting each rule application, it may not be immediately obvious how to find natural deduction proofs. In the next section, we will present a proof search procedure for the implicational fragment of Lambek calculus natural deduction.

2.2 Proof search in natural deduction

For our proof search procedure, the notion of *result* is useful.

Definition 2.2 *Given a formula F , its result is the atomic subformula of F defined as follows.*

$$\begin{aligned} \text{result}(A) &= A && \text{if } A \text{ atomic} \\ \text{result}(A/B) &= \text{result}(A) \\ \text{result}(B \backslash A) &= \text{result}(A) \end{aligned}$$

Essentially, the result is the atomic formula we obtain once we have combined a formula with all its arguments. So the result of $(np \backslash s) / np$ is s and the result of $(n \backslash n) / (s / np)$ is n . Proof search for a sequent $A_1, \dots, A_n \vdash C$ in natural deduction works as follows (a more precise description can be found in Moot and Retoré 2012).

1. If C is a complex formula, apply the appropriate introduction rules until we obtain an atomic formula p (this may add formulas to the left of A_1 and to the right of A_n).
2. Select an active hypothesis H of the proof such that $\text{result}(H) = p$ (that is, select a formula which eventually produces the current atomic goal formula).
3. Our current sequent is of form $A_1, \dots, A_{i-1}, H, A_{i+1}, \dots, A_n \vdash p$ and we need to subdivide the formulas to the left of H (A_1, \dots, A_{i-1}) into m subsequences $\Gamma_1, \dots, \Gamma_m$, where m is the number of arguments H takes to its left, and we need to subdivide the formulas to the right of H (A_{i+1}, \dots, A_n) into k subsequences $\Delta_1, \dots, \Delta_k$

where k is the number of arguments H takes to its right (this fails if H selects no arguments to its left and A_1, \dots, A_{i-1} is not empty, and similarly if H has no arguments to its right and A_{i+1}, \dots, A_n is not empty). We apply all elimination rules to H until we arrive at atomic formula p , then recursively find the proofs from step 1 for each of the arguments: proofs $\Gamma_q \vdash B_q$ for arguments to the left and proofs $\Delta_r \vdash D_r$ for arguments to the right. In the simplest case with a single argument on the right and a single argument on the left, $H = (B \setminus p) / D$; there is no need for splitting the A_i further and we simply try to find proofs for $A_1, \dots, A_{i-1} \vdash B$ and for $A_{i+1}, \dots, A_n \vdash D$. Succeed if all recursive steps succeed. If not, try other subdivisions of the hypotheses. Fail when there is no way to divide the hypotheses such that all subproofs succeed.

The algorithm above has non-determinism in two places. The first step is deterministic, but in the second step there may be several choices for the atomic goal formula and in the third step there may be several ways to split up the sequence of formulas (we need multiple arguments either to the right or to the left for this).

As an example, the sequent $n, (n \setminus n) / (s / np), np, (np \setminus s) / np \vdash n$ has an atomic conclusion, so nothing needs to be done for the first step. For the second step, there is the choice of two formulas: either n (corresponding to “moons”) or $(n \setminus n) / (s / np)$ (corresponding to “which”). The first choice fails immediately since there are still formulas to the right which are not arguments of the formula producing the result (as it is atomic). The second choice provides a formula looking for an s / np to its right and an n to its left. Simply writing out the required elimination rules and separating the hypotheses produces the following.

$$\begin{array}{c}
 \begin{array}{c} \text{moons} \\ \hline n \\ \vdots \\ \delta_1 \\ \hline n \end{array} \text{Lex} \quad \begin{array}{c} \text{which} \\ \hline (n \setminus n) / (s / np) \\ \hline n \end{array} \text{Lex} \quad \begin{array}{c} \text{Galileo} \\ \hline np \\ \hline \end{array} \text{Lex} \quad \begin{array}{c} \text{discovered} \\ \hline (np \setminus s) / np \\ \hline \end{array} \text{Lex} \\
 \begin{array}{c} \vdots \\ \delta_2 \\ \hline s / np \end{array} /E \\
 \hline n \setminus n \setminus E \\
 \hline n
 \end{array}$$

We now need to complete the procedure recursively to find the proofs δ_1 and δ_2 . The first subproof is trivial: we are looking for a noun and there is one, so δ_1 is empty and the n premiss and the n conclusion

of δ_1 become the same formula occurrence. The second subproof has a complex goal, so according to step 1 we apply the introduction rule for “/” which produces the following.

$$\frac{\frac{\text{moons}}{n} \text{ Lex} \quad \frac{\text{which}}{(n \backslash n)/(s/np)} \text{ Lex} \quad \frac{\frac{\text{Galileo}}{np} \text{ Lex} \quad \frac{\text{discovered}}{(np \backslash s)/np} \text{ Lex} \quad [np]_1}{\frac{s}{s/np} /I_1} /E}{\frac{n \backslash n}{n} \backslash E} \backslash E$$

Our subproof δ_3 requires us to prove $np, (np \backslash s)/np, np \vdash s$. Since s is atomic and only the transitive verb has s as its goal, this produces the following.

$$\frac{\frac{\text{moons}}{n} \text{ Lex} \quad \frac{\text{which}}{(n \backslash n)/(s/np)} \text{ Lex} \quad \frac{\frac{\text{Galileo}}{np} \text{ Lex} \quad \frac{\text{discovered}}{(np \backslash s)/np} \text{ Lex} \quad [np]_1}{\frac{s}{s/np} /I_1} /E}{\frac{n \backslash n}{n} \backslash E} \backslash E$$

We can complete the proof by identifying the atomic noun phrases in δ_4 and in δ_5 . The given proof procedure is top-down and enumerates eta-long beta-normal form proofs. In addition, different proofs correspond to different meanings, that is, different proofs will have different lambda terms assigned to them using the term assignment of Figure 1. This correspondence between natural deduction proofs and lambda-terms is the well-known Curry-Howard correspondence. It is not an isomorphism for the Lambek calculus, since not all intuitionistic proofs have a corresponding Lambek calculus proof. Even stronger, not all multiplicative intuitionistic linear logic proofs have a corresponding Lambek calculus proof: the Lambek calculus is a logic without contraction and weakening, like linear logic, but also without the exchange rule.

Adding the term assignment of Figure 1 produces the following proof.

$$\frac{\text{moons}}{n : m} \text{Lex} \frac{\text{which}}{(n \backslash n) / (s / np) : w} \text{Lex} \frac{\text{Galileo}}{np : g} \text{Lex} \frac{\text{discovered}}{(np \backslash s) / np : d} \text{Lex} \frac{[np : x]_1}{np \backslash s : (d x)} /E \backslash E /E$$

$$\frac{n \backslash n : (w(\lambda x.((d x) g)))}{n : ((w(\lambda x.((d x) g))) m)} \backslash E$$

Each lexical assumption of the proof is assigned a unique variable (in the example above, this variable is the first letter of the corresponding word for the convenience of the reader) and these have exactly one free occurrence in the final term (these are linear lambda terms, since each abstraction binds exactly one variable as well). The lexicon assigns both a formula and a corresponding lambda term to each lexical entry. Computing the formal semantics corresponds to replacing each word by its lexical semantics. In the current case, ignoring complications such as tense and the plural, many words have a trivial meaning assignment, so we replace m by the constant $moon^{e \rightarrow t}$ (or, if we prefer, by its eta expansion $\lambda x^e.moon(x)$), d by the constant $discover^{e \rightarrow (e \rightarrow t)}$, g by the constant $Galileo^e$. The crucial case is the lexical assignment for “which”, for which we replace w by $\lambda Q^{e \rightarrow t} \lambda P^{e \rightarrow t} \lambda y^e.(P y) \wedge (Q y)$. Making all lexical substitutions in our original term $((w(\lambda x.((d x) g))) m)$ produces the following term.

$$((\lambda Q \lambda P \lambda y.((P y) \wedge (Q y)))(\lambda x.((discover x) Galileo))) moon$$

We apply beta-reduction by substituting $\lambda x.((discover x) Galileo)$ for Q , which produces the following.

$$(\lambda P \lambda y.((P y) \wedge ((\lambda x.((discover x) Galileo)) y))) moon$$

A second beta-reduction replaces x by y as follows.

$$(\lambda P \lambda y.((P y) \wedge ((discover y) Galileo))) moon$$

The final beta-reduction replaces P by $moon$ to produce the following normal form.

$$\lambda y.((moon y) \wedge ((discover y) Galileo))$$

According to the standard notational conventions of Montague semantics (Gamut 1991), this corresponds to the following more natural term.

$$\lambda y.(moon(y) \wedge discover(Galileo, y))$$

That is, the y such that they are moons and were discovered by Galileo.

2.3

Proof nets

Type-logical grammars generally have multiple proof systems which are provably equivalent (in the sense that they derive the same theorems). Having multiple proof systems available is a great benefit, because meta-theoretical properties are often easier to prove in one system than in another.

Even though natural deduction is a nice proof system producing proofs which are fairly easy to read and which have a direct connection to the semantics, we will introduce a second proof system, *proof nets*, which makes some aspects of proof combinatorics easier to see².

Proof nets are a proof system introduced for linear logic by Girard (1987). Proof nets represent proofs as (hyper)graphs, where the vertices are (polarized) formulas and the hyperlinks represent a connection between the main formula of a rule and its immediate subformulas. The links for the Lambek calculus are shown in Table 1. The formulas above a link are its premisses whereas the formulas below it are its conclusions. The axiom link, on the top left of Table 1 has no premisses and two conclusions (the order between them is irrelevant), whereas the cut link, on the top right, has two premisses (in any order) and no conclusions). The cut link is presented only for completeness, since the Lambek calculus satisfies cut elimination, we never need to use a cut link when using proof nets for proof search.

The links for the negative implications correspond to the natural deduction elimination rules, though the complex formula is the conclusion of the link and the main premiss of the elimination rule. The links for the positive implications correspond to the introduction rules, with the withdrawn hypothesis as the negative premiss of the rule.

Positive and negative formulas correspond essentially to un-negated and negated formulas (as in the classical equivalences be-

²Another advantage of proof nets is that, unlike natural deduction, adding the product rules to the proof net calculus presents no complications.

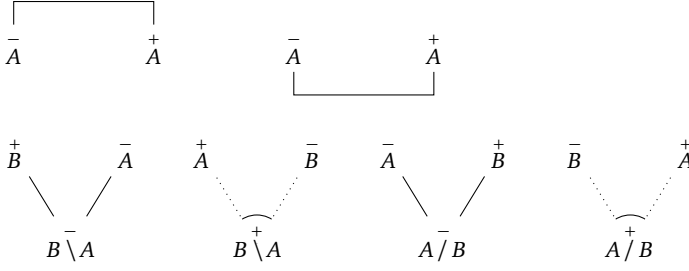


Table 1:
Links for Lambek
calculus proof
structures

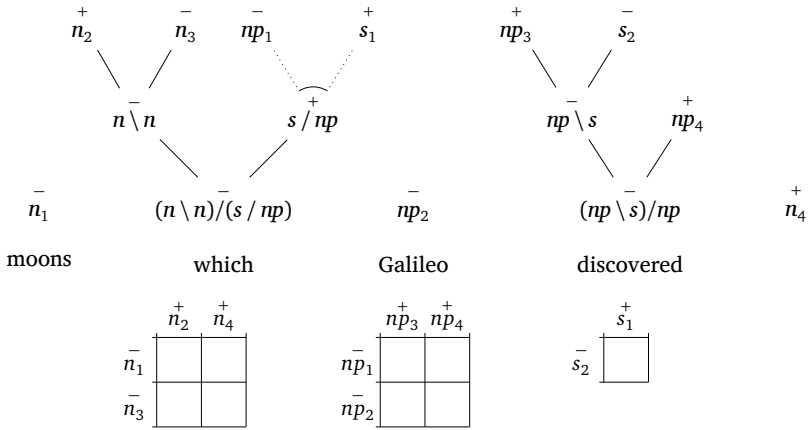
tween $B \rightarrow A \equiv \neg B \vee A$ and $\neg(B \rightarrow A) \equiv B \wedge \neg A$). In the context of linear logic, polarities function as a restriction on classical formulas to ensure the intuitionistic restriction to a single conclusion. The distinction between solid and dotted links corresponds to the distinction between a logical conjunction (solid) and a logical disjunction (dotted). This distinction plays a key role in deciding the correctness of proof structures below.

Given a statement $A_1, \dots, A_n \vdash C$, we obtain a proof frame by unfolding the formulas according to the logical links on the bottom row of Table 1, using the negative unfolding for the A_i and the positive unfolding for C , until we reach the atomic formulas. We then connect the atomic formulas by means of the axiom link shown on the top left of Table 1. We need to respect the linear order of the premisses for the logical links (and the linear order of the formulas in the sequent), but the axiom link can connect a positive and a negative atom in either order.

Figure 2 shows the formula unfolding corresponding to “moons which Galileo discovered”. The occurrences of the atomic formulas have been numbered to allow easy reference to them; these numbers are not a formal part of the proof structure. We saw the natural deduction proof for this noun in Section 2.1. When all axioms of a formula unfolding have been linked we call the resulting structure a *proof structure*. Not all proof structures correspond to proofs. The proof structures which do are *proof nets*. As we will see, we can distinguish proof nets from other proof structures just by looking at properties of the graph.

For the formula unfolding, it is immediately clear what the search space for potential proofs is: we need to find a 1-1 matching between positive and negative occurrences of the same atomic formula. So for Figure 2, we need to match the positive s_1 to the negative s_2 (there

Figure 2:
Formula
unfolding for
“moons which
Galileo
discovered”



is only one possible solution here), the two positive n 's to the two negative ones, and the two positive np 's to the two negative ones. The squares at the bottom of Figure 2 summarise the possibilities.

For Lambek calculus proof nets, the matching of atomic formulas must be planar. Planarity corresponds to non-commutativity of the logic and it therefore holds for the Lambek calculus but not for its extensions. Given that there is only one possibility for s , planarity constrains the possible axiom connections for the np formulas: when the two s formulas have been connected, the negative np_2 corresponding to “Galileo” can only be connected to the leftmost (subject) noun phrase np_3 of “discovered” since connecting it to the rightmost (object) noun phrase would force the link to cross the s axiom link. Similarly, the negative np_1 of “which” can only be linked to the object np_4 of “discovered” when we require a planar connection.

Finally, there are two planar matchings possible between the two positive and the two negative nouns: we either connect the negative n_1 of “moons” to the positive n_2 of “which” and the negative n_3 of “which” to the positive goal n_4 , or vice versa (we had the same choice for natural deduction proof search in Section 2.2).

However, only one of these two possibilities produces a proof net. There are many graph-theoretical ways to characterise the proof nets among other proof structures. One simple way, due to Danos (1990), uses the graph contractions shown in Figure 3.

We first remove all formula information from a proof structure, replacing formula occurrence by unique vertex indices v_0, v_1, \dots , then

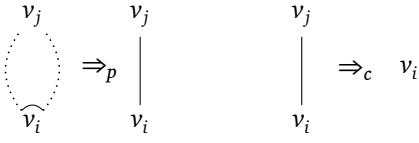
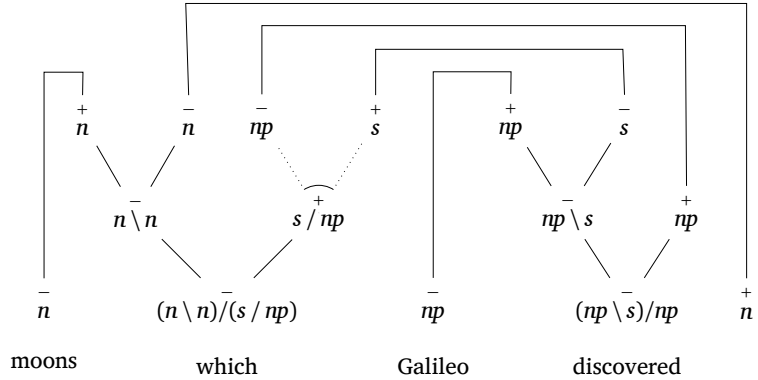


Figure 3:
Contractions for proof nets

applying the contractions. A proof structure is a proof net iff it contracts to a single vertex using these contractions. The condition on the rightmost contraction is that the two vertices are distinct. In other words, if the proof structure has a cycle containing only solid links, then we can use this contraction to reduce this cycle to a self-loop, but we can never eliminate it. Similarly, the contractions only shorten paths, but do not create new ones. Therefore, if a proof structure is disconnected, it will never contract to a single point. The leftmost reduction corresponds to the dotted links for the positive implications and it essentially requires us to “join” the two premisses of the link. As the previous discussion suggests, the contraction criterion is quite close to the more well-known acyclicity and connectedness condition (Danos and Regnier 1989). However, the contraction condition has the advantage of allowing a compact representation of intermediate structures in proof search, and is therefore more suitable for proof search (Moot 2017).

With this in mind, it is clear that of the two possible connections between the n formulas, connecting the two n formulas of “which” together produces a cycle of solid links, and therefore a structure which doesn’t contract to a point. In addition, connecting the noun “moons” to the goal formula produces an axiom link disconnected from the rest of the structure. Therefore, the structure shown in Figure 4 is the only proof net given this sequence of formulas. It is easy to verify it contracts to a point given the contractions of Figure 3. One way of proceeding is eliminating all “dangling” links (that is, links connecting a single vertex to the rest of the structure). When such links have been recursively removed, we end up with the pair of dotted links and a solid path between the positive s and the negative np of this dotted link. We can contract this path to a single vertex, producing the correct configuration for contracting the pair of dotted links and apply the final contraction to the resulting solid link to produce a single vertex.

Figure 4:
Proof net for
“moons which
Galileo
discovered”



3 COMBINATORICS AND COMPLEXITY

Using type-logical grammars for computing the meaning of sentences and using the resulting meaning for different tasks (entailment, question answering, etc.) has the following bottlenecks.

1. Lexical lookup. For wide-coverage grammars, the number of formulas that the lexicon assigns to many common words is rather large.
2. Proof combinatorics. Finding a proof (or the best proof for some numerical definition of best) is NP-complete for most type-logical grammars.
3. Meaning computation. Computing the meaning of a sentence is done by substituting lexical lambda terms and then normalising the resulting simply typed lambda term. Normalising simply typed lambda terms is known to be of non-elementary complexity.
4. Meaning use. Questions of logical entailment between sentences are undecidable, even in the first-order case.

The focus of the rest of this article will be on Item 2, proof combinatorics, but I will offer some brief remarks on the other items.

A probabilistic lexicon The number of lexical formulas per word is a major problem for real-world applications. However, when we fix a set of possible formulas and have enough examples of sentences with the correct formula assignment, we can define a probability model

over words, together with a limited amount of context (typically the two preceding and succeeding words). This general approach is called supertagging (Bangalore and Joshi 2011) and it has been applied successfully to many formalisms including type-logical grammars (Moot 2010, 2014b, presents supertaggers for multimodal type-logical grammars for Dutch and for French).

Since the topic of supertagging has been discussed at length elsewhere and provides good, practical solutions to the problem of lexical ambiguity we will not elaborate on it here.

Meaning computation Schwichtenberg (1982) shows that normalising simply typed lambda terms has non-elementary worst-case complexity. This complexity result essentially exploits recursive copying. However, there are many implementations of the simply typed lambda calculus for computational linguistics which perform rather efficiently, and this in spite of the fact that, in general, little effort is spent on optimising the implementation of the normalisation component. We claim that the meaning recipes necessary for the lexicon are all terms of soft linear logic, and hence can be reduced in polynomial time (Lafont 2004; Baillot and Mogbil 2004). This accounts for the observed fact that lambda term normalisation is not a real bottleneck in practice (Moot and Retoré 2016).

Logical entailment Given that logical entailment is undecidable in general, there are two basic strategies:

1. We can use an off-the-shelf theorem prover (generally using some time limit) and simply see whether it finds a proof. Bos and Markert (2005) use this approach (as well as some more approximative measures).
2. We can use an incomplete but decidable logical fragment for computing entailment. Abzianidze (2017) uses this approach.

In both cases, the result is a high-precision but low-recall system (that is, when the system produces an answer, it is usually right, but there are many correct answers for which no proofs are found). The main bottleneck to improving recall is adding a logical formalisation of a sufficient amount of world knowledge (without reducing prover performance), a classic problem in artificial intelligence.

WEIGHTED PROOF SYSTEMS

Given a sequent $\Gamma \vdash C$, the formula unfolding into a proof frame gives a compact representation of the proof combinatorics for the given sequent. We can combine positive and negative occurrences of the same atomic formulas until we obtain a proof structure. An atomic formula a with n positive and n negative occurrences (the number of positive and negative occurrences for each atomic formula must be equal if the sequent is derivable; this is called the *count check*) corresponds to an $n \times n$ matrix, and a potential reading for this sequent, that is a proof structure, is a perfect matching between positive and negative occurrences.

When we fill the matrix with weights (we will discuss some different ways of computing these weights below), it becomes possible to compute the best proof structure according to these weights. We can either use best-first search, connecting the “best” axiom links first for each local choice, or use a k -best proof structure computation, computing the k total links which are the best globally. Given that computing the k -best proof structures this way can be done in polynomial time (Kuhn 1955), there is no guarantee that the best proof structure is actually a proof net (unless $P = NP$). However, we can use a polynomial k -best system as an incomplete approximation of proof search.

Given two atomic formulas a_1 and a_2 of the same type (n, np, s) but of opposite polarity there are different ways of assigning a weight to the possible axiom link between them.

1. We can use the distance between words as weight, using distance 0 when the two atomic formulas are subformulas of the same formula occurrence, distance 1 between adjacent words, etc.
2. We can use a probability-like measure, estimated from proof nets in a large corpus.
3. We can use the word similarity measure between w_1 and w_2 .

We will discuss each of these alternative measures in turn in the next sections.

4.1

Word distance

One simple metric to use is word distance, preferring axiom connections between closer words. This gives a strong preference to linking

atomic subformulas of the same formula, followed by linking atomic subformulas of adjacent formulas.

Given a proof net with k axiom links, when processing this proof net left-to-right performing axiom links as soon as possible, this will mean each link of distance of n will cause the leftmost corresponding atom to be open/unlinked for n steps. Measuring the number of open axioms after each word has been proposed as a straightforward model of human sentence processing which, in spite of its simplicity, makes a number of correct predictions about processing (Johnson 1998; Morrill 1998, 2011).

We illustrate this by examining the contrast between Dutch and German verb clusters. Verb clusters in both Dutch and German can have a sequence of verb arguments followed by a sequence of verbs selecting these arguments. The difference between German and Dutch is that German verbs select these arguments right-to-left (that is, the leftmost verb selects the rightmost argument) and the Dutch verb select these arguments left-to-right (that is, the leftmost verb selects the leftmost argument). This is illustrated by the following contrast.

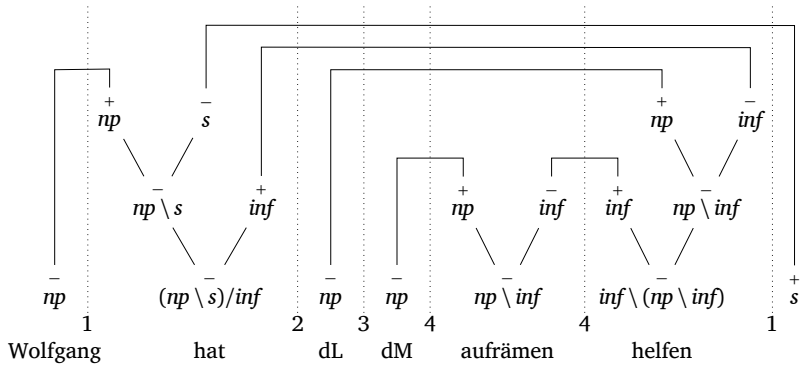
- (1) Wolfgang hat die Lehrerin die Murmeln aufräumen helfen
 Wolfgang has the teacher the marbles collect help
 Wolfgang helped the teacher collect the marbles
- (2) Jantje heeft de lerares de knikkers helpen opruimen
 Jantje has the teacher the marbles help collect
 Jantje helped the teacher collect the marbles

Bach *et al.* (1986) find that, in an experimental setting, German sentences like the one above are harder for German native speakers than the Dutch sentences are for Dutch native speaker: although the difference is not very large, the German sentences are not only judged as harder by the German speakers, but the German test subjects also make more comprehension errors.

Figure 5 shows the proof net corresponding to sentence (1). The noun phrases “die Lehrerin” and “die Murmeln” have been not been treated as a combination of np/n and n but as a simple np to reduce the size of the proof net. This will not affect the comparison with the Dutch example.³

³Some other simplifications have been made: neither the German auxiliary “hat” nor the Dutch auxiliary “heeft” can select a simple infinitive argument (i.e.

Figure 5:
Proof net for
“Wolfgang hat
die Lehrerin die
Murmeln
aufräumen
helfen”



Computing the processing complexity using a proof net such as the one shown in Figure 5 requires us to make some choices. We can assume that the hearer knows to expect a sentence (and therefore put the goal formula initially). Morrill (2011) chooses this option. We can also keep the goal formula at the end, as done here.

There are some other potential complexity issues to take into account: some choices of lexical formulas and of axiom links lead to failure and it is possible that this affects processing complexity (at least it does so for a computer implementation). The size of the partial proof net constructed so far may also play a role (the size of the contracted partial proof net according to the contractions of Figure 3 seems a good candidate for such a size measure).

Morrill and Johnson use the simplest solution here, measuring complexity by the successful proof nets when processed left to right, counting the number of unlinked axioms at each step.

Figure 5 shows a dotted column after each word, together with a count of the number of axioms it crosses. In the example, after the first word, “Wolfgang”, there is a single unlinked *np*, therefore the count is 1. After “hat”, the *np* of “Wolfgang” becomes linked but an unlinked *s* and an unlinked *inf* are added, leaving a total of 2 unlinked atoms.

we have “Die Lehrerin hat die Murmeln aufgeräumt”, with a past participle rather than an infinitive); they only take an infinitive argument when this infinitive itself selects for another infinitive. This can be solved either by adding features or by distinguishing the atomic types. Bach *et al.* (1986) note that for German (but not for Dutch) both grammar textbooks and speakers disagree over whether the final verb should be an infinitive or a past participle.

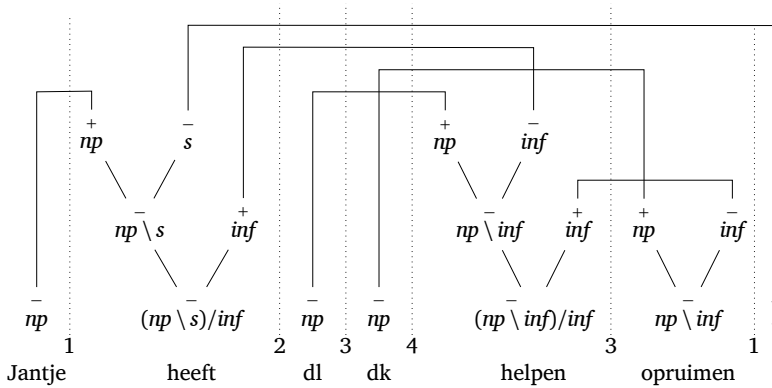


Figure 6:
Proof net for
“Jantje heeft de
lerares de
knikkers helpen
opruimen”

In general, the complexity profile of verb clusters in German (and in Dutch) rises when the arguments of the verbs in the cluster are encountered (the noun phrases “die Lehrerin” and “die Murmeln”), then descends when the verbs start selecting their arguments. This provides an explanation for why these sentences quickly become unacceptable with multiple levels of embedding.

The complexity profile of Figure 5 has a maximum complexity of 4 and a total complexity of 15.

To provide a proof net for the Dutch example (2), we need somewhat more complex proof net machinery, since the crossed dependencies of Dutch cannot be handled by planar structures. Since these more complicated proof nets do not affect our chosen complexity measure, we will simply look at the complexity profile for the non-planar proof net in Figure 6: this is not a Lambek calculus proof net, and we assume a proof net calculus which allows only this structure for the example sentence.

The complexity profiles of the two sentences are the same until the first infinitive, but then the Dutch sentence has a slight advantage: its maximum complexity, like the German example, is 4; but its total complexity is 14, compared to 15 for the German example. This advantage becomes somewhat more pronounced when we add a third and a fourth verb.

These examples are interesting because many known psycholinguistic facts, even some fairly subtle ones like shown here, are a direct consequence of rather minimal assumptions about a model of processing. Morrill (2011) presents many other examples.

4.2

Corpus estimation

Given a sequent, it is not hard to define a probability distribution over its proof structures: to obtain such a probability distribution we simply need to fill the $n \times n$ matrix for each atomic formula in such a way that all rows and all columns sum to 1. In other words, each atomic formula is assigned a probability distribution over the atomic formulas of opposite polarity. For example, looking to Figure 6, for a full proof search the negative np corresponding to ‘Jantje’ can be linked to the positive np of ‘heeft’, the positive np of ‘helpen’ and the positive np of ‘opruimen’ and the sum of these probabilities must be 1. Similarly, the positive np of ‘heeft’ can be linked to the negative np of ‘Jantje’, the negative np of ‘de lerares’ and the negative np of ‘de knikkers’.

Mathematically, it is much harder to define a probability distribution over proof *nets*. When we define a probability distribution over proof structures, we assign a non-zero probability to structures which do not correspond to proofs. Even though it makes sense a priori to want non-proofs to have zero probability, this entails that an undervivable sequent should fail to be assigned a probability distribution at all, since no axiom link can contribute to a proof. However, this means assigning probabilities becomes an NP-hard problem, since we would be able to decide derivability of a sequent from the success or failure of computing a probability distribution.

Accepting the assignment of non-zero probability to axiom links which are not part of any proof is comparable to probabilistic context-free grammar parsers assigning non-zero probability to constituents which cannot be part of a derivation of the complete string.

The question for probability assignment is how likely is the n th atomic formula of word w_1 to combine with the k th atomic formula of w_2 (with some form of backoff, for example to the two part-of-speech tags). In general, we can use well-known statistical methods (Berger *et al.* 1996) to compute a probability function from any combination of properties from formulas, words and context. This possibility has so far been little explored in the context of type-logical grammars.

In terms of assigning weights to atomic formulas, this is not fundamentally different from the other weighted approaches discussed here. It has the advantage over the other methods that it can distinguish between different arguments of the same formula (neither the distance

measure nor the vector similarity measure does this). However, it has the disadvantage that it requires a large amount of annotated data to estimate the probabilities.

4.3

Vector similarity

An advantage of using vector similarity rather than a large corpus of parsed text is that it is much easier to obtain the former than it is to obtain the latter: a high-quality parsed corpus of sufficiently large size requires an enormous effort in times of person-hours; on the other hand, computing word vectors can be done automatically and, using the enormous amount of text available on the internet, on a scale unrealistic for any manual method. Computing word vectors from the web still requires an important effort in crawling, cleaning, duplicate detection, etc., but nowhere near the person-hours needed to manually annotate a similar size corpus, something especially relevant for under-resourced languages: as discussed by Kilgarriff and Grefenstette (2003), even ‘smaller’ languages such as Icelandic, Basque, Latin and Esperanto have over 50 million words of text available according to conservative estimates. For many of the most-used languages on the internet, cleaned-up and (automatically) annotated versions of this content are freely available and can be used to extract word vectors (Baroni *et al.* 2009).

Weighting axiom links according to the similarity of the words given by distributional semantics means preferring connections between words with related meanings (this appears to be close to the notion of *discourse coherence* as used by Asher and Lascarides 2003, only in a more shallow, syntactic context).

Even though this basic idea is easily stated, implementing it requires making some choices. While distributional semantic similarity is easily defined for two words, defining it for two complex expressions is essentially the compositionality problem for vector space semantics.

A simple solution would be to choose the vector sum for composition, and this already performs surprisingly well on several similarity tasks. However, the vector sum approach is ill-adapted to preferences in type-logical proofs: given that we need to match the atomic subformulas of all words in a sentence in any case, and given that the vector sum operation is associative and commutative, this would not allow

us to distinguish between different word groupings (or even between different word orders).

We therefore need a slightly more sophisticated method for combining vector similarity with proof rules. We adapt the basic idea of lexicalised parsing with context-free grammars and use a *head word* for each expression — the verb for a verb phrase, the noun for a noun phrase, etc. In the context of type-logical grammar, we therefore specify the following general principles, as sort of type-logical equivalences to the head percolation principles of Magerman (1994):

1. the head of a lexical hypothesis is the word itself,
2. the head of the combination of A/A with A and of A with $A\backslash A$ is the head of A
3. the head of np/n with n (resp. pp/np with np) is the head of the noun n (resp. the head of the noun phrase np)
4. the head of other formulas A/B and $B\backslash A$, with $A \neq B$, is the head of A .

Moreover, for the semantic assignments in a type-logical lexicon, we can distinguish between the lexical entries whose semantic content is purely logical (using only the logical constants like “ \neg ” “ \wedge ”, “ \forall ”, and a few other predicates whose meaning is invariant across models, like “ $=$ ” and “ $<$ ”) and those whose semantic content is not (these typically contain predicates like “*love*” and “*book*” corresponding to the lexical entry itself).⁴

The basic elements in our formal setup are now triples containing a *formula*, a *head word*, and *vector distance weight*, with each lexical entry starting with the word lemma as its head and distance zero. For each elimination rule, the new head word is defined by the propagation rules above (it is the head of the argument when the functor is a modifier, a determiner or a preposition, and the head of the functor otherwise) and the weight is updated by adding the weights of the two premisses and additionally adding the distance of the two head words according to the vector model. The rule be-

⁴This contrast is unfortunately not as sharp as we would like it to be: while it is simple to see “all” and “some” as purely logical, it is much less easy to see how other words such as “few” and “many” can be interpreted in terms of purely logical operators.

low presents a general elimination rule operating on triples (with “ \multimap ” generalising over both “/” and “\”) according to this description.

$$\frac{\langle w_1, h_1, A \multimap B \rangle \quad \langle w_2, h_2, A \rangle}{\langle w_1 + w_2 + d(h_1, h_2), h, B \rangle} \multimap E$$

From a purely logical point of view (that is, looking only at the third element of the triple), this rule operates just like a normal elimination rule. The head h of the conclusion will be either h_1 or h_2 depending on the head propagation rules described above. The weight computation uses a distance measure d computing the distance between the two head words and simple addition. Nothing in particular hinges on the use of “+” here; any function monotone in both its arguments can be used here. Many choices are also possible for the distance measure d , but in the examples below we use the simple cosine measure which is the most commonly used for distributional similarity. The cosine measure produces 1 when the vectors point in the exact same direction, 0 when the vectors are orthogonal.⁵ This ensures the highest-weight proof combines the nearest vectors.

For the introduction rules, it is somewhat more difficult to define the proper elements for the discharged hypothesis of the rule. We can simply choose zero for its weight, but it is unclear what the proper head word for a hypothesised constituent is. One simple solution is to assign the empty word ϵ to such hypotheses and stipulate that the empty word has distance zero to all other words (i.e. for all w , $d(\epsilon, w) = d(w, \epsilon) = 0$). This would give the following introduction rule.

$$\frac{\begin{array}{c} \langle 0, \epsilon, B \rangle \\ \vdots \\ \langle w, h, A \rangle \end{array}}{\langle w, h, B \multimap A \rangle} \multimap I$$

We can also use a somewhat more sophisticated rule for subproofs of the following form.

⁵In principle, we can have -1 when the vectors point in the exact opposite direction, although many methods are guaranteed to obtain positive vectors only.

$$\frac{\frac{\vdots}{\langle w_1, h_1, (B \multimap A) \multimap C \rangle} \quad \frac{\frac{\vdots}{\langle w_2, h_2, A \rangle} \quad \langle w_2, h_2, B \multimap A \rangle}{\langle w_2, h_2, B \multimap A \rangle} \multimap I_k}{\langle w_1 + w_2, h_1, C \rangle} \multimap E$$

Where for relative pronouns $A = s$, $B = np$ and $C = n \setminus n$, and for generalised quantifiers $A = s$, $B = np$ and $C = s$. Essentially, h_1 is propagated from the left premiss of the elimination rule to the hypothesis of the introduction rule. This works well since the head word of a determiner phrase (of type $(np \multimap s) \multimap s$) is its noun, and similarly, the noun argument of the relative pronoun is semantically identical to the extracted noun phrase $B = np$.

As a concrete example, a French fragment like “concert de piano gratuit”, like its English translation “free piano concert”, has two possible readings, one where there is a piano concert which is free and one where a free piano is used to give a concert. This type of ambiguity, although somewhat reduced by noun/adjective agreement, is quite common in the French Treebank (Abeillé *et al.* 2003) and apparently a difficult construction both for journalists and annotators. Treating this example according to the method described above provides the following (to reduce horizontal space, we have used w_1 , w_2 and w_3 for the weight terms to be discussed later).⁶

$$\frac{\frac{\text{concert}}{\langle 0, \text{concert}, n \rangle} \text{Lex} \quad \frac{\frac{\text{de}}{\langle 0, \text{de}, (n \setminus n) / n \rangle} \text{Lex} \quad \frac{\text{piano}}{\langle 0, \text{piano}, n \rangle} \text{Lex}}{\langle w_1, \text{piano}, n \setminus n \rangle} \text{/E}}{\langle w_2, \text{concert}, n \rangle} \text{\setminus E}}{\langle w_3, \text{concert}, n \rangle} \text{Lex}}{\langle 0, \text{gratuit}, n \setminus n \rangle} \text{\setminus E}} \text{Lex}$$

The weight w_1 of the proof showing “de piano” is of type $n \setminus n$ is equal to the weight of its two premisses (both zero) plus the distance between the heads of the two premisses of the rule, “de” and “piano” in our case, which have a distance of 0.0740, so we conclude w_1 is 0.0740. We can now compute the weight of the proof showing “concert de piano” to be of type n by combining the weight 0 of “concert” with

⁶ Here and elsewhere, all weights are computed using the models provided by Fauconnier (2016) at <http://fauconnier.github.io/#software>

the weight 0.0740 of “de piano” and adding the distance between the two head words “concert” and “piano”, which is 0.4398 (that is, these words are fairly close) to arrive at $w_2 = 0.5138$. Finally, we combine this n with the adjective “gratuit” to compute the final weight w_3 by adding the previously computed w_2 to 0 (the weight of “gratuit”) and adding the distance between “concert” and “gratuit”, which is 0.1921. This gives us a total weight w_3 of 0.7059 for this reading (note that this number is not a probability and meaningful only in comparison to similarly computed numbers).

The second proof looks as follows.

$$\frac{\frac{\text{concert}}{\langle 0, \text{concert}, n \rangle} \text{Lex} \frac{\frac{\text{de}}{\langle 0, \text{de}, (n \setminus n) / n \rangle} \text{Lex} \frac{\frac{\text{piano}}{\langle 0, \text{piano}, n \rangle} \text{Lex} \frac{\frac{\text{gratuit}}{\langle 0, \text{gratuit}, n \setminus n \rangle} \text{Lex}}{\langle w_1, \text{piano}, n \rangle} \setminus E}{\langle w_2, \text{piano}, n \setminus n \rangle} / E}{\langle w_3, \text{concert}, n \rangle} \setminus E$$

Even though the second reading uses the exact same words, it combines them in a different way, and this affects the weight calculations. We now combine “piano” and “gratuit” first, to obtain $w_1 = 0 + 0 + d(\text{piano}, \text{gratuit}) = 0.0695$. We then combine this result with “de” and calculate $w_2 = 0 + w_1 + d(\text{de}, \text{piano}) = 0.0695 + 0.0740 = 0.1435$. Finally, we combine the previous result with “concert” and calculate $w_3 = 0 + w_2 + d(\text{concert}, \text{piano}) = 0.1435 + 0.4398 = 0.5833$.

These calculations show a preference for the first reading, where the concert is free rather than the piano. The key difference between the two readings is that $d(\text{concert}, \text{gratuit}) > d(\text{piano}, \text{gratuit})$, whereas the other computed terms are equal.

We can use the same method to compute “voir la fille avec les lunettes” (to see the girl with the glasses), since $d(\text{voir}, \text{lunettes}) > d(\text{fille}, \text{lunettes})$, which gives a preference for “voir ... avec les lunettes” over “fille avec les lunettes”.

Using semantic relatedness like this is, of course, not without its defects. For example, the verb phrase “saw the star with the telescope” is structurally identical to the example above, but has only one plausible reading, where “with the telescope” is a verb phrase modifier. However, “star” and “telescope” are closer semantically than “girl” and “telescope” are. The problem here is essentially that semantic vector similarity as we are using it here doesn’t give us any information about argument structure.

This suggests the need for more sophisticated ways to combine vector semantics, such as used by Baroni and Lenci (2010). In the context of a real-world system, the lexicon is a probability distribution over a finite set of formulas and therefore the highest-weight proof for a sentence must be a combination of the probability over the formulas with the weight over the axioms. The right way of combining the weights of the supertagger (a probability distribution over formulas) with the vector weights needs to be determined empirically, of course. Two simple possibilities are:

1. taking the best supertagger sequence for which a proof is found, then finding the maximum weight proof for this sequence;
2. combine the supertagger probabilities with the weight of the proof into a single weighted sum; that is, we treat finding the relative importance of the two weights as a standard machine learning objective to be determined empirically.

In the case of “voir l’étoile avec le télescope” (*see the star with the telescope*), the supertagger (Moot 2014b) strongly prefers adverbial use of “with” over adjectival use (26.4% against 0.9%), a very strong preference in favour of the preferred reading. Therefore, in a real-world system this problem disappears unless the weighted sum gives a very strong priority to the vector weight component.

4.4 *Vector similarity and proof nets*

We can adapt the above strategy with minor modifications to a proof net parser. This is done by replacing atomic formulas by binary predicates, where the first argument represents the head and the second argument the weight. For weights, we use the real numbers with the usual function “+” and the $d(w_1, w_2)$ function computing the distance between two words w_1 and w_2 . Instead of having extra-logical head percolation and weight computation principles, these now form a part of the lexical entries (although these extra-logical principles would explain many common patterns occurring in the lexical entries and would make it possible to create an automatic compilation step adding the head and weight arguments to a ‘standard’ lexicon). Using first-order arguments has many applications, including as a solution for the Dutch verb clusters we’ve seen in Section 4.1 (Moot 2014a). As we use them here, these arguments are extra-grammatical and serve

$$\begin{aligned}
lex(book) &= n(book, 0) \\
lex(the) &= np(X, w)/np(X, w) \\
lex(interesting) &= n(X, w + d(interesting, X))/n(X, w) \\
lex(read) &= (np(X, w_1) \setminus s(read, w_1 + w_2 + d(w_1, read) + d(w_2, read)))/np(Y, w_2) \\
lex(which) &= (n(X, w_1) \setminus n(X, w_1 + w_2))/(s(Y, w_2)/np(X, 0)) \\
lex(every) &= (s(Y, w_1 + w_2)/(np(X, 0) \setminus s(Y, w_2)))/n(X, w_1)
\end{aligned}$$

Table 2:
Lexical
assignments with
head word and
weight
computation
information

only as a way to compute preferences among different proofs using the same formulas.

As before, atomic content words, like the noun “book” are assigned the entry $n(book, 0)$. That is, the head constituent of the noun is “book” itself and the weight assigned to this expression is zero. Table 2 lists some other lexical entries in the current context.

As shown in the table, the determiner “the” is assigned an entry simply copying both the head word and the weight, following the principles of a purely logical word in the previous section.

An adjective such as “interesting” on the other hand copies the head word, but adds the distance between the head word to the previous weight.⁷

Similarly, the transitive verb “read” adds both the distance between the verb and its subject and the distance between the verb and its object to the weight of its two arguments.

The weighted entry for “which” requires some explanation. First of all, the head word X of the resulting noun is the same as the head of the argument noun (this behaviour is consistent with a noun modifier, and it makes, for example, “book which ...” behave the same as “book” in this respect). Second, the extracted np , being a hypothetical element, starts at weight 0 and shares the head with the noun argument. This approach therefore expects the long-distance dependency

⁷ Modifiers of modifiers (e.g. adverbs like “very”) cannot be handled in the same way as in the natural deduction based account of the previous section. Since they modify the adjective or adverb they select, and since these no longer contain this adjective or adverb as their head word, we can no longer compute the distance between e.g. “very” and “interesting”, and between “very” and “quickly” since both “interesting” and “quickly” are not arguments of any of their atomic subformulas. The natural deduction approach of the previous section assigns heads to formulas with no requirement that these be atomic, and this provides a potential benefit for these cases.

between a noun and a relative clause to occur most likely between the head noun and the verb which is semantically closest to it.

Compared to the standard proof net matching algorithm using minimisation (or maximisation) of weight, we have now added computation of weights to the matching process. This presents something of a complication. However, since we need to do only simple computations (addition, vector cosine) in each cell of the matrices representing the search space, this doesn't make a big difference computationally.

4.5

Limitations

The current method doesn't distinguish between object and subject arguments and this is an important weakness.⁸ This limitation is essentially a consequence of the division of labour between the distributional and type-logical approaches: the type-logical component of the system is solely responsible for word order while the distributional component only tests for similarity between a verb and its argument, taking neither grammatical nor structural considerations into account. We therefore need either a more subtle similarity measure or another way of distinguishing the likely relations between a verb and its different arguments.

5

CONCLUSIONS AND FUTURE WORK

We have given an overview of several methods of adding weights to proof search in type-logical grammars. With the exception of Bonfante and de Groote (2001), this possibility has been seldom discussed in the type-logical grammar literature, to our surprise. We have given applications of weighted proof search to modelling human processing, to finding parses most similar to those found in a given corpus, and to finding parses which prefer grouping similar words together. These methods still need to be thoroughly evaluated beyond the manually calculated examples shown here. Fortunately, for many current type-logical grammars and their theorem provers, the groundwork for incorporating weighted proof search has already been laid down. Given

⁸ As discussed in Section 4.2, we can incorporate this when estimating probabilities from a corpus, but not for the other methods discussed here, i.e. vector similarity and word distance.

the many potential applications of weighted proof search, we look forward to testing these methods against available data for parsing and human processing.

REFERENCES

- Anne ABEILLÉ, Lionel CLÉMENT, and François TOUSSENEL (2003), Building a Treebank for French, in Anne ABEILLÉ, editor, *Treebanks*, volume 20 of *Text, Speech and Language Technology*, pp. 165–187, Springer.
- Lasha ABZIANIDZE (2017), *A natural proof system for natural language*, Ph.D. thesis, Tilburg University.
- Nicolas ASHER and Alex LASCARIDES (2003), *Logics of Conversation*, Cambridge University Press.
- Emmon BACH, Colin BROWN, and William MARSLEN-WILSON (1986), Crossed and Nested Dependencies in German and Dutch: A Psycholinguistic Study, *Language and Cognitive Processes*, 1(4):249–262.
- Patrick BAILLOT and Virgile MOGBIL (2004), Soft Lambda-calculus: A Language for Polynomial Time Computation, in *Foundations of software science and computation structures*, pp. 27–41, Springer.
- Srinivas BANGALORE and Aravind JOSHI (2011), *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*, MIT Press, Cambridge, Massachusetts.
- Marco BARONI, Silvia BERNARDINI, Adriano FERRARESI, and Eros ZANCHETTA (2009), The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora, *Language Resources and Evaluation*, 43(3):209–226.
- Marco BARONI and Alessandro LENCI (2010), Distributional Memory: A General Framework for Corpus-based Semantics, *Computational Linguistics*, 36(4):673–721.
- Adam BERGER, Stephen DELLA PIETRA, and Vincent DELLA PIETRA (1996), A Maximum Entropy Approach to Natural Language Processing, *Computational Linguistics*, 22(1):39–71.
- Guillaume BONFANTE and Philippe DE GROOTE (2001), Stochastic Lambek Categorical Grammars, in Geert-Jan KRUIJFF, Larry MOSS, and Richard T. OEHRLE, editors, *Proceedings of FGMOL 2001*, volume 53 of *Electronic Notes in Theoretical Computer Science*, Elsevier.
- Johan BOS and Katja MARKERT (2005), Recognising Textual Entailment with Logical Inference, in *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pp. 628–635.

Vincent DANOS (1990), *La logique linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ -calcul)* [Linear logic applied to the study of various normalisation processes (mainly of the lambda calculus)], Ph.D. thesis, University of Paris VII.

Vincent DANOS and Laurent REGNIER (1989), The Structure of Multiplicatives, *Archive for Mathematical Logic*, 28:181–203.

Jean-Philippe FAUCONNIER (2016), *Acquisition de liens sémantiques à partir d'éléments de mise en forme des textes : exploitation des structures énumératives* [Acquisition of semantic relations from text layout elements: exploitation of enumerative structures], Ph.D. thesis, Université de Toulouse.

L. T. F. GAMUT (1991), *Logic, Language and Meaning*, volume 2, The University of Chicago Press.

Jean-Yves GIRARD (1987), Linear Logic, *Theoretical Computer Science*, 50:1–102.

Mark JOHNSON (1998), Proof Nets and the Complexity of Processing Center-Embedded Constructions, *Journal of Logic, Language and Information*, 7(4):443–447.

Adam KILGARRIFF and Gregory GREFENSTETTE (2003), Introduction to the Special Issue on the Web as Corpus, *Computational Linguistics*, 29:333–347.

Harold W. KUHN (1955), The Hungarian Method for the Assignment Problem, *Naval Research Logistics Quarterly*, 2:83–97.

Yves LAFONT (2004), Soft Linear Logic and Polynomial Time, *Theoretical Computer Science*, 318(1):163–180.

Joachim LAMBEK (1958), The Mathematics of Sentence Structure, *American Mathematical Monthly*, 65:154–170.

David M. MAGERMAN (1994), *Natural language parsing as statistical pattern recognition*, Ph.D. thesis, University of Pennsylvania.

Jeff MITCHELL and Mirella LAPATA (2010), Composition in Distributional Models of Semantics, *Cognitive Science*, 34:1388–1429.

Michael MOORTGAT (1997), Categorical Type Logics, in Johan VAN BENTHEM and Alice TER MEULEN, editors, *Handbook of Logic and Language*, chapter 2, pp. 93–177, Elsevier/MIT Press.

Richard MOOT (2010), Automated Extraction of Type-logical Supertags from the Spoken Dutch Corpus, in Srinivas BANGALORE and Aravind JOSHI, editors, *Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach*, chapter 12, pp. 291–312, MIT Press, Cambridge, Massachusetts.

Richard MOOT (2014a), Extended Lambek Calculi and First-order Linear Logic, in Claudia CASADIO, Bob COECKE, Michael MOORTGAT, and Philip SCOTT, editors, *Categories and Types in Logic, Language, and Physics: Essays dedicated to*

Jim Lambek on the Occasion of this 90th Birthday, number 8222 in Lecture Notes in Artificial Intelligence, pp. 297–330, Springer, Heidelberg.

Richard MOOT (2014b), A Type-logical Treebank for French, *Journal of Language Modelling*, 2(2).

Richard MOOT (2017), The Grail Theorem Prover: Type Theory for Syntax and Semantics, in Zhaohui LUO and Stergios CHATZIKYRIAKIDIS, editors, *Modern Perspectives in Type Theoretical Semantics*, Springer.

Richard MOOT and Christian RETORÉ (2012), *The Logic of Categorical Grammars: A Deductive Account of Natural Language Syntax and Semantics*, number 6850 in Lecture Notes in Artificial Intelligence, Springer, Heidelberg.

Richard MOOT and Christian RETORÉ (2016), Natural Language Semantics and Computability, Technical report, LIRMM.

Glyn MORRILL (1998), Incremental Processing and Acceptability, Technical Report LSI-98-46-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.

Glyn MORRILL (2011), *Categorical Grammar: Logical Syntax, Semantics, and Processing*, Oxford University Press, Oxford.

The Nghia PHAM (2016), *Sentential Representations in Distributional Semantics*, Ph.D. thesis, University of Trento.

Helmut SCHWICHTENBERG (1982), Complexity of Normalization in the Pure Typed Lambda-Calculus, in *The L. E. J. Brouwer Centenary Symposium*, pp. 453–457, North-Holland.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



Static and dynamic vector semantics for lambda calculus models of natural language

Mehrnoosh Sadrzadeh¹ and Reinhard Muskens²

¹ School of Electronic Engineering and Computer Science,
Queen Mary University of London.

² Department of Philosophy, Tilburg University

ABSTRACT

Vector models of language are based on contextual aspects of language – distributions of words and how they co-occur in text. Truth conditional models focus on logical aspects of language and on how words combine to contribute to these aspects. In the truth conditional approach, there is a focus on the denotations of phrases. In vector models, the degree of co-occurrence of words in context determines how similar their meanings are. The two approaches have complementary virtues. In this paper we combine them and develop a vector semantics for language, based on the typed lambda calculus. We provide two types of vector semantics: a static one using techniques from the truth conditional tradition, and a dynamic one with a form of interpretation inspired by Heim’s context change potentials. We show, with examples, how the dynamic model can be applied to entailment between a corpus and a sentence.

Keywords: simply typed lambda calculus, vector semantics, composition, context update potential, dynamic logic

1

INTRODUCTION

Vector semantic models, otherwise known as distributional models, are based on the contextual aspects of language, i.e. the company each word keeps, and patterns of use in corpora of documents. Truth conditional models focus on the logical and denotational aspects of language. They typically describe how words can be represented by

functions over sets, and how these functions can be composed. Vector semantics and truth conditional models are based on different philosophies: one takes the stance that language is contextual, the other asserts that it is logical. In recent years, there has been much effort to bring these two together. We have models based on a certain type of grammatical representation, e.g. the pregroup model (Coecke *et al.* 2010), the Lambek Calculus model (Coecke *et al.* 2013), and the combinatorial categorial models (Krishnamurthy and Mitchell 2013; Maillard *et al.* 2014). We also have more concrete models that draw inspiration from type theory, but whose major contribution lies in developing concrete ways of constructing linear and multi-linear algebraic counterparts for syntactic types, e.g. matrices and tensors (Grefenstette and Sadrzadeh 2015; Baroni *et al.* 2014), and relational clusters (Lewis and Steedman 2013).

What some of these approaches (Coecke *et al.* 2010; Krishnamurthy and Mitchell 2013; Maillard *et al.* 2014) lack more than others (Baroni *et al.* 2014; Lewis and Steedman 2013) is acknowledgement of the inherent gap between contextual and truth conditional semantics: they closely follow truth theoretic conditions to assign vector representations to (readings of) phrases and sentences.¹ Indeed, it is possible to develop a stand-alone compositional vector semantics along these lines, but this will result in a static semantics. From the perspective of the underlying theory, it will also be quite natural to have a vector semantics work in tandem with a dynamic theory, and let the two modules model different aspects of meaning. Distributional semantics is particularly apt at modelling associative aspects of meaning, while truth-conditional and dynamic forms of semantics are good at modelling the relation of language to reality, and also at modelling entailment. It is quite conceivable that a theory combining the two as separate modules will be simpler than trying to make one of the approaches do things it was never intended for.

In this paper, we first sketch how an approach to semantics, derived in many of its aspects from that pioneered by Montague (1974), can be used to assign vector meanings to linguistic phrases. The theory will be based on simply typed lambda calculus and, as a result, will

¹ Below, when we refer to phrases and sentences, strictly speaking, we mean *readings* of phrases and sentences.

be neutral with respect to the linguist's choice of syntax, in the sense that it can be combined with any existing syntax-semantics interface that assumes that the semantics is based on lambdas.² Our reason for using lambda calculus is that it directly relates our semantics to higher order logic, and makes standard ways of treating long-distance dependencies and coordination accessible to vector-based semantics. This approach results in a semantics similar to those of the static approaches listed above. The reason for providing it is to show that a lambda calculus model of language can be directly provided with a straightforward vector semantics. As will be seen, abstract lambda terms, which can be used as translations of linguistic expressions, have much in common with the Logical Forms of these expressions, and the lambda binders in them facilitate the treatment of long-distance dependencies. The use of lambda terms also makes standard ways of dealing with coordination accessible to distributional semantics. We provide extensive discussion of this process, and examples where the direct use of lambdas is an improvement on the above-listed static approaches.

The above semantics does not have an explicit notion of context, however. The second contribution of this paper is that, based on the same lambda calculus model of natural language, we develop a dynamic vector interpretation for this type theory, where denotations of sentences are "context change potentials", as introduced by Heim (1983). We show how to assign such a vector interpretation to words, and how these interpretations combine so that the vectors of the sentences containing them change the context, in a dynamic style similar to that proposed by Heim. As context can be interpreted in different ways, we work with two different notions of context in distributional semantics: co-occurrence matrices, and entity-relation graphs,

²Linguistic trees, for example, can be associated with the abstract lambda terms considered below via type-driven translation (Partee 1986; Klein and Sag 1985; Heim and Kratzer 1998). But other syntactic structures can be provided with them as well. In the framework of Lexical-Functional Grammar, abstract lambda terms can be assigned to f-structures with the help of linear logic acting as 'glue' (Dalrymple *et al.* 1993). In Combinatory Categorical Grammar, derivations can be associated with abstract lambda terms using combinators (Steedman 2000), while proofs in Lambek Categorical Grammar can be provided with them by the Curry-Howard morphism (van Benthem 1986).

encoded here in the form of cubes. Both of these are built from corpora of documents and record co-occurrence between words: in a simple neighbourhood window, in the case of co-occurrence matrices, and in a window structured by grammatical dependencies, in the case of an entity-relation cube. We believe our model is flexible enough for other distributional notions of contexts, such as networks of grammatical dependencies. We show how our approach relates to Heim’s original notion of ‘files’ as contexts. Other dynamic approaches, such as update semantics (Veltman 1996) and continuation-based semantics (de Groote 2006), can also be used; we aim to do this in the future.

Compositional vector semantics is our goal, but the nature of this paper is theoretical. So we shall not propose – from the armchair so to speak – concrete representations of contexts and updates and a set of concrete vector composition operations for combining phrases, or concrete matrices or cubes that embody them. We thus leave exhaustive empirical evaluation of our model to future work, but show, by means of examples, how the notion of “admittance of sentences by contexts” from the context update logic of Heim (1983) and Karttunen (1974) can be applied to develop a relationship between matrix and cube contexts and sentences, and how this notion can be extended from a usual Boolean relation to one which has degrees, based on the notion of degrees of similarity between words. As this notion resembles that of “contextual entailment” between corpora and sentences, we review the current entailment datasets that are mainstream in distributional semantics and discuss how they can or cannot be applied to test this notion, but leave experimental evaluation to future work.

The lambda calculus approach we use is based on Lambda Grammars (Muskens 2001, 2003), which were independently introduced as Abstract Categorical Grammars (ACGs) in de Groote (2001). The theory developed here, however, can be based on any syntax-semantics interface that works with a lambda calculus semantics – our approach is agnostic as to the choice of a syntactic theory.

This paper is the journal version of our previous short paper (Muskens and Sadrzadeh 2016a) and extended abstract (Muskens and Sadrzadeh 2016b).

Lambda Grammars (Muskens 2001, 2003) were independently introduced as Abstract Categorical Grammars (ACGs, de Groote 2001). An ACG generates two languages, an *abstract* language and an *object* language. The abstract language will simply consist of all linear lambda terms (each lambda binder binds exactly one variable occurrence) over a given vocabulary typed with *abstract types*. The object language has its own vocabulary and its own types. We give some basic definitions here, assuming familiarity with the simply typed λ -calculus.

If \mathcal{B} is some set of basic types, we write $TYP(\mathcal{B})$ for the smallest set containing \mathcal{B} such that $(\alpha\beta) \in TYP(\mathcal{B})$ whenever $\alpha, \beta \in TYP(\mathcal{B})$. Let \mathcal{B}_1 and \mathcal{B}_2 be sets of basic types. A function η from $TYP(\mathcal{B}_1)$ to $TYP(\mathcal{B}_2)$ is said to be a *type homomorphism* if $\eta(\alpha\beta) = (\eta(\alpha)\eta(\beta))$, for all $\alpha, \beta \in TYP(\mathcal{B}_1)$. It is clear that a type homomorphism η with domain $TYP(\mathcal{B})$ is completely determined by the values of η for types $\alpha \in \mathcal{B}$.

Let us look at an example of a type homomorphism that can be used to provide a language fragment with a classical Montague-like meaning. Let $\mathcal{B}_1 = \{D, N, S\}$ (D stands for determiner phrases, N for nominal phrases, S for sentences), let $\mathcal{B}_2 = \{e, s, t\}$ (e is for entities, s for worlds, and t for truth-values), and let h_0 be defined by: $h_0(D) = e$, $h_0(N) = est$,³ and $h_0(S) = st$. Then the types in the second column of Table 1 have images under h_0 as given in the fourth column. Additional information about the conventions used in Table 1 is given in a footnote.⁴

We now define the notion of *term homomorphism*. If C is some set of typed constants, we write $\Lambda(C)$ for the set of all lambda terms with constants only from C . The set of *linear* lambda terms over C is denoted by $\Lambda_0(C)$. Let C_1 be a set of constants typed by types from $TYP(\mathcal{B}_1)$ and let C_2 be a set of constants typed by types from $TYP(\mathcal{B}_2)$. A function $\vartheta : \Lambda(C_1) \rightarrow \Lambda(C_2)$ is a *term homomorphism based on η* if $\eta : TYP(\mathcal{B}_1) \rightarrow TYP(\mathcal{B}_2)$ is a type homomorphism and, whenever $M \in \Lambda(C_1)$:

³ Association in types is to the right and outer parentheses are omitted; so *est* is short for $(e(st))$, arguably a good type for *predicates*.

⁴ In Table 1, p is a variable of type *st*, while x is of type *e*. The variables w and w' are of type *s*, and P and P' are of type *est*. The constant K of type *ess* denotes the epistemic accessibility relation.

Table 1:
An Abstract Categorical
Grammar / Lambda
Grammar connecting
abstract terms with
Montague-like meanings

constant c	type τ	$H_0(c)$	$h_0(\tau)$
woman	N	<i>woman</i>	<i>est</i>
man	N	<i>man</i>	<i>est</i>
tall	NN	<i>tall</i>	$(est)est$
smokes	DS	<i>smoke</i>	<i>est</i>
loves	DDS	<i>love</i>	<i>eest</i>
knows	SDS	$\lambda p \lambda x \lambda w. \forall w' (Kxww' \rightarrow pw')$	$(st)est$
every	$N(DS)S$	$\lambda P' \lambda P \lambda w. \forall x (P'xw \rightarrow Pxw)$	$(est)(est)st$
a	$N(DS)S$	$\lambda P' \lambda P \lambda w. \exists x (P'xw \wedge Pxw)$	$(est)(est)st$

- $\vartheta(M)$ is a term of type $\eta(\tau)$, if M is a constant of type τ ;
- $\vartheta(M)$ is the n -th variable of type $\eta(\tau)$, if M is the n -th variable of type τ ;
- $\vartheta(M) = (\vartheta(A)\vartheta(B))$, if $M \equiv (AB)$;
- $\vartheta(M) = \lambda y. \vartheta(A)$, where $y = \vartheta(x)$, if $M \equiv (\lambda x. A)$.

Note that this implies that $\vartheta(M)$ is a term of type $\eta(\tau)$, if M is of type τ .

Clearly, a term homomorphism ϑ with domain $\Lambda(C)$ is completely determined by the values $\vartheta(c)$ for $c \in C$. This continues to hold if we restrict the domain to the set of linear lambda terms $\Lambda_0(C)$. In order to show how this mechanism can be used, let us continue with the same example. Consider the (abstract) constants in the first column of Table 1, typed by the (abstract) types in the second column. We can now define a term homomorphism H_0 by sending the constants in the first column to their images in the third column, making sure that these have types as in the fourth column. Since H_0 is assumed to be a type homomorphism, *all* lambda terms over the constants in the first column will now automatically have images under H_0 . For example, H_0 sends the abstract term:⁵

$$((a \text{ woman})\lambda\xi((\text{every man})(\text{loves } \xi)))$$

⁵We use the standard notation of lambda terms. The application of M to N is written as (MN) (not as $M(N)$) and lambda abstractions are of the form $(\lambda X. A)$. The usual redundancy rules for parentheses apply, but will often not be used in abstract terms, in order to emphasise their closeness to linguistic expressions. In some cases, to improve clarity, we will bend the rules and write $M(N_1, \dots, N_n)$ for $(MN_1 \dots N_n)$ or $A \wedge B$ for $\wedge AB$, for example.

(in which ξ is of type D), to a term $\beta\eta$ -equivalent with:

$$\lambda w \exists y (woman\ yw \wedge \forall x (man\ xw \rightarrow love\ yxw)) .$$

This term denotes the set of worlds in which some specific woman is loved by all men.

This example sends abstract terms to translations that are close to those of Montague (1974). While such translations obviously will not serve as *vector* semantics, we will show in the next sections that it is possible to alter the object language while retaining the general translation mechanism. For more information about the procedure of obtaining an object language from an abstract language, see de Groote (2001) and Muskens (2003, 2010).

3 A STATIC VECTOR SEMANTICS

3.1 *Vector interpretations for the object language*

In order to provide an interpretation of our object language, the type theory used must be able to talk about vectors over some field, for which we choose the reals. We need a basic object type R such that, in all interpretations under consideration, the domain D_R of type R is equal to or ‘close enough’ to the set of reals \mathbb{R} , so that constants such as the following (of the types shown) have their usual interpretation:⁶

$$\begin{array}{ll} 0 & : \quad R \\ 1 & : \quad R \\ + & : \quad RRR \\ \cdot & : \quad RRR \\ < & : \quad RRt \end{array}$$

This can be done by imposing one of the sets of second-order axioms in Tarski (1965). Given these axioms, we have $D_R = \mathbb{R}$ in full models, whereas we have non-standard models under the Henkin interpretation (Henkin 1950).

Vectors can now be introduced as objects of type IR , where I is interpreted as some finite index set. Think of I as a set of words; if

⁶ Constants such as $+$, \cdot , and $<$ will be written between their arguments.

Table 2:
Vector types
and their abbreviations

Type	Math Abbreviation	Letter Abbreviation	Description
IR	(I^1R)	V	Vector
IIR	I^2R	M	Matrix
$IIIR$	I^3R	C	Cube
\vdots	\vdots		

a word is associated with a vector $v : IR$, v assigns a real number to each word, which gives information about the company the word keeps. Since IR will be used often, we will abbreviate it as V . Similarly, IIR , abbreviated as M , can be associated with the type of *matrices*, and $IIIR$, abbreviated as C , with the type of *cubes*, and so on (see Table 2). In this paper, we work with a single index type, but one may also consider cases with several index types, so that phrases of distinct categories can live in their own space.

We need a toolkit of functions combining vectors, matrices, cubes, etc. In the following definitions, r is of type R ; i , j , and k are of type I ; v and u are of type V ; m and c are of types M and C respectively; and indices are written as subscripts, so v_i is syntactic sugar for vi .

$$\begin{aligned}
 * &:= \lambda rvi.r \cdot v_i : RVV \\
 \boxplus &:= \lambda vui.v_i + u_i : VVV \\
 \odot &:= \lambda vui.v_i \cdot u_i : VVV \\
 \times_1 &:= \lambda mvi.\sum_j m_{ij} \cdot v_j : MVV \\
 \times_2 &:= \lambda cvij.\sum_k m_{ijk} \cdot v_k : CVM \\
 \langle \cdot | \cdot \rangle &:= \lambda uv.\sum_i u_i + v_i : VVR
 \end{aligned}$$

The reader will recognise $*$ as scalar product, \boxplus as pointwise addition, \odot as pointwise multiplication, \times_1 and \times_2 as matrix-vector and cube-vector multiplication, and $\langle \cdot | \cdot \rangle$ as the dot product. One can also consider further operations, such as various *rotation* operations with type $\rho : VVV$.

3.2 Abstract types and type and term homomorphisms

Let us assume again that our basic abstract types are D for determiner phrases, S for sentences, and N for nominal phrases. But this time our

type and term homomorphisms will be chosen in a different way from that used in Section 2. A very simple type homomorphism h can be defined by:

$$h(D) = h(S) = h(N) = V .$$

So h assigns vectors to determiners, nominal phrases, and sentences. There are other possibilities for the range of h and, in the following section, we will sketch a more elaborate assignment in which a running context is used. The above simple h is chosen for the expository purposes of this section.

In Table 3, we again provide abstract constants in the first column and their abstract types in the second column; h assigns to these the object types in the fourth column. Here, Z is a variable of type VV , and v and u are of type V . As an example, consider the constant *woman*; it has the abstract type N , and a term homomorphic image *woman*, which is assigned the type V by h . We say that the translation of *woman* is of type V . Similarly, the translations of *tall* and *smoke* are of type VV , *love* and *know* are of type VVV , and those of *every* and *a* are of type VV . The term homomorphism H is defined by letting its value for any abstract constant in the first column be the corresponding object term in the third column. Using this table, we automatically obtain homomorphic images of any lambda term over the constants. But now our previous example term:⁷

$$((a \text{ woman})\lambda\xi((\text{every man})(\text{loves } \xi)))$$

is sent to a term that is $\beta\eta$ equivalent with:

$$(\text{love } \times_2 (a \times_1 \text{ woman})) \times_1 (\text{every } \times_1 \text{ man}) .$$

In Table 3, nominal phrases like *woman* are represented by vectors, adjectives and intransitive verbs like *tall* and *smoke* by matrices, and transitive verbs (*love*) by cubes, as are constants like *know*. Generalised quantifiers are functions that take vectors to vectors. The composition operations used (\times_1 and \times_2) are cube-vector and matrix-vector instances of tensor contraction. There is still much debate as to

⁷ The entry for *man* is no longer present in Table 3. But *man* can be treated in full analogy to *woman*. In further examples we will also use constants whose entries can easily be guessed.

Table 3:
A fragment of static vector
semantics. Abstract
constants c are typed with
abstract types τ and their
term homomorphic images
 $H(c)$ typed by $h(\tau)$

c	τ	$H(c)$	$h(\tau)$
woman	N	woman	V
tall	NN	$\lambda v.(\text{tall} \times_1 v)$	VV
smokes	DS	$\lambda v.(\text{smoke} \times_1 v)$	VV
loves	DDS	$\lambda uv.(\text{love} \times_2 u) \times_1 v$	VVV
knows	SDS	$\lambda uv.(\text{know} \times_2 u) \times_1 v$	VVV
every	$N(DS)S$	$\lambda vZ.Z(\text{every} \times_1 v)$	$V(VV)V$
a	$N(DS)S$	$\lambda vZ.Z(a \times_1 v)$	$V(VV)V$

Table 4:
Term homomorphic
images $H(c)$ for pointwise
addition and
multiplication, and matrix
multiplication as
composition operations

Addition	Multiplication	Matrix Multiplication
$H(c)$	$H(c)$	$H(c)$
woman	woman	woman
$\lambda v.(\text{tall} \boxplus v)$	$\lambda v.(\text{tall} \odot v)$	$\lambda v.(\text{tall} \times_1 v)$
$\lambda v.(\text{smoke} \boxplus v)$	$\lambda v.(\text{smoke} \odot v)$	$\lambda v.(\text{smoke} \times_1 v)$
$\lambda uv.(\text{love} \boxplus u) \boxplus v$	$\lambda uv.(\text{love} \odot u) \odot v$	$\lambda uv.(\text{love} \times_2 u) \times_1 v$
$\lambda uv.(\text{know} \boxplus u) \boxplus v$	$\lambda uv.(\text{know} \odot u) \odot v$	$\lambda uv.(\text{know} \times_2 u) \times_1 v$
$\lambda vZ.Z(\text{every} \boxplus v)$	$\lambda vZ.Z(\text{every} \odot v)$	$\lambda vZ.Z(\text{every} \times_1 v)$
$\lambda vZ.Z(a \boxplus v)$	$\lambda vZ.Z(a \odot v)$	$\lambda vZ.Z(a \times_1 v)$

the best operations for composing vectors. Mitchell and Lapata (2010) consider pointwise addition and multiplication of vectors, while matrix multiplication is used in Baroni and Zamparelli (2010). Such operations are available to our theory. The table for these will have a different $H(c)$ column and will be the same in all other columns. The $H(c)$ columns for these models are given in Table 4.⁸

In this paper, we will not choose between these operations. Instead, we will explore how to combine such functions once an initial set has been established (and validated empirically). Functions in the initial set will typically combine vector meanings of adjacent phrases. Like Baroni *et al.* (2014), who provide an excellent introduction to and review of compositional vector semantics, our aim has been to pro-

⁸In Table 4, we use the same typographical conventions for variables as in Table 3, while, in its first two alternative columns, all constants written in sans serif are taken to be of type V . In its third column, the types of these constants (and in fact the whole column) are as in Table 3 again.

pose a general theory that also includes dependencies between non-adjacent phrases, e.g., in topicalisation or relative clause formation.

4 DYNAMIC VECTOR SEMANTICS WITH CONTEXT CHANGE POTENTIALS

4.1 *Heim's files and distributional contexts*

Heim describes her contexts as files that have some kind of information written on (or in) them. Context changes are operations that update these files, e.g. by adding or deleting information from the files. Formally, a context is taken to be a set of sequence-world pairs in which the sequences come from some domain \mathcal{D}_I of individuals, as follows:

$$ctx \subseteq \{(g, w) \mid g: \mathbb{N} \rightarrow \mathcal{D}_I, w \text{ a possible world}\}$$

We follow Heim (1983) here in letting the sequences in her sequence-world-pairs be infinite, although they are best thought of as finite.

Sentence meanings are *context change potentials* (CCPs) in Heim's work, functions from contexts to contexts – given any context, a sentence will transform it into a new context. In particular, a sentence S comes provided with a sequence of instructions that, given any context ctx , updates its information so that a new context results, denoted as:

$$ctx + S$$

The sequence of instructions that brings about this update is derived compositionally from the constituents of S .

In distributional semantics, contexts are words somehow related to each other via their patterns of use, e.g. by co-occurring in a neighbourhood word window of a fixed size, or via a dependency relation. In practice, one builds a context matrix M over \mathbb{R}^2 , with rows and columns labelled by words from a vocabulary Σ , and with entries taking values from \mathbb{R} (for a full description see Rubenstein and Goode-nough 1965). Thus, M can be seen as the set of its vectors:

$$\{\vec{v} \mid \vec{v}: \Sigma \rightarrow \mathbb{R}\},$$

where each \vec{v} is a row or column in M .

If we take Heim’s domain of individuals \mathcal{D}_I to be the vocabulary of a distributional model of meaning, that is $\mathcal{D}_I := \Sigma$, then a context matrix can be seen as a *quantized* version of a Heim context:

$$\{(\vec{g}, w) \mid \vec{g} : \Sigma \rightarrow \mathbb{R}, w \text{ a possible world}\}.$$

Thus a distributional context matrix is obtainable by endowing Heim’s contexts with \mathbb{R} . In other words, we are assuming not only that a file has a set of individuals, but also that these individuals take some kind of values, e.g. from reals.

The role of possible worlds in distributional semantics is arguable, as vectors retrieved from a corpus are not naturally truth conditional. Keeping the possible worlds in the picture provides a mechanism to assign a proposition to a distributional vector by other means and can become very useful. We leave working with possible worlds to future studies and in this paper only work with sets of vectors as our contexts, as follows:

$$ctx \subseteq \{\vec{g} \mid \vec{g} : \Sigma \rightarrow \mathbb{R}, g \in M\}.$$

Distributional versions of CCPs can be defined based on Heim’s intuitions and definitions. In what follows, we show how these instructions let contexts thread through vectorial semantics in a compositional manner.

4.2 *Dynamic type and term homomorphisms and their interpretations*

On the set of basic abstract types D, S, N , a *dynamic* type homomorphism ρ that takes into account the contexts of words is defined as follows:

$$\rho(N) = (VU)U, \quad \rho(D) = V, \quad \rho(S) = U.$$

Here, sentences are treated as *context change potentials*. They update contexts, and we therefore assign the type U (for ‘update’) to them. A context can be a matrix or a cube, so it can be of type I^2R or I^3R . A sentence can then be of type $(I^2R)(I^2R)$ or $(I^3R)(I^3R)$. We have previously abbreviated IR to V , I^2R to M , and I^3R to C . The sentence type then becomes MM or CC . The notation U can abbreviate either, depending on whether we choose to model contexts as cubes or as matrices. The concrete semantics obtained by each choice will be discussed in more detail in Section 5 and Section 6, respectively.

Update functions are presented in Table 5, where ρ is a type homomorphism, i.e. $\rho(AB) = \rho(A)\rho(B)$. Here, Z is a variable of type VU , Q is of type $(VU)U$, v of type V , c of type M , and p and q are of type U . The functions F , G , I , and J are explained in the following paragraphs. In the schematic entry for *and*, we write $\rho(\bar{a})$ for $\rho(\alpha_1) \cdots \rho(\alpha_n)$, if $\bar{a} = \alpha_1 \cdots \alpha_n$. Simple words such as names, nouns, adjectives, and verbs are first assigned vectors, denoted by constants such as *anna*, *woman*, *tall*, and *smoke* (all of type V). These are then used by the typed lambda calculus given via $H(a)$, in the third column, to build certain functions, which will act as the meanings of these words in context. The object types assigned by ρ are as follows:

Type of nouns	: $(VU)U$
Type of adjectives	: $((VU)U)(VU)U$
Type of intransitive verbs	: VU
Type of transitive verbs	: VVU

The function Z updates the context of proper names and nouns based on their vectors e.g. *anna* and *woman*. These are essentially treated as vectors of type V , but, since they must be made capable of dynamic behaviour, they are ‘lifted’ to the higher type $(VU)U$.

The function F of an adjective takes a vector for the adjective, e.g. *tall*, a vector for its argument, e.g. v , and a vector for its context, e.g. c , then updates the context, e.g. as in $F(\text{tall}, v, c)$. The output of this function is then lifted to the higher type, i.e. $((VU)U)((VU)U)$, via the functions Z and Q , respectively.

a	τ	$H(a)$	$\rho(\tau)$
<i>Anna</i>	$(DS)S$	$\lambda Z.Z(\text{anna})$	$(VU)U$
<i>woman</i>	N	$\lambda Z.Z(\text{woman})$	$(VU)U$
<i>tall</i>	NN	$\lambda QZ.Q(\lambda vc.ZvF(\text{tall}, v, c))$	$((VU)U)(VU)U$
<i>smokes</i>	DS	$\lambda vc.G(\text{smoke}, v, c)$	VU
<i>loves</i>	DDS	$\lambda uvc.I(\text{love}, u, v, c)$	VVU
<i>knows</i>	SDS	$\lambda pvc.pJ(\text{know}, v, c)$	UVU
<i>every</i>	$N(DS)S$	$\lambda Q.Q$	$((VU)U)(VU)U$
<i>who</i>	$(DS)NN$	$\lambda Z'QZ.Q(\lambda vc.Zv(QZ'c))$	$(VU)((VU)U)(VU)U$
<i>and</i>	$(\bar{a}S)(\bar{a}S)(\bar{a}S)$	$\lambda R'\lambda R\lambda \bar{X}\lambda c.R'\bar{X}(R\bar{X}c)$	$(\rho(\bar{a})U)(\rho(\bar{a})U)(\rho(\bar{a})U)$

Table 5:
A fragment of
dynamic vector
semantics.
Abstract
constants a
typed with
abstract types τ
and their term
homomorphic
images $H(a)$
typed by $\rho(\tau)$

Functions G and I update contexts of verbs; they take a vector for the verb as well as a vector for each of its arguments, plus an input context, and then return a context as their output. So, the function G takes a vector for an intransitive verb, e.g. *smoke*, a vector v for its subject, plus a context c , and returns a modified context $G(\text{smoke}, v, c)$. The function I takes a vector for a transitive verb, a vector for its subject, a vector for its object, and a context, and returns a context.

The meanings of function words, such as conjunctions, relative pronouns, and quantifiers, will not (necessarily) be identified with vectors. The type of the quantifier *every* is $((VU)U)(VU)U$, where its noun argument has the required ‘quantifier’ type $(VU)U$. The lambda calculus entry for ‘every’, $\lambda Q.Q$, is the identity function; it takes a Q and then spits it out again. The alternative would be to have an entry similar to that of ‘tall’, but this would not make much sense. Content words, and not function words, seem to be important in a distributional setting.

The word *and* is treated as a generalised form of function composition. Its entry is schematic, as *and* does not only conjoin sentences, but also other phrases of any category. So the type of the abstract constant connected with the word is $(\bar{a}S)(\bar{a}S)(\bar{a}S)$, in which \bar{a} can be any sequence of abstract types. Ignoring this generalisation for the moment, we obtain SSS as the abstract type for sentence conjunction, with a corresponding object type UUU , and meaning $\lambda p q c.p(qc)$, which is just function composition. This is defined such that the context updated by *and*’s left argument will be further updated by its right argument. So ‘Sally smokes and John eats bananas’ will, given an initial context c , first update c to $G(\text{Sally}, \text{smoke}, c)$, which is a context, and then update further with ‘John eats bananas’ to $I(\text{eat}, \text{John}, \text{bananas}, G(\text{smoke}, \text{Sally}, c))$. This treatment of *and* is easily extended to coordination in all categories. For example, the reader may check that *and admires loves* (which corresponds to *loves and admires*) has $\lambda u v c.I(\text{admire}, u, v, I(\text{love}, u, v, c))$ as its homomorphic image.

The update instructions pass through phrases and sentences compositionally. The sentence *every tall woman smokes*, for example, will be associated with the following lambda expression:

$$(((\text{every } (\text{tall woman})) \text{ smokes}))$$

This in its turn has a term homomorphic image that is β -equivalent with the following:

$$\lambda c.G(\text{smoke}, \text{woman}, F(\text{tall}, \text{woman}, c))$$

which describes a distributional context update for it. This term describes an initial update of the context c according to the rule for the constant *tall*, and then a second update according to the rule for the constant *smokes*. As a result of these, the value entries at the crossings of $\langle \text{tall}, \text{woman} \rangle$ and $\langle \text{woman}, \text{smokes} \rangle$ are increased. Much longer chains of context updates can be ‘threaded’ in this way.

In the following, we give some examples. In each case, sentence *a* is followed by an abstract term in *b*, thus capturing its syntactic structure. The update potential that follows in *c* is the homomorphic image of this abstract term.

- (1) a. Sue loves and admires a stockbroker.
 b. $(\text{a stockbroker})\lambda\xi.\text{Sue}(\text{and admires loves } \xi)$
 c. $\lambda c.I(\text{admire}, \text{stockbroker}, \text{sue}, I(\text{love}, \text{stockbroker}, \text{sue}, c))$
- (2) a. Bill admires but Anna despises every cop.
 b. (every cop)
 $(\lambda\xi.\text{and}(\text{Anna}(\text{despise } \xi))(\text{Bill}(\text{admire } \xi)))$
 c. $\lambda c.I(\text{despise}, \text{cop}, \text{anna}, I(\text{admire}, \text{cop}, \text{bill}, c))$
- (3) a. The witch who Bill claims Anna saw disappeared.
 b. $\text{the}(\text{who}(\lambda\xi.\text{Bill}(\text{claims}(\text{Anna}(\text{saw } \xi))))\text{witch})$
 disappears
 c. $\lambda c.G(\text{disappear}, \text{witch}, I(\text{see}, \text{witch}, \text{anna}, J(\text{claim}, \text{bill}, c)))$

5 CO-OCCURENCE MATRIX CONTEXT AND ITS UPDATE

In this section, we assume that our contexts are the co-occurrence matrices of distributional semantics (Rubenstein and Goodenough 1965). Given a corpus of texts, a co-occurrence matrix has, for each of its entries, the degree of co-occurrence between that word and neighbouring words. The neighbourhood is usually a window of k words on either side of the word. The update type U associated with sentences

will thus take the form $(I^2R)(I^2R)$, abbreviated to MM . That is, a sentence will take a co-occurrence matrix as input, update it with new entries, and return the updated matrix as output.

Since we are working with co-occurrence matrices, the updates simply increase the degrees of co-occurrence between the labelling words of the rows and columns of the matrix. In this paper, to keep things simple, we work on a co-occurrence matrix with raw co-occurrence numbers as entries. In this case, the update functions just add 1 to each entry at each update step. This may be extended to (or replaced with) logarithmic probabilistic entries, such as Pointwise Mutual Information (PMI) or its positive or smoothed version PPMI, $PPMI_\alpha$, in which case the update functions have to recalculate these weighting schemes at each step (for an example, see Table 6). The cells whose entries are increased are chosen according to the grammatical roles of the labelling words. These are implemented in the functions F, G, I, J , which apply the updates to each word in the sentence. Updates are compositional, i.e. they can be applied compositionally to the words within a sentence. This is evident as the updates induced by words in a sentence are based on the grammatical roles of those words, which act as glue.

More formally, the object terms corresponding to a word a update a context matrix c with the information in a and the information in the vectors of arguments u, v, \dots of a . The result is a new context matrix c' , with different entry values, depicted below:

$$\begin{pmatrix} m_{11} & \cdots & m_{1k} \\ m_{21} & \cdots & m_{2k} \\ \vdots & & \\ m_{n1} & \cdots & m_{nk} \end{pmatrix} + \langle a, u, v, \dots \rangle = \begin{pmatrix} m'_{11} & \cdots & m'_{1k} \\ m'_{21} & \cdots & m'_{2k} \\ \vdots & & \\ m'_{n1} & \cdots & m'_{nk} \end{pmatrix}$$

The m_{ij} and m'_{ij} entries are described as follows:

- The function $G(\text{smoke}, v, c)$ increases the entry value of m_{ij} in c by 1 in case i is the index of smoke and j is the index of its subject v . In all other cases $m'_{ij} = m_{ij}$.
- The function $I(\text{love}, u, v, c)$ increases the entry values of m_{ij} , m_{jk} , and m_{ik} in c by 1 in case i is the index of loves, j is the index of its subject u , and k the index of its object v . In all other cases $m'_{ij} = m_{ij}$.

- The function $F(\text{tall}, v, c)$ increases the entry value of m_{ij} in c by 1 in case i is the index of tall and j is the index of its modified noun v . In all other cases $m'_{ij} = m_{ij}$. The entry for *tall* in Table 1 uses this function, but allows for further update of context.
- The function $J(\text{know}, v, c)$ increases the entry value of m_{ij} in c by 1 in case i is the index of know and j is the index of its subject v . In all other cases $m'_{ij} = m_{ij}$. The updated matrix becomes the input for further update (by the context change potential of the sentence that is known).

As an example, consider the co-occurrence matrices depicted in Figure 1. The initial matrix is a snapshot just before a series of updates are applied. The rationale of this example is as follows: Anna is a woman and so this word is not frequently found in the context man; as a result, it has a low value of 100 at that entry; Anna loves cats (and has some herself), so the entry at the context cat is 700; she loves other things, such as smoking, and so there is a substantial

		1	2	3	4	5
		man	cat	loves	fears	sleeps
1	Anna	100	700	800	500	400
2	woman	500	650	750	750	600
3	tall	300	50	500	400	400
4	smokes	400	50	600	600	200
5	loves	350	250	ε	600	500
6	knows	300	50	200	250	270

Figure 1:
An example of updates
by functions F, G, I, J on
a co-occurrence matrix

a series of updates by
 $\xrightarrow{F, G, I, \text{ and } J}$

		1	2	3	4	5
		man	cat	loves	fears	sleeps
1	Anna	100	700	800	500	400
2	woman	500	650	750	750	600
3	tall	650	50	500	400	400
4	smokes	700	50	600	600	200
5	loves	550	750	ε	600	500
6	knows	600	250	450	510	700

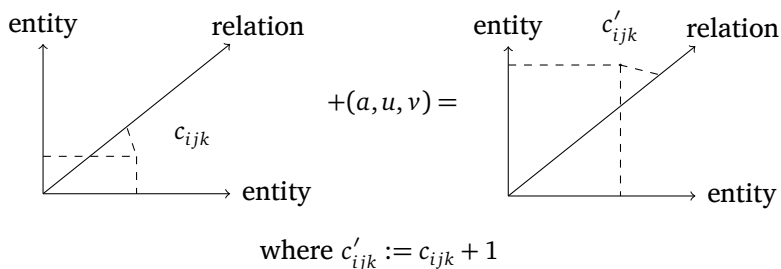
entry at the context loves; and so on. The entries of the other words, i.e. tall, smokes, loves, knows, are also initialised to their distributional co-occurrence matrix vectors. When an entry c_{ij} corresponds to the same two words, e.g. when i and j are both love, as in the initial matrix in Figure 1, we use ϵ to indicate a predefined fixed value.

The intransitive verb *smokes* updates the initial matrix in Figure 1, via the function G at the entries c_{4j} . Here, in principle, j can be 1 and 2, as both man and cat, in their singular or plural forms, could have occurred as subjects of *smokes* in the corpus. Assuming that cats do not smoke and that a reasonable number of men do, a series of, for instance, 300 occurrences of *smokes* with the subject man, updates this entry and raises its value from 400 to 700. Similarly, the adjective *tall* updates the entries of the c_{3j} cells of the matrix via the function F , where j can in principle be 1 and 2, but since cats are not usually tall, it only updates c_{31} . Again, a series of, for example, 350 occurrences of the adjective *tall* as the modifier of man would raise this number from 300 to 650. The case for *loves* and function I is similar. For *knows*, men know cats love mice, and love to play and be stroked, etc.; they know that cats fear water and objects such as vacuum cleaners, and that they sleep a lot. As a result, the values of all of the entries in row 6, that is $c_{61}, c_{62}, c_{63}, c_{64}$ and c_{65} , will be updated by function J , for instance, to the values in the updated matrix.

6 ENTITY RELATION CUBE CONTEXT AND ITS UPDATE

A corpus of texts can be seen as a sequence of lexical items occurring in the vicinity of each other, and can thus be transformed into a co-occurrence matrix. It can also be seen as a sequence of entities

Figure 2:
Updates of
entries in an
entity relation
cube



related to each other via predicate-argument structures, which can therefore be transformed into an entity relation graph. This can be modelled in our setting by taking the contexts to be cubes, thus setting S to have the update type $U = (I^3R)(I^3R)$, abbreviated to CC . The entity relation graph approach needs a more costly preprocessing of the corpus, but it is useful for a systematic treatment of logical words such as negation and quantification, as well as coordination.

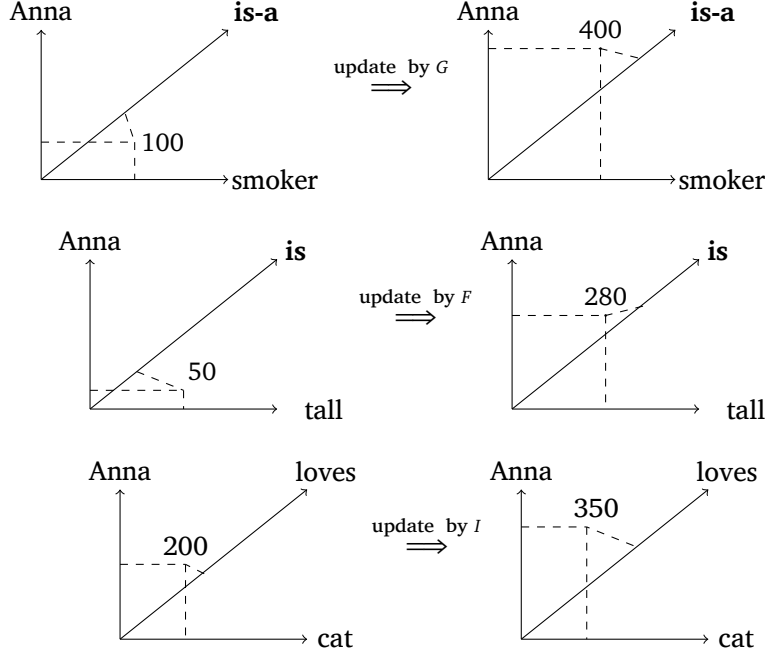
An entity relation graph can be derived from a variety of resources: a semantic network of concepts, a knowledge base such as WordNet or FrameNet. We work with entities and relations extracted from text. Creating such graphs from text corpora automatically has been the subject of much recent research (see, for example, Yao *et al.* 2012, and Riedel *et al.* 2010, for a direct approach; see also Kambhatla 2004, and Poon and Domingos 2009, for an approach based on semantic parsing). The elements of an entity relation graph are argument-relation-argument triples, sometimes referred to as *relation paths* (Yao *et al.* 2012). Similarly to Lewis and Steedman (2013), we position ourselves in a binary version of the world, where all relations are binary; we turn unary relations into binary ones using the is-a predicate.

Similar to the matrix case, the object terms corresponding to a constant a update a context cube c with the information in a and the information in the vectors of arguments of a . The result is a new context cube c' , with entry values greater than or equal to the originals, as depicted in Figure 2.

The c_{ijk} and c'_{ijk} entries are similar to those in the matrix case, for example:

- The function $G(\text{smoke}, v, c)$ increases the entry value c_{ijk} of c in case i is the fixed index of is-a, j is the index of smoker, and k is the index of v , the subject of smoke. Other entry values remain unchanged.
- The function $F(\text{tall}, v, c)$ increases the entry value c_{ijk} of c in case i is the fixed index of is, j is the index of tall, and k is the index of v , the modified noun. Other entry values remain unchanged.
- The function denoted $I(\text{love}, u, v, c)$ increases the entry value c_{ijk} of c in case i is the index of love, j is the index of its subject u , and k is the index of its object v . Other entry values remain unchanged.

Figure 3:
An example of
updates by
functions F, G, I
on an
entity-relation
cube



As an example, consider the series of updates depicted in Figure 3. Relative pronouns such as *who* update the entry corresponding to the head of the relative clause and the rest of the clause. For example, in the clause ‘the man who went home’, we update c_{ijk} for i the index of ‘man’ as subject of the verb ‘went’ with index j and its object ‘home’ with index k (see also Section 4, example (3)). Propositional attitudes such as ‘know’ update the entry value of c_{ijk} for i the index of their subject, j the index of themselves, and k the index of their proposition. For instance, in the sentence ‘John knows Mary slept’, we update the entry value for ‘John’, ‘know’ and the proposition ‘Mary slept’. The conjunctive *and* is modelled as before (in Section 4, compare examples (1) and (2)).

Negation can be modelled by providing an abstract term not of type SS with a term homomorphic image $\lambda pc.c \dot{-} pc$ of type UU , where $\dot{-}$ is pointwise subtraction of cubes (i.e. $\lambda cc'ijk.cijk - c'ijk$). The operation denoted by this term first updates the context with the non-negated sentence, after which the result is subtracted from the context again.

7 LOGIC FOR CONTEXT CHANGE POTENTIALS

The logic for sentences as context change potentials has the following syntax.

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi$$

Disjunction and implication operations are defined using the De Morgan duality.

$$\begin{aligned}\phi \vee \psi &:= \neg(\neg\phi \wedge \neg\psi) \\ \phi \rightarrow \psi &:= \neg\phi \vee \psi\end{aligned}$$

This logic is the propositional fragment of the logic of context change potentials, presented in Muskens *et al.* (1997), based on the ideas of Heim (1983). Heim extends Karttunen's theory of presuppositions (Karttunen 1974) and defines the context change potential of a sentence as a function of the context change potentials of its parts, an idea that leads to the development of the above logic. The logic we consider here is the same logic but without the presupposition operation.

We refer to the language of this logic as \mathcal{L}_{ccp} . For a context c , a context change potential is defined as follows.

$$\begin{aligned}\|p\|(c) &:= c + \|p\| \\ \|\neg\phi\|(c) &:= c - \|\phi\|(c) \\ \|\phi \wedge \psi\| &:= \|\psi\|(\|\phi\|(c))\end{aligned}$$

It is easy to verify that:

$$\begin{aligned}\|\phi \vee \psi\| &= \|\psi\|(c) - \|\psi\|(\|\phi\|(c)) \\ \|\phi \rightarrow \psi\|(c) &= c - (\|\phi\|(c) - \|\psi\|(\|\phi\|(c))) .\end{aligned}$$

Here, $\|\phi\|$ is the context change potential of ϕ and a function from contexts to contexts. Whereas, for Heim, both contexts and context change potentials of atomic sentences $\|p\|$ are sets of valuations, for us, contexts are co-occurrence matrices or entity relation cubes, and context change potentials of atomic sentences are vectors. Thus, where the context change potential operation (Heim 1983) simply takes the intersection of a context and a context change potential $c \cap \|p\|$, we perform an operation that acts on matrices/cubes rather than sets. We use the update operation of term homomorphisms, defined in the previous sections, and define a context change potential as follows.

Definition 1. For S a sentence in \mathcal{L}_{ccp} , $\|S\|$ its context change potential, $H(S)$ the term homomorphic image of S , and c a co-occurrence matrix or an entity relation cube, we define:

$$\begin{aligned}\|S\|(c) &:= c +' H(S) \\ c - H(S) &:= (c +' H(S))^{-1},\end{aligned}$$

for $+'$ the update operation defined on term homomorphisms and $-'$ its inverse, defined as follows for matrices.

$$\begin{aligned}\begin{pmatrix} m_{11} & \cdots & m_{1k} \\ m_{21} & \cdots & m_{2k} \\ \vdots & & \\ m_{n1} & \cdots & m_{nk} \end{pmatrix} +' \langle a, u, v, \dots \rangle &= \begin{pmatrix} m'_{11} & \cdots & m'_{1k} \\ m'_{21} & \cdots & m'_{2k} \\ \vdots & & \\ m'_{n1} & \cdots & m'_{nk} \end{pmatrix} \\ \text{for } m'_{ij} &:= \begin{cases} 1 & m_{ij} = 1 \\ 1 & m_{ij} = 0 \end{cases} \\ \begin{pmatrix} m_{11} & \cdots & m_{1k} \\ m_{21} & \cdots & m_{2k} \\ \vdots & & \\ m_{n1} & \cdots & m_{nk} \end{pmatrix} -' \langle a, u, v, \dots \rangle &= \begin{pmatrix} m'_{11} & \cdots & m'_{1k} \\ m'_{21} & \cdots & m'_{2k} \\ \vdots & & \\ m'_{n1} & \cdots & m'_{nk} \end{pmatrix} \\ \text{for } m'_{ij} &:= \begin{cases} 0 & m_{ij} = 1 \\ 0 & m_{ij} = 0 \end{cases}\end{aligned}$$

The definitions of $+'$ and $-'$ for cubes are similar.

The $+'$ operation updates the co-occurrence matrix in a binary fashion: if the entry m_{ij} of the matrix has already been updated and thus has value 1, then a succeeding update will not increase the value from 1 to 2 but will keep it as 1. Conversely, when the $-'$ operation acts on an entry m_{ij} which is already 0, it will not change its value, but if it acts on a non-zero m_{ij} , that is an m_{ij} which has value 1, it will decrease it to 0. The procedure is similar for cubes. The resulting matrices and cubes will have binary entries, that is, they will either be 1 or 0. A 1 indicates that at least one occurrence of the roles associated with the entries has previously been seen in the corpus; a 0 indicates that none has been seen or that a role and its negation have occurred.

Fixing a bijection between the elements $[1, n] \times [1, k]$ of our matrices and natural numbers $[1, n \times k]$ and between elements $[1, n] \times [1, k] \times [1, z]$ of the cubes and natural numbers $[1, n \times k \times z]$, one can show that $c +' H(S)$ is the table of a binary relation in the case of matrices, and of a ternary relation in the case of cubes. Those entries (i, j) of the matrices and (i, j, k) of the cubes that have a non-zero entry value are mapped to an element of the relation. An example of this isomorphism is shown below for a 2×2 matrix:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \mapsto \begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 1 & 0 \\ 2 & 1 & 1 \end{array} \quad \{(1, 1), (2, 1), (2, 2)\} .$$

These binary updates can be seen as providing a notion of ‘contextual truth’, that is, for example, a sentence S is true in a given a context c , whenever the update resulting from s is already included in the matrix or cube of its context, i.e. its update is one that does not change c .

As argued in Muskens *et al.* (1997), the semantics of this logic is dynamic, in the sense that the context change potential of a sequence of sentences is obtained by function composition, as follows:

$$\|S_1, \dots, S_n\|(c) := \|S_1\| \circ \dots \circ \|S_n\|(c) .$$

Using this dynamic semantics, it is straightforward to show that:

Proposition 1. *The context c corresponding to the sequence of sentences S_1, \dots, S_n , is the zero vector updated by that sequence of sentences:*

$$c = \|S_1, \dots, S_n\|(0) ,$$

where

$$\begin{aligned} \|S\|(c) &:= c + H(S) \\ c - H(S) &:= (c + H(S))^{-1} . \end{aligned}$$

In the case of co-occurrence matrices, c is the co-occurrence matrix and 0 is the zero matrix. In the case of entity relation cubes, c is the entity relation cube and 0 is the zero cube. We are using the usual real number addition and subtraction on the m_{ij} and c_{ijk} entries of the matrices and cubes:

$$\begin{aligned} m'_{ij} &:= m_{ij} + 1 & m'_{ij} &:= m_{ij} - 1 \\ c'_{ijk} &:= c_{ijk} + 1 & c'_{ijk} &:= c_{ijk} - 1 . \end{aligned}$$

We will refer to a sequence of sentences as a *corpus*.

8 ADMITTANCE OF SENTENCES BY CONTEXTS

The notion of *admittance of a sentence by a context* was developed by Karttunen (1974) for presuppositions, and extended by Heim (1983) for context change potentials. We here define it as follows, for c a context and ϕ a proposition of \mathcal{L}_{ccp} .

$$\text{context } c \text{ admits proposition } \phi \iff \|\phi\|(c) = c$$

We use this notion and develop a similar notion between a corpus and a sentence.

Definition 2. *A corpus admits a sentence iff the context c (a co-occurrence matrix or entity relation cube) built from it admits it.*

Consider the following corpus:

Cats and dogs are animals that sleep. Cats chase cats and mice. Dogs chase all animals. Cats like mice, but mice fear cats, since cats eat mice. Cats smell mice and mice run from cats.

It admits the following sentences:

Cats are animals.
Dogs are animals.
Cats chase cats.
Cats chase mice.
Dogs chase cats and dogs.

Note that this notion of admittance caters for monotonicity of inference. For instance, in the above example, from the sentences “Cats [and dogs] are animals [that sleep]” and “Dogs chase all animals”, we can infer that the context admits the sentence “Dogs chase cats”.

On the other hand, c does not admit the negation of the above, for example it does not admit:

(*) Dogs do not chase cats.
(*) Dogs do not chase dogs.

It also does not admit the negations of derivations of the above or negations of sentences of the corpus, for example, it does not admit:

(*) Cats are not animals.
(*) Dogs do not sleep.

The corpus misses a sentence asserting that mice are also animals. Hence, c does not admit the sentence ‘dogs chase mice’. Some other sentences that are not admitted by c are as follows:

- (*) Cats like dogs.
- (*) Cats eat dogs.
- (*) Dogs run from cats.
- (*) Dogs like mice.
- (*) Mice fear dogs.
- (*) Dogs eat mice.

One can argue that by binarizing the update operation and using $+$ and $-$ rather than the original $+$ and $-$, we are losing the full power of distributional semantics. It seems wasteful simply to record the presence or absence of co-occurrence, rather than build context matrices by counting co-occurrences. This can be overcome by working with a pair of contexts: a binarized one and a numerical one. The binarized context allows a notion of admittance to be defined as before, and the numerical one allows the use of numerical values, e.g. the degrees of similarity between words. The notion of word similarity used in distributional semantics is a direct consequence of the distributional hypothesis, where words that often occur in the same contexts have similar meanings (Firth 1957). Various formal notions have been used to measure the above degree of similarity; amongst the successful ones is the cosine of the angle between the vectors of the words. If the vectors are normalised to have length 1, which we shall assume, cosine becomes the same as the dot product of the vectors. One can then use these degrees of similarity to assign a numerical value to the admittance relation, e.g. as follows:

A pair of binary and numerical co-occurrence matrices c and c' admit a sentence s' with degree d , if c admits s , and s' is obtained from s by replacing a word w of s with a word w' such that w' has the same grammatical role in s' as w in s and the degree of similarity between w and w' is d , computed from the numerical entries of c' .

Here, c admits s , and if there is a word in s that is similar to another word w' , then if we replace w in s with w' (keeping the grammatical role that w had in s), the sentence resulting from this substitution,

Table 6: The normalised co-occurrence matrix built from the example corpus with the co-occurrence window taken to be occurrence within the same sentence

	1 animal	2 sleep	3 chase	4 like	5 fear	6 eat	7 smell	8 run
1- cats	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
2- mice	0	0	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
3- dogs	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0

i.e. s' , is also admitted by c , albeit with a degree equal to the degree of similarity between w and w' . This degree is computed using the numerical values recorded in c' . The above can be extended to the case where one replaces more than one word in s with words similar to them. Then the degree of entailment may be obtained by multiplying the degrees of similarity of the individually replaced words.

The normalised context matrix of our example corpus above is as in Table 6, where for simplicity the co-occurrence window is taken to be “occurrence within the same sentence”.

From the context matrix, one obtains the following degrees of similarity.

$$\begin{aligned}
 \cos(\text{cats}, \text{mice}) &= 6 \times \left(\frac{1}{6} \times \frac{1}{8} \right) = \frac{1}{8} \\
 \cos(\text{cats}, \text{dogs}) &= \left(\frac{1}{2} \right) \times 2 \times \left(\frac{1}{4} \times \frac{1}{8} \right) = \frac{1}{32} \\
 \cos(\text{dogs}, \text{mice}) &= \frac{1}{4} \times \frac{1}{6} = \frac{1}{24}
 \end{aligned}$$

The corpus lacks an explicit sentence declaring that mice are also animals. Hence, from the sentences of the corpus, the negation of ‘dogs chase mice’ follows, which is a wrong entailment in the real world. This wrong can now be put right, since we can replace the word ‘Cats’ in the admitted sentence ‘Cats are animals’ with ‘Mice’; as we have $\cos(\text{cats}, \text{mice}) = \frac{1}{8}$, thus obtaining the situation where c admits the following, both with degree $\frac{1}{8}$:

Mice are animals.
Dogs chase mice.

These were not possible before. We also obtain admittance of the following sentences, albeit with a lower degree of $\frac{1}{24}$:

- (*) Cats like dogs.
- (*) Cats eat dogs.
- (*) Dogs run from cats.

Some other examples are as follows, with a still lower degree of $\frac{1}{32}$:

- (*) Dogs like mice.
- (*) Mice fear dogs.
- (*) Dogs eat mice.

Some of the above are as likely as those that were derived with degree $\frac{1}{8}$. This is because the degrees come from co-occurrences in corpora, and our corpus is quite limited. One hopes that the bigger the corpus, the more reflective of the real world it will be. Another way of improving word-based entailments is by using linguistic resources such as WordNet, e.g. replacing words with their hypernyms.

8.1 *Evaluating on existing entailment datasets*

It remains to see if the notion of admittance of a sentence by a context can be applied to derive entailment relations between sentences. In future work, we will put this method to the test on inference datasets such as FraCaS (Cooper *et al.* 1996), SNLI (Bowman *et al.* 2015), the dataset in Zeichner *et al.* (2012), and the datasets in the RTE challenge. The FraCaS inferences are logical and the lambda calculus models of language should help in deriving them. As an example, consider the fracas-013 test case:

- fracas-013 answer: yes
- P1 Both leading tenors are excellent.
- P2 Leading tenors who are excellent are indispensable.
- Q Are both leading tenors indispensable?
- H Both leading tenors are indispensable.

In our setting, using the updates resulting from P1 and P2, one can contextually derive H. In Zeichner *et al.* (2012), the similarity between words is also taken into account. An example is the following entailment between two sentences; this entailment was judged to be valid with confidence by human annotators:

Parents have great influence on the career development of their children.

Parents have a powerful influence on the career development of their children.

We can derive the above with a contextual entailment consisting of a cube updated by just the above two sentences, with the degree of similarity between ‘powerful’ and ‘great’, mined from the co-occurrence matrix of a large corpus.

Judgements on the SNLI dataset are more tricky, as they rely on external knowledge. For example, consider the entailment between the following phrases:

A soccer game with multiple males playing.

Some men are playing a sport.

or the contradiction between the following:

A black race car starts up in front of a crowd of people.

A man is driving down a lonely road.

Deciding these correctly is a challenge for our framework. The strength of our approach is in deciding whether a set of sentences follows from a given corpus of texts, rather than in judging entailment relations between a given pair or triple of sentences. Nevertheless, we shall try to experiment with all these datasets.

9 CONCLUSION AND FUTURE DIRECTIONS

We showed how a static interpretation of a lambda calculus model of natural language provides vector representations for phrases and sentences. Here, the type of the vector of a word depended on its abstract type, and could be an atomic vector, a matrix, or a cube, or a tensor of higher rank. Combinations of these vary, based on the tensor rank of the type of each word involved in the combination. For instance, one could take the matrix multiplication of the matrix of an intransitive verb with the vector of its subject, whereas for a transitive verb the sequence of operations was a contraction between the cube of the verb and the vector of its object, followed by a matrix multiplication between the resulting matrix and the vector of the subject. A toolkit of functions needed to perform these operations was defined. This toolkit can be restated for types of tensors of higher order, such as I^2R and I^3R ,

rather than the current *IR*, to provide a means of combining matrices, cubes, and their updates, if needed.

We extended the above setting by reasoning about the notion of context and its update, and developing a dynamic vector interpretation for the language of lambda terms. Truth conditional and vector models of language follow two very different philosophies. Vector models are based on contexts, truth models on denotations. Our first interpretation was static and based on truth conditions. Our second approach is based on a dynamic interpretation, where we followed the context update model of Heim (1983), and hence is deemed the more appropriate choice. We showed how Heim's files can be turned into vector contexts and how her context change potentials can be used to provide vector interpretations for phrases and sentences. We treated sentences as Heim's context change potentials and provided update instructions for words therein – including quantifiers, negation, and coordination words. We provided two concrete realisations of contexts, i.e. co-occurrence matrices and entity relation cubes, and in each case detailed how these context update instructions allow contexts to thread through vector semantics in a compositional manner. With an eye towards a large-scale empirical evaluation of the model, we defined a notion of 'contexts admitting sentences' and degrees thereof between contexts and sentences, and showed, by means of examples, how these notions can be used to judge whether a sentence is entailed by a cube context or by a pair of cube and matrix contexts. A large-scale empirical evaluation of the model is currently underway.

Our approach is applicable to the lambda terms obtained via other syntactic models, e.g. CCG, and Lambek grammars, and can also be modified to develop a vector semantics for LFG. We also aim to work with other update semantics, such as continuation-based approaches. One could also have a general formalisation wherein both the static approach of previous work and the dynamic one of this work cohabit. This can be achieved by working out a second pair of type-term homomorphisms that will also work with Heim's possible world part of the contexts. In this setting, the two concepts of meaning: truth theoretic and contextual, each with its own uses and possibilities, can work in tandem.

An intuitive connection to fuzzy logic is imaginable, wherein one interprets the logical words in more sophisticated ways: for instance,

conjunction and disjunction take max and min of their entries, or add and subtract them. It may be worth investigating if such connections add to the applicability of the current model and, if so, make the connection formal.

ACKNOWLEDGEMENTS

We wish to thank the anonymous referees for their very valuable remarks. The anonymous referees of a short version of this paper, presented at LACL 2017, also gave excellent feedback. Carmela Chateau Smith's meticulous copy-editing considerably improved the readability of the paper and the grammaticality of its phrasing. All remaining errors are ours. The research for this paper was supported by the Royal Society International Exchange Award IE161631.

REFERENCES

- Marco BARONI, Raffaella BERNARDI, and Roberto ZAMPARELLI (2014), Frege in space: A program for compositional distributional semantics, *Linguistic Issues in Language Technology*, 9:241–346.
- Marco BARONI and Roberto ZAMPARELLI (2010), Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space, in Hang LI and Lluís MÀRQUEZ, editors, *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pp. 1183–1193, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/D10-1115>.
- Johan VAN BENTHEM (1986), *Essays in Logical Semantics*, Studies in Linguistics and Philosophy 29, Reidel, Dordrecht.
- Samuel BOWMAN, Gabor ANGELI, Christopher POTTS, and Christopher MANNING (2015), A large annotated corpus for learning natural language inference, in Lluís MÀRQUEZ, Chris CALLISON-BURCH, and Jian SU, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pp. 632–642, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/D15-1075>.
- Bob COECKE, Edward GREFENSTETTE, and Mehrnoosh SADRADEH (2013), Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus, *Annals of Pure and Applied Logic*, 164(11):1079–1100.
- Bob COECKE, Mehrnoosh SADRADEH, and Stephen CLARK (2010), Mathematical foundations for distributed compositional model of meaning, *Linguistic Analysis*, 36:345–384.

Robin COOPER, Dick CROUCH, Jan VAN EIJCK, Chris FOX, Johan VAN GENABITH, Jan JASPARS, Hans KAMP, David MILWARD, Manfred PINKAL, and Massimo POESIO (1996), *Using the framework*, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Mary DALRYMPLE, John LAMPING, and Vijay SARASWAT (1993), LFG semantics via constraints, in *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*, EACL '93, pp. 97–105, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/E93-1013>.

John Rupert FIRTH (1957), A synopsis of linguistic theory, 1930–1955, in *Studies in Linguistic Analysis*, pp. 1–32, Blackwell, Oxford.

Edward GREFENSTETTE and Mehrnoosh SADRZADEH (2015), Concrete models and empirical evaluations for the categorical compositional distributional model of meaning, *Computational Linguistics*, 41:71–118.

Philippe DE GROOTE (2001), Towards abstract categorial grammars, in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and 10th Conference of the European Chapter of the Association for Computational Linguistics*, ACL '01, pp. 252–259, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/P01-1033>.

Philippe DE GROOTE (2006), Towards a Montagovian account of dynamics, *Semantics and Linguistic Theory*, 16:1–16.

Irene HEIM (1983), On the projection problem for presuppositions, in *Proceedings of the Second Annual West Coast Conference on Formal Linguistics*, pp. 114–125, reprinted in Portner and Partee (2002).

Irene HEIM and Angelika KRATZER (1998), *Semantics in Generative Grammar*, Blackwell textbooks in linguistics, Blackwell, Oxford, ISBN 0-631-19712-5.

Leon HENKIN (1950), Completeness in the theory of types, *Journal of Symbolic Logic*, 15:81–91.

Nanda KAMBHATLA (2004), Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations, in *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/P04-3022>.

Lauri KARTTUNEN (1974), Presupposition and linguistic context, *Theoretical Linguistics*, 1(1–3):182–194.

Ewan KLEIN and Ivan SAG (1985), Type-driven translation, *Linguistics and Philosophy*, 8(2):163–201.

Jayant KRISHNAMURTHY and Tom MITCHELL (2013), Vector space semantic parsing: A framework for compositional vector space models, in *Proceedings of*

the 2013 ACL Workshop on Continuous Vector Space Models and their Compositionality, pp. 1–10, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/W13-3201>.

Mike LEWIS and Mark STEEDMAN (2013), Combined distributional and logical semantics, *Transactions of the Association for Computational Linguistics*, 1:179–192, <http://aclweb.org/anthology/Q13-1015>.

Jean MAILLARD, Stephen CLARK, and Edward GREFENSTETTE (2014), A type-driven tensor-based semantics for CCG, in Robin COOPER, Simon DOBNIK, Shalom LAPPIN, and Staffan LARSSON, editors, *Proceedings of the EACL 2014 on Type Theory and Natural Language Semantics (TTNLS)*, pp. 46–54, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/W14-1406>.

Jeff MITCHELL and Mirella LAPATA (2010), Composition in distributional models of semantics, *Cognitive Science*, 34(8):1388–1439.

Richard MONTAGUE (1974), The proper treatment of quantification in ordinary English, in Richmond THOMASON, editor, *Formal Philosophy. Selected Papers of Richard Montague*, pp. 247–270, Yale University Press, New Haven, CT.

Reinhard MUSKENS (2001), Categorical grammar and lexical-functional grammar, in Miriam BUTT and Tracy Holloway KING, editors, *Proceedings of the LFG01 Conference, University of Hong Kong*, pp. 259–279, CSLI, Stanford, CA, <http://cslipublications.stanford.edu/LFG/6/lfg01.html>.

Reinhard MUSKENS (2003), Language, lambdas, and logic, in Geert-Jan KRUIJFF and Richard OEHRLE, editors, *Resource-Sensitivity, Binding and Anaphora*, Studies in Linguistics and Philosophy, pp. 23–54, Kluwer, Dordrecht.

Reinhard MUSKENS (2010), New directions in type-theoretic grammars, *Journal of Logic, Language and Information*, 19(2):129–136.

Reinhard MUSKENS and Mehrnoosh SADRZADEH (2016a), Context update for lambdas and vectors, in Maxime AMBLARD, Philippe DE GROOTE, Sylvain POGODALLA, and Christian RETORÉ, editors, *Proceedings of the 9th International Conference on Logical Aspects of Computational Linguistics (LACL 2016)*, volume 10054 of LNCS, pp. 247–254, Springer-Verlag, Berlin, Heidelberg.

Reinhard MUSKENS and Mehrnoosh SADRZADEH (2016b), Lambdas and vectors, in *Workshop on Distributional Semantics and Linguistic Theory (DSALT), 28th European Summer School in Logic, Language and Information (ESSLLI)*, Free University of Bozen-Bolzano.

Reinhard MUSKENS, Johan VAN BENTHEM, and Albert VISSER (1997), Dynamics, in Johan VAN BENTHEM and Alice TER MEULEN, editors, *Handbook of Logic and Language*, pp. 587–648, Elsevier.

Barbara PARTEE (1986), Noun phrase interpretation and type-shifting principles, in Jeroen GROENENDIJK, Dick DE JONGH, and Martin STOKHOF,

editors, *Studies in Discourse Representation and the Theory of Generalized Quantifiers*, pp. 115–143, Foris, Dordrecht.

Hoifung POON and Pedro DOMINGOS (2009), Unsupervised semantic parsing, in Philipp KOEHN and Rada MIHALCEA, editors, *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-9): Volume 1*, pp. 1–10, Association for Computational Linguistics, Stroudsburg, PA, ISBN 978-1-932432-59-6, <http://aclweb.org/anthology/D09-1001>.

Paul PORTNER and Barbara PARTEE (2002), *Formal Semantics: The Essential Readings*, Blackwell, Oxford.

Sebastian RIEDEL, Limin YAO, and Andrew MCCALLUM (2010), Modeling relations and their mentions without labeled text, in José Luis BALCÁZAR, Francesco BONCHI, Aristides GIONIS, and Michèle SEBAG, editors, *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'10): Part III*, volume 6323 of LNAI, pp. 148–163, Springer-Verlag, Berlin, Heidelberg, ISBN 3-642-15938-9, 978-3-642-15938-1, <http://dl.acm.org/citation.cfm?id=1889788.1889799>.

Herbert RUBENSTEIN and John GOODENOUGH (1965), Contextual correlates of synonymy, *Communications of the ACM*, 8(10):627–633.

Mark STEEDMAN (2000), *The Syntactic Process*, MIT Press.

Alfred TARSKI (1965), *Introduction to Logic and to the Methodology of Deductive Sciences*, Oxford University Press, Oxford, 3rd edition.

Frank VELTMAN (1996), Defaults in update semantics, *Journal of Philosophical Logic*, 25(3):221–261.

Limin YAO, Sebastian RIEDEL, and Andrew MCCALLUM (2012), Unsupervised relation discovery with sense disambiguation, in Haizhou LI, Chin-Yew LIN, Miles OSBORNE, Gary Geunbae LEE, and Jong C. PARK, editors, *Proceedings of the 50th annual meeting of the Association for Computational Linguistics: long papers – Volume 1*, ACL '12, pp. 712–720, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/P12-1075>.

Naomi ZEICHNER, Jonathan BERANT, and Ido DAGAN (2012), Crowdsourcing inference-rule evaluation, in Haizhou LI, Chin-Yew LIN, Miles OSBORNE, Gary Geunbae LEE, and Jong C. PARK, editors, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers – Volume 2*, ACL '12, pp. 156–160, Association for Computational Linguistics, Stroudsburg, PA, <http://aclweb.org/anthology/P12-2031>.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



A note on movement in logical grammar^{*}

Glyn Morrill

Department of Computer Science, Universitat Politècnica de Catalunya

ABSTRACT

In this article, we make some brief remarks on overt and covert movement in logical grammar. With respect to covert movement (e.g. quantification), we observe how a treatment in terms of displacement calculus interacts with normal modalities for intensionality to allow a coding in logical grammar of the distinction between weak and strong quantifiers (i.e. those that may or may not scope nonlocally such as *a* and *every* respectively). With respect to overt movement (e.g. relativisation), we observe how displacement calculus can support a coding of a linear filler-gap dependency similar to that employed in lambda grammars, but we argue that this general approach does not extend to either the multiplicity nor the island-sensitivity of parasitic gaps, for which we advocate instead treatment in terms of a bracket-conditioned contraction subexponential.

Keywords:
covert and overt
movement,
parasitic
extraction,
relativisation,
weak and strong
quantification

1 COVERT MOVEMENT: QUANTIFICATION

Montague's rule S14 of quantification (ignoring pronoun binding) can be expressed as follows:

$$\frac{\Delta(N: x) \Rightarrow S: \omega}{\Delta(QP: \chi) \Rightarrow S: (\chi \lambda x \omega)}$$

That is quantifier phrases occupy nominal positions and take semantic scope at the sentence level, applying to the lambda abstraction of the

^{*}Research partially supported by an ICREA Acadèmia 2012, and by SGR2014-890 (MACDA) of the Generalitat de Catalunya, and grants APCOM TIN2014-57226-P and TIN2017-89244-R from MINECO (Ministerio de Economía, Industria y Competitividad). I am grateful to three reviewers for their valuable comments and suggestions.

sentence context over the meaning of the nominal position occupied. Montague's rule allows any quantifier to take scope at the level of any superordinate clause, correctly generating, for example, de re and de dicto readings of:

(1) John thinks a spy sleeps.

But incorrectly overgenerating two readings of e.g.

(2) John thinks every spy sleeps.

In the logical rules of the calculus of Lambek (1958) $\Delta(\Gamma)$ signifies context configuration Δ with a distinguished subconfiguration Γ :

$$\begin{array}{c}
 \frac{\Gamma \Rightarrow B \quad \Delta(C) \Rightarrow D}{\Delta(C/B, \Gamma) \Rightarrow D} /L \qquad \frac{\Gamma, B \Rightarrow C}{\Gamma \Rightarrow C/B} /R \\
 \\
 \frac{\Gamma \Rightarrow A \quad \Delta(C) \Rightarrow D}{\Delta(\Gamma, A \setminus C) \Rightarrow D} \setminus L \qquad \frac{A, \Gamma \Rightarrow C}{\Gamma \Rightarrow A \setminus C} \setminus R \\
 \\
 \frac{\Delta(A, B) \Rightarrow D}{\Delta(A \bullet B) \Rightarrow D} \bullet L \qquad \frac{\Delta \Rightarrow A \quad \Gamma \Rightarrow B}{\Delta, \Gamma \Rightarrow A \bullet B} \bullet R \\
 \\
 \frac{\Delta(\Lambda) \Rightarrow A}{\Delta(I) \Rightarrow A} IL \qquad \frac{}{\Lambda \Rightarrow I} IR
 \end{array}$$

Here, we allow the metalinguistic empty antecedent Λ , and we have added the product unit I (such that $I \bullet A \Leftrightarrow A$ and $A \Leftrightarrow A \bullet I$).

Using Lambek's system requires lexical ambiguity to obtain both sentence left-peripheral quantification (e.g. *Everyone loves Mary*) and right-peripheral quantification (e.g. *John loves someone*):

- (3) $a: (S/(N \setminus S))/CN: \lambda x \lambda y \exists z[(x \ z) \wedge (y \ z)]$
 $a: ((S/N) \setminus S)/CN: \lambda x \lambda y \exists z[(x \ z) \wedge (y \ z)]$
every: $(S/(N \setminus S))/CN: \lambda x \lambda y \forall z[(x \ z) \rightarrow (y \ z)]$
every: $((S/N) \setminus S)/CN: \lambda x \lambda y \forall z[(x \ z) \rightarrow (y \ z)]$

And would require still further lexical ambiguity for medial quantification:

(4) John sent every student to Mary.

Moot and Retoré (2016) give a counting argument showing that in the Lambek calculus no finite number of lexical entries can generate all $n!$ quantifier scoping proofs.

In the Lambek calculus, $(\backslash, \bullet, /; \Rightarrow)$ is a *residuated triple*:

$$(5) \quad B \Rightarrow A \backslash C \text{ iff } A \bullet B \Rightarrow C \text{ iff } A \Rightarrow C / B$$

The Lambek calculus is a logic of concatenation, with the inference of the residuated triple $\{\backslash, \bullet, /\}$ hinging on the metasyntactic concatenative comma “,”. To account also for discontinuity, Morrill *et al.* (2011) define the displacement calculus. In the displacement calculus, types are sorted by naturals according to the number of points of discontinuity their expressions contain. In addition to a residuated triple $\{\backslash, \bullet, /\}$ of continuous connectives, there are residuated discontinuous connectives $\{\downarrow_k, \odot_k, \uparrow_k\}$ for which inference hinges on the metasyntactic intercalation “ $|_k$ ” where the positive integer k indicates the point of discontinuity in question counting from the left (it defaults to 1 under omission.)

Configurations \mathcal{O} are defined by the following (where the *separator* 1 marks points of discontinuity):

$$(6) \quad \begin{aligned} \mathcal{O} &::= \Lambda \mid \mathcal{T}, \mathcal{O} \\ \mathcal{T} &::= 1 \mid \mathcal{F}_0 \mid \mathcal{F}_{i>0} \{ \underbrace{\mathcal{O} : \dots : \mathcal{O}}_{i \text{ } \mathcal{O}'\text{'s}} \} \end{aligned}$$

For a type A , its sort $s(A)$ is the i such that $A \in \mathcal{F}_i$. For a configuration Γ , its sort $s(\Gamma)$ is $|\Gamma|_1$, i.e. the number of points of discontinuity 1 which it contains.

Sequents are of the form:

$$(7) \quad \mathcal{O} \Rightarrow \mathcal{F} \text{ such that } s(\mathcal{O}) = s(\mathcal{F})$$

The figure \vec{A} of a type A is defined by:

$$(8) \quad \vec{A} = \begin{cases} A & \text{if } s(A) = 0 \\ A \{ \underbrace{1 : \dots : 1}_{s(A) \text{ } 1'\text{'s}} \} & \text{if } s(A) > 0 \end{cases}$$

Where Γ is a configuration of sort i and $\Delta_1, \dots, \Delta_i$ are configurations, the *fold* $\Gamma \otimes \langle \Delta_1 : \dots : \Delta_i \rangle$ is the result of replacing the successive 1's in Γ by $\Delta_1, \dots, \Delta_i$ respectively. Where Γ is of sort i , the hyperoccurrence notation $\Delta \langle \Gamma \rangle$ abbreviates $\Delta_0(\Gamma \otimes \langle \Delta_1 : \dots : \Delta_i \rangle)$, i.e. a context configuration Δ (which is externally Δ_0 and internally $\Delta_1, \dots, \Delta_i$) with

a potentially discontinuous distinguished subconfiguration Γ (continuous if $i = 0$, discontinuous if $i > 0$). Where Δ is a configuration of sort $i > 0$ and Γ is a configuration, the k th *metalinguistic intercalation* $\Delta|_k\Gamma$, $1 \leq k \leq i$, is given by:

$$(9) \quad \Delta|_k\Gamma =_{df} \Delta \otimes \underbrace{\langle 1 : \dots : 1 \rangle}_{k-1 \text{ 1's}} : \Gamma : \underbrace{\langle 1 : \dots : 1 \rangle}_{i-k \text{ 1's}}$$

That is $\Delta|_k\Gamma$ is the configuration resulting from replacing by Γ the k th separator in Δ .

The logical rules of the displacement calculus are as follows, where as we have said $\Delta\langle\Gamma\rangle$ signifies a configuration Δ with a potentially discontinuous distinguished subconfiguration Γ :

$$\begin{array}{ll} \frac{\Gamma \Rightarrow B \quad \Delta\langle\vec{C}\rangle \Rightarrow D}{\Delta\langle\vec{C/B}, \Gamma\rangle \Rightarrow D} /L & \frac{\Gamma, \vec{B} \Rightarrow C}{\Gamma \Rightarrow C/B} /R \\[10pt] \frac{\Gamma \Rightarrow A \quad \Delta\langle\vec{C}\rangle \Rightarrow D}{\Delta\langle\Gamma, A\backslash\vec{C}\rangle \Rightarrow D} \backslash L & \frac{\vec{A}, \Gamma \Rightarrow C}{\Gamma \Rightarrow A\backslash C} \backslash R \\[10pt] \frac{\Delta\langle\vec{A}, \vec{B}\rangle \Rightarrow D}{\Delta\langle\vec{A\bullet B}\rangle \Rightarrow D} \bullet L & \frac{\Delta \Rightarrow A \quad \Gamma \Rightarrow B}{\Delta, \Gamma \Rightarrow A\bullet B} \bullet R \\[10pt] \frac{\Delta\langle\Lambda\rangle \Rightarrow A}{\Delta\langle\vec{I}\rangle \Rightarrow A} IL & \frac{}{\Lambda \Rightarrow I} IR \\[10pt] \frac{\Gamma \Rightarrow B \quad \Delta\langle\vec{C}\rangle \Rightarrow D}{\Delta\langle\vec{C\uparrow_k B}|_k\Gamma\rangle \Rightarrow D} \uparrow_k L & \frac{\Gamma|_k\vec{B} \Rightarrow C}{\Gamma \Rightarrow C\uparrow_k B} \uparrow_k R \\[10pt] \frac{\Gamma \Rightarrow A \quad \Delta\langle\vec{C}\rangle \Rightarrow D}{\Delta\langle\Gamma|_k A\downarrow_k\vec{C}\rangle \Rightarrow D} \downarrow_k L & \frac{\vec{A}|_k\Gamma \Rightarrow C}{\Gamma \Rightarrow A\downarrow_k C} \downarrow_k R \\[10pt] \frac{\Delta\langle\vec{A}|_k\vec{B}\rangle \Rightarrow D}{\Delta\langle\vec{A\odot_k B}\rangle \Rightarrow D} \odot_k L & \frac{\Delta \Rightarrow A \quad \Gamma \Rightarrow B}{\Delta|_k\Gamma \Rightarrow A\odot_k B} \odot_k R \\[10pt] \frac{\Delta\langle 1\rangle \Rightarrow A}{\Delta\langle\vec{J}\rangle \Rightarrow A} JL & \frac{}{1 \Rightarrow J} JR \end{array}$$

$$\begin{array}{c}
 \frac{\Xi(\vec{A}:x) \Rightarrow B:\psi}{\Xi(\vec{\Box A}:z) \Rightarrow B:\psi\{\vee z/x\}} \Box L \qquad \frac{\Box \Xi \Rightarrow A:\phi}{\Box \Xi \Rightarrow \Box A:\wedge \phi} \Box R \\
 \\
 \frac{\Xi(\vec{A}:x) \Rightarrow B:\psi}{\Xi(\vec{\blacksquare A}:x) \Rightarrow B:\psi} \blacksquare L \qquad \frac{\Box \Xi \Rightarrow A:\phi}{\Box \Xi \Rightarrow \blacksquare A:\phi} \blacksquare R
 \end{array}$$

Figure 1:
Normal modalities, where
 \Box marks a structure all the
types of which have main
connective a box

Then for all of left-peripheral and right-peripheral and medial quantification à la Montague we require just single type assignments:

$$\begin{aligned}
 (10) \quad a: ((S \uparrow N) \downarrow S) / CN: \lambda x \lambda y \exists z [(x z) \wedge (y z)] \\
 \text{every}: ((S \uparrow N) \downarrow S) / CN: \lambda x \lambda y \forall z [(x z) \rightarrow (y z)]
 \end{aligned}$$

Hence the rule of S14 is lexicalised in a single lexical type.

Morrill (1990) and Hepple (1990) invoke semantically active (\Box) and inactive (\blacksquare) normal modalities respectively for grammatical domains. These are normal (i.e. distributive) **S4** modalities; the former for semantic, e.g. intensional or temporal, domains, and the latter for syntactic domains. Morrill (2015) combines these as shown in Figure 1. Adding these to displacement calculus we can approach the capture of clause-locality invoking sensitivity to intensionality:

$$\begin{aligned}
 (11) \quad a: \blacksquare(((S \uparrow \blacksquare N) \downarrow S) / CN): \lambda x \lambda y \exists z [(x z) \wedge (y z)] \\
 \text{every}: \blacksquare(((S \uparrow N) \downarrow S) / CN): \lambda x \lambda y \forall z [(x z) \rightarrow (y z)] \\
 \text{John}: \blacksquare N: j \\
 \text{sleep}: \Box(N \setminus S): \text{sleep} \\
 \text{spy}: \Box CN: \text{spy} \\
 \text{thinks}: \Box((N \setminus S) / \Box S): \text{think}
 \end{aligned}$$

A subordinate clause such as the complement of *thinks* is an intensional domain $\Box S$ and thus requires its elements to be modal at the moment of \Box proof. There is no problem when either *a* or *every* scopes locally within an intensional domain such as the complement clause of *thinks* since their lexical types, like all lexical types, are modal. But while the hypothetical subtype $\blacksquare N$ of *a* bears a modality, that *N* of *every* does not, and so only the former can take wide scope out of its intensional domain.

2 OVERT MOVEMENT: RELATIVISATION

In the ACG of de Groote (2001) or Lambda Grammar of Muskens (2001b) a relative pronoun is assigned type:

$$\lambda\rho\lambda\sigma.\sigma + \textit{that} + (\rho\ 0): (N \multimap S) \multimap CN \multimap CN: \lambda x\lambda y\lambda z[(y\ z) \wedge (x\ z)]$$

But ACG has the KLM problem; see Muskens (2001a); Kubota (2010); Kubota and Levine (2012); Moot (2014); and Kubota and Levine (2015), section 4.1.2, whereby the nondirectional dependents of an argument to a higher order functor can commute. This is because $\vdash A \multimap B \multimap C \Rightarrow B \multimap A \multimap C$. For example, in *TV* coordination the natural seeming translation of Lambek coordination would be to assign the coordinator type:

$$\lambda\rho_2\lambda\rho_1\lambda\beta\lambda\alpha.\alpha + (\rho_1\ 0\ 0) + \textit{and} + (\rho_2\ 0\ 0) + \beta: X \multimap X \multimap X,$$

where $X = N \multimap N \multimap S$

Then

(12) John saw and praised Mary.

gets assigned the following readings which are all incorrect except the first:

- (13) “J saw M and J praised M”
 “J saw M and M praised J”
 “M saw J and J praised M”
 “M saw J and M praised J”

The more general point is that all alternative terms overgenerate as well, which is argued in Moot (2014).

In HTLG (Kubota and Levine 2012) there are both directional Lambek connectives for continuity and a nondirectional linear connective for discontinuity. The KLM problem above is evaded by assigning a *TV* coordinator the directed type:

$$\textit{and}: (X \setminus X) / X, \text{ where } X = (N \setminus S) / N$$

But directional (concatenative) and nondirectional (functional) types cannot freely interweave in HTLG: interpreting concatenation as function composition only makes sense for functions from string position

to string position (i.e. simple strings) and not for more complex functions; other things being equal, directed types cannot contain nondirectional subtypes. Thus, other things being equal, the assignment of a relative pronoun on the pattern of that above for lambda grammar must be:

$$that: (CN \setminus CN) \upharpoonright (S \upharpoonright N): \lambda x \lambda y \lambda z [(y \ z) \wedge (x \ z)]$$

with an outermost nondirectional slash because the argument has a nondirectional slash to allow medial extraction. But this means there is a potential KLM problem for the $\upharpoonright (S \upharpoonright N)$ argument. For example, to generate the following we require *or*: $(X \upharpoonright X) \upharpoonright X$, where $X = CN \upharpoonright (S \upharpoonright N)$:

(14) animal that or person who John saw today

However, the same types overgenerate the following, where the right node raised $S \upharpoonright N$ is medial in one or both of the disjuncts.

- (15) a. *animal that outside or person who John saw today
 b. *animal that or person who inside John saw today
 c. *animal that outside or person who inside John saw today

In response to this, Yusuke Kubota (personal communication) suggests that a relative pronoun be assigned type

$$that: (CN \setminus CN) / \wedge (S \upharpoonright N): \lambda x \lambda y \lambda z [(y \ z) \wedge (x \ z)]$$

where \wedge is the defined connective “bridge” of displacement calculus. That is, in this case the KLM problem would be resolved through the use of an additional connective; however, note that while this use is motivated by a desire to correct empirical predictions, it is a technically anomalous addition to HTLG.

In determiner gapping in HTLG, see Kubota and Levine (2013) and Kubota and Levine (2016), there is a further remnant KLM problem (Kubota, personal communication):

- (16) a. Most cats like Alpo and (most) dogs (like) Whiskas.
 b. I like most cats and you (like) (most) dogs.

$$\lambda \rho_2 \lambda \rho_1 \lambda \phi \lambda \sigma . ((\rho_1 \ \phi) \ \sigma) + and + ((\rho_2 \ \lambda \chi \lambda \psi (\psi \ \chi)) \ 0): (X \upharpoonright X) \upharpoonright X,$$

$$\text{where } X = (S \upharpoonright TV) \upharpoonright Q$$

This overgenerates the following, where the determiner and the transitive verb orders are not consistent in the conjuncts:

(17) *Most cats like Alpo and John (likes) (most) dogs.

(18) *John likes most dogs and (most) cats (like) Alpo.

This overgeneration arises because the left-to-right positions of the two discontinuous dependencies are not identified. For an account of gapping that includes determiner gapping without this problem, in terms of a version of displacement calculus, see Morrill and Valentín (2017); that formulation evades the overgeneration because in displacement calculus the discontinuous dependents are indexed for left-to-right position, allowing the parallel grammatical determiner gapping of (16) but not the nonparallel ungrammatical cases (17) and (18).

In displacement calculus, a relative pronoun can be assigned type:

$$that: (CN \setminus CN) / ((S \uparrow N) \odot I): \lambda x \lambda y \lambda z [(y \ z) \wedge (\pi_1 x \ z)]$$

or

$$that: (CN \setminus CN) / ^{(S \uparrow N)}: \lambda x \lambda y \lambda z [(y \ z) \wedge (x \ z)]$$

There is no KLM problem of any kind. However, we offer two reasons to question any use of a discontinuous linear operator for relativisation.

First, let us observe that using displacement operators for both quantification and relativisation risks running into an inconsistency. This is as follows: on the one hand, quantifiers must be allowed to scope out of, for example, subjects, which are (weak) islands, so to treat quantification, displacement must be able to penetrate islands. But then, on the other hand, the linear proposal for relativisation above will fail to be sensitive to islands.

Second, nor do the linear proposals above take into account parasitic extraction:

(19) man that the friends of admire

Therefore, we suggest treatment of the mediality of extraction and the potential for parasitic extraction not via an island-insensitive discontinuous linear implication, but via a permutation and island-conditioned contraction (but not weakening) subexponential; see Figure 2 which uses a stoup, as in Girard (2011), to store the structurally modalised resources; this formulation is essentially like that of Mor-

$$\begin{array}{c}
 \frac{\Xi(\zeta \uplus \{A: x\}; \Gamma_1, \Gamma_2) \Rightarrow B: \psi}{\Xi(\zeta; \Gamma_1, !A: x, \Gamma_2) \Rightarrow B: \psi} !L \quad \frac{!A \Rightarrow B: \phi}{!A \Rightarrow !B: \phi} !R \\
 \\
 \frac{\Xi(\zeta; \Gamma_1, A: x, \Gamma_2) \Rightarrow B: \psi}{\Xi(\zeta \uplus \{A: x\}; \Gamma_1, \Gamma_2) \Rightarrow B: \psi} !P \\
 \\
 \frac{\Xi(\zeta \uplus \{A: x\}; \Gamma_1, [\{A: y\}; \Gamma_2], \Gamma_3) \Rightarrow B: \psi}{\Xi(\zeta \uplus \{A: x\}; \Gamma_1, [[\Gamma_2]], \Gamma_3) \Rightarrow B: \psi\{x/y\}} !C_{bb}
 \end{array}$$

Figure 2:
Exponentials

rill (2011) in that parasitic domains must be doubly bracketed in the linguistic input.¹

A relative pronoun is to bear the permutation and bracket-conditioned contraction subexponential on its hypothetical subtype:

$$that: (CN \setminus CN) / (!N \setminus S): \lambda x \lambda y \lambda z [(y \ z) \wedge (x \ z)]$$

When this subtype has been lowered into the antecedent, it can be moved into the local stoup by $!L$; then it can be copied into the stoups of any number of (doubly bracketed) parasitic domains by $!C_{bb}$; then it can be moved into any local host position by $!P$. The stoup contents of the parasitic domains can themselves be copied into the stoups of any number of doubly bracketed parasitic subdomains $!C_{bb}$ and so forth, and then into local subhost positions by $!P$.

The bracket conditioning of contraction ensures that parasitic gaps can only appear within singly bracket modalized islands, hosted by a non-island gap; a discontinuous linear operator can deliver neither such multiple binding nor such island-conditioning.

¹ And the formulation stands in contrast to Morrill (2017) which has the contraction rule without brackets in the linguistic input:

$$\frac{\Xi(\zeta \uplus \{A: x\}; \Gamma_1, [\{A: y\}; \Gamma_2], \Gamma_3) \Rightarrow B: \psi}{\Xi(\zeta \uplus \{A: x\}; \Gamma_1, \Gamma_2, \Gamma_3) \Rightarrow B: \psi\{x/y\}} !C_b$$

which gives rise to undecidability as shown in Kanovich *et al.* (2017), and which furthermore overgenerates parasitic extraction in which a whole island domain is a parasitic gap, such as the subject island in the example:

*man that likes

which counterexample is due to Stepan Kuznetsov (personal communication).

The resulting picture, then, is one in which displacement calculus is used to characterise the covert movement of quantification, including employment of semantic modalities for the distinction between strong and weak quantifiers, but in which an exponential modality rather than a discontinuous linear operator is used for the overt movement of relativisation, for the reasons given above.

REFERENCES

- Philippe DE GROOTE (2001), Towards Abstract Categorical Grammars, in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, Toulouse.
- Jean-Yves GIRARD (2011), *The Blind Spot*, European Mathematical Society, Zürich.
- Mark HEPPLER (1990), *The Grammar and Processing of Order and Dependency*, Ph.D. thesis, University of Edinburgh.
- Max KANOVICH, Stepan KUZNETSOV, and Andre SCEDROV (2017), Undecidability of the Lambek calculus with subexponential and bracket modalities, in *Proc. FCT*, volume 10472 of *LNCS*, pp. 326–340.
- Yusuke KUBOTA (2010), *(In)flexibility of Constituency in Japanese in Multi-Modal Categorical Grammar with Structured Phonology*, Ph.D. thesis, Ohio State University.
- Yusuke KUBOTA and Robert LEVINE (2012), Gapping as Like-Category Coordination, in Denis BECHET and Alexander DIKOVSKY, editors, *Logical Aspects of Computational Linguistics*, volume 7351 of *Lecture Notes in Computer Science*, pp. 135–150, Springer Berlin Heidelberg, ISBN 978-3-642-31261-8, http://dx.doi.org/10.1007/978-3-642-31262-5_9.
- Yusuke KUBOTA and Robert LEVINE (2013), Determiner Gapping as Higher-Order Discontinuous Constituency, in Glyn MORRILL and Mark-Jan NEDERHOF, editors, *Formal Grammar: 17th and 18th International Conferences, FG 2012, Opole, Poland, August 2012, Revised Selected Papers, FG 2013, Düsseldorf, Germany, August 2013. Proceedings*, pp. 225–241, Springer, Berlin, Heidelberg, doi:10.1007/978-3-642-39998-5_14.
- Yusuke KUBOTA and Robert LEVINE (2015), Against ellipsis: Arguments for the direct licensing of ‘non-canonical’ coordinations, *Linguistics and Philosophy*, 38(6):521–576, doi:10.1007/s10988-015-9179-7.
- Yusuke KUBOTA and Robert LEVINE (2016), Gapping as hypothetical reasoning, *Natural Language and Linguistic Theory*, 34(1):107–156.
- Joachim LAMBEK (1958), The mathematics of sentence structure, *American Mathematical Monthly*, 65:154–170, doi:10.2307/2310058.

Richard MOOT (2014), Hybrid Type-Logical Grammars, First-Order Linear Logic and the Descriptive Inadequacy of Lambda Grammars, Technical report, LaBRI (CNRS), Bordeaux University.

Richard MOOT and Christian RETORÉ (2016), Natural Language Semantics and Computability, <https://arxiv.org/pdf/1605.04122.pdf>.

Glyn MORRILL (1990), Intensionality and Boundedness, *Linguistics and Philosophy*, 13(6):699–726.

Glyn MORRILL (2015), Structural Ambiguity in Montague Grammar and Categorical Grammar, *The Linguistic Review*, 32(1):87–113, doi:10.1515/tlr-2014-0017.

Glyn MORRILL (2017), Grammar Logicised: Relativisation, *Linguistics and Philosophy*, 40(2):119–163, doi:10.1007/s10988-016-9197-0, open access.

Glyn MORRILL and Oriol VALENTÍN (2017), A Reply to Kubota and Levine on Gapping, *Natural Language and Linguistic Theory*, 35(1):257–270, doi:10.1007/s11049-016-9336-x.

Glyn MORRILL, Oriol VALENTÍN, and Mario FADDA (2011), The Displacement Calculus, *Journal of Logic, Language and Information*, 20(1):1–48, doi:10.1007/s10849-010-9129-2.

Glyn V. MORRILL (2011), *Categorical Grammar: Logical Syntax, Semantics, and Processing*, Oxford University Press, New York and Oxford.

Reinhard MUSKENS (2001a), Categorical Grammar and Lexical-Functional Grammar, in *Proceedings of the LFG01 Conference*, pp. 259–279, University of Hong Kong.

Reinhard MUSKENS (2001b), Lambda Grammars and the Syntax-Semantics Interface, in R. VAN ROOY and M. STOKHOF, editors, *Proceedings of the Thirteenth Amsterdam Colloquium*, pp. 150–155, Amsterdam.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



EXTERNAL REVIEWERS 2016–2018

The mainstay of any peer-reviewed journal are its reviewers, and JLM is no exception here. Each paper is reviewed by at least 3 carefully selected reviewers, including at least one representing the JLM Editorial Board. To increase reviewing anonymity, we do not give the names of the 32 JLM EB reviewers, but we would like to heartily thank them for their hard and timely work. We also express our sincere gratitude to the following 117 external reviewers for papers reviewed during 2016–2018:

Afra Alishahi

Tilburg University

Daniel Altshuler

Hampshire College

Doug Arnold

University of Essex

Mark Baker

Rutgers University;
The State University of New Jersey

Emily Bender

University of Washington

Alexandra Birch

University of Edinburgh

Henrik Björklund

Umeå Universitet

James Blevins

University of Cambridge

Paul Boersma

Universiteit van Amsterdam

Johan Bos

Rijksuniversiteit Groningen

Katarzyna Budzyńska

Instytut Filozofii i Socjologii PAN

Dylan Bumford

New York University

Milos Cernak

IEEE

Lucas Champollion

New York University

Stergios Chatzikyriakidis

Göteborgs Universitet

Khalid Choukri

European Language Resources
Association

Alexander Clark

King's College London

Robin Cooper

Göteborgs Universitet

Ann Copestake

University of Cambridge

Berthold Crysmann

CNRS;

Université Paris 7 Denis Diderot

Natalie DelBusso

Rutgers University

Simon Dobnik

Göteborgs Universitet

Anaïd Donabedian

Institut National des Langues
et Civilisations

Katrin Erk

University of Texas at Austin

Christiane Fellbaum

Princeton University

Edward Flemming
Massachusetts Institute of Technology

Sean Fulop
California State University Fresno

Nina Gierasimczuk
Universiteit van Amsterdam

Carlos Gomez-Rodriguez
Universidade da Coruña

Thomas Graf
Stony Brook Univiersty

John Hale
Cornell University

Thomas Hanneforth
Universität Potsdam

Daniel Harbour
Queen Mary University of London

Petter Haugereid
Universitetet i Bergen

Jules Hedges
University of Oxford

Aurelie Herbelot
University of Cambridge; Universitat
Pompeu Fabra in Barcelona

Mans Hulden
University of Colorado

Adam Jardine
Rutgers University

Bryan Jurish
Berlin-Brandenburgische Akademie
der Wissenschaften

Sylvain Kahane
Université Paris Ouest – Nanterre

Simin Karimi
University of Arizona

Maciej Karpieński
Uniwersytet im. Adama Mickiewicza

Dimitrios Kartsaklis
Queen Mary University of London

Andre Kempe
Nuance Communications, Germany

Elma Kerz
Aachen University

Ezra Keshet
University of Michigan

Anna Kibort
University of Oxford

Katarzyna Klessa
Uniwersytet im. Adama Mickiewicza

Gregory Kobele
Universität Leipzig

Jacek Koronacki
Instytut Podstaw Informatyki PAN

Katarzyna Krasnowska
Instytut Podstaw Informatyki PAN

Yusuke Kubota
University of Tsukuba

Marco Kuhlmann
Linköpings Universitet

Daniel Lassiter
Stanford University

Lothar Lemnitzer
Berlin-Brandenburgische Akademie
der Wissenschaften

Robert Levine
Ohio State University

Hans-Heinrich Lieb
Freie Universität Berlin

John Lowe
University of Oxford

Zhaohui Luo
Royal Holloway, University of London

Veronika Lux-Pogodalla
ATILF CNRS

Andreas Maletti

Universität Stuttgart

Francesco Mambrini

Deutsches Archäologisches Institut,
Berlin Zentrale

Louise McNally

Universitat Pompeu Fabra

Chiara Melloni

Università degli Studi di Verona

Nazarre Merchant

Eckerd College

Paola Merlo

University of Geneva

Jens Michaelis

Universität Bielefeld

Marcin Miłkowski

Instytut Filozofii i Socjologii PAN

Richard Moot

CNRS; LIRMM;

Université de Montpellier

Lawrence Moss

Indiana University

Reinhard Muskens

Tilburg University

Günter Neumann

Deutsches Forschungszentrum
für Künstliche Intelligenz

Garrett Nicolai

University of Alberta

Wojciech Niemirow

Uniwersytet Warszawski

Denis Paperno

University of California, Los Angeles

Michal Peliš

Univerzita Karlova;

Filosofický ústav akademie věd ČR

Katya Pertova

University of Carolina at Chapel Hill

Miriam R.L. Petruck

International Computer Science
Institute

Sylvain Pogodalla

INRIA Nancy-Grand Est;

Université de Lorraine; CNRS

Carl Pollard

Ohio State University

Laurette Pretorius

University of South Africa

Piotr Przybyła

University of Manchester

Daniel Quernheim

Universität Stuttgart

Livy Real

IBM research Sao Paolo

Christian Retoré

Université de Montpellier

Frank M. Richter

Universität Frankfurt

Laura Rimell

University of Cambridge

Serge Rosmorduc

Conservatoire National des Arts
et Métiers

Anna Rumshisky

University of Massachusetts Lowell

Josef Ruppenhofer

Institut für Deutsche Sprache

Jan Rybicki

Uniwersytet Jagielloński

Mehrnoosh Sadrzadeh

Queen Mary University of London

Benoît Sagot

INRIA; Université Paris 7

Sylvain Salvati

INRIA

Uli Sauerland

Leibniz-Zentrum Allgemeine
Sprachwissenschaft

Helmut Schmid

Ludwig-Maximilians-Universität
München

Richard Sproat

Google

Miloš Stanojević

University of Edinburgh

Jakub Szymanik

Universiteit van Amsterdam

Julia Taylor Rayz

Purdue University;
Purdue Polytechnic Institute

Ida Toivonen

Carleton University

Oriol Valentín

Universitat Politècnica de Catalunya

Menno Van Zaanen

Universiteit van Tilburg

Cristina Vertan

Universität Hamburg

Laure Vieu

Universite Paul Sabatier

Jürgen Wedekind

Københavns Universitet

Gijs Wijnholds

Queen Mary University of London

Alexander Williams

University of Maryland

Andrzej Wiśniewski

Uniwersytet im. Adama Mickiewicza

Jacek Witkoś

Uniwersytet im. Adama Mickiewicza

Alina Wróblewska

Instytut Podstaw Informatyki PAN

Leszek Wroński

Uniwersytet Jagielloński

Christian Wurm

Heinrich-Heine-Universität Düsseldorf

Anssi Yli-Jyrä

University of Helsinki

Annie Zaenen

Stanford University

Sina Zarrieß

Bielefeld University

Hendrik Zeevat

Universiteit van Amsterdam