Semantic Vector Models and Functional Models for Pregroup Grammars

Anne Preller, Mehrnoosh Sadrzadeh

Semantic Vector Models and Functional Models for Pregroup Grammars

Anne Preller *†
LIRMM-CNRS, Montpellier, France
preller@lirmm.fr

Mehrnoosh Sadrzadeh ‡
Computing Laboratory, Oxford, U.K.
mehrs@comlab.ox.ac.uk

Abstract

We show that vector space semantics and functional semantics in two-sorted first order logic are equivalent for pregroup grammars. An algorithm translates functional expressions to vector expressions and vice-versa. The semantics is compositional, variable free and invariant under change of order or multiplicity. It includes the semantic vector models of Information Retrieval Systems and has an interior logic admitting a comprehension schema. A sentence is true in the interior logic if and only if the 'usual' first order formula translating the sentence holds. The examples include negation, universal quantifiers and relative pronouns.

Keywords: compositional semantics, quantum logic, pregroup grammars, semantic vector models, symmetric compact closed categories, two-sorted functional first order logic

1 Introduction

Pregroup grammars, [Lambek, 1999], are based on compact bilinear logic, [Lambek, 1993], a simplification of the Syntactic Calculus, [Lambek, 1958]. Compact bilinear logic or *Pregroup Calculus* places pregroup grammars in the family of categorial grammars where 'the grammar is in the lexicon'. The pregroup grammars proposed so far cover a variety of languages, but most of them with no reference to semantics. Indeed, giving up Syntactic Calculus meant giving up higher order logic, i.e. the Montague semantics of traditional categorial grammars. The semantic structures appropriate for pregroup grammars had to be rediscovered.

To fill this gap, compact *2*-categories are proposed in [Preller, 2005], followed by functional models in two-sorted first order logic in [Preller, 2007]. This semantics views verbs as predicates defined on individuals and sets of individuals.

*published in JoLLI, 2011
†Support by TALN/LIRMM is gratefully acknowledged
‡Support by EPSRC is gratefully acknowledged

On the other hand, [Sadrzadeh, 2007] and [Clark et al., 2008] consider the category of finite-dimensional real vector spaces and linear maps, with the explicit aim to add compositionality to distributive semantic vector models. They define the meaning of a transitive verb as a vector in the tensor product of three vector spaces. The basis vectors of two of the spaces are identified with nouns. The basis vectors of the third space are the pairs of these nouns. The value of a subject-object sentence formed by the verb and two nouns is computed with the help of the interior product. Later this method has been extended to negated sentences in [Preller and Sadrzadeh, 2011].

The starting point for this paper is the observation that the category of finite-dimensional real vector spaces and the category of sets and two-sorted functions are compact closed symmetric monoidal categories, or equivalently, symmetric compact *2*-categories. We prove the equivalence of the two approaches relating two-sorted first order logic to the theory of compact *2*-categories and thus to quantum logic. Indeed, [Abramsky and Coecke, 2004] formulate quantum protocols in the theory of compact *2*-categories. The logic for Information Retrieval Systems of [Rijsbergen, 2004] uses the lattice of projectors in finite-dimensional Hilbert spaces, also a compact *2*-category. A study by [Clarke, 2007] of context-theoretical semantics states that it is the lattice structure of real numbers that works best for semantic vector models. Here we show that the compact *2*-category of finite-dimensional semi-modules over a bounded lattice of real numbers handles both the quantum logic of semantic vector models and the functional models.

Section 2 introduces pregroup semantics in both the functional and the categorical versions. Section 3 presents an algorithm transforming one into the other. Section 4 gives an isomorphic embedding of the category of two-sorted functions into the category of semantic vector models over the lattice of real numbers. Follows in Section 5 a sketch of quantum logic in the category of semantic vector models over the lattice of real numbers. Section 6 motivates functions of two-sorted first order logic by their relevance to natural language and defines the structure of a symmetric compact *2*-category of the category of sets and two-sorted functions. In Section 7, we define the interior logic and show by examples that the vector of a sentence is true in the interior logic if and only if the 'usual' first order formula interpreting the sentence holds. The examples include universal quantifiers, negation and relative pronouns. In particular, the interpretation of the relative pronoun introduces a comprehension schema based on the interior truth of the sentence vectors, in accordance with an observation by [Benthem and Doets, 1983] that the comprehension schema holds in two-sorted first order logic for certain families of expressions.

2 The two semantics

Pregroup grammars are based on compact bilinear logic, also known as pregroup calculus. The set of pregroup types *generated by a partially ordered set* $\mathcal{B} = \langle B, \leq \rangle$ is the free monoid $\mathcal{T}_\mathcal{B}$ generated by the following set whose elements are

called *simple types*

$$\mathcal{S}_{\mathcal{B}} = \left\{ \boldsymbol{a}^{(z)} : \boldsymbol{a} \in B, z \in \mathbb{Z} \right\} .$$

The notation $\boldsymbol{a}^{(z)}$ designates the ordered pair formed by the element $\boldsymbol{a}$ of B and the integer z. A simple type of the form $\boldsymbol{a}^{(0)}$ is identified with $\boldsymbol{a} \in B$ and called a *basic type*. Strings of simple types are called *types*. By convention, if a type T is given in the form $T = t_1 \dots t_n$, the t_i's are simple types for $1 \leq i \leq n$. In the case where $n = 0$, the string $t_1 \dots t_n$ is empty, denoted ϵ. It is the unit for the binary operation of concatenation in the free monoid.

Moreover, there are two unary operations $(\,)^{\ell}$ and $(\,)^{r}$, called *left* and *right adjoints*, defined by

$$\begin{aligned} (\boldsymbol{a}_1^{(z_1)} \dots \boldsymbol{a}_n^{(z_n)})^{\ell} &= \boldsymbol{a}_n^{(z_n-1)} \dots \boldsymbol{a}_1^{(z_1-1)} \\ (\boldsymbol{a}_1^{(z_1)} \dots \boldsymbol{a}_k^{(z_n)})^{r} &= \boldsymbol{a}_n^{(z_n+1)} \dots \boldsymbol{a}_1^{(z_1+1)} . \end{aligned}$$

This definition includes the case where $n = 0$. Hence the equalities $\epsilon^{\ell} = \epsilon = \epsilon^{r}$ hold for the empty string. It follows that $\boldsymbol{a}^{\ell} = (\boldsymbol{a}^{(0)})^{\ell} = \boldsymbol{a}^{(-1)}$ and $\boldsymbol{a}^{r} = (\boldsymbol{a}^{(0)})^{r} = \boldsymbol{a}^{(1)}$

Finally, the binary *derivability relation* on types, denoted $\rightarrow$, is the smallest reflexive and transitive relation satisfying

(Induced step)	$T\boldsymbol{x}^{(z)}T' \rightarrow T\boldsymbol{y}^{(z)}T'$
(Generalised contraction)	$T\boldsymbol{x}^{(z)}\boldsymbol{y}^{(z+1)}T' \rightarrow TT'$
(Generalised expansion)	$TT' \rightarrow T\boldsymbol{x}^{(z+1)}\boldsymbol{y}^{(z)}T'$

where either z is even and $\boldsymbol{x} \leq \boldsymbol{y}$ or z is odd and $\boldsymbol{y} \leq \boldsymbol{x}$. Capitals T, T' etc. designate arbitrary types, bold face letters $\boldsymbol{x}$, $\boldsymbol{y}$ etc. basic types.

It follows that a type T' is derivable from type T, if either they are identical strings or there is a sequence $T_1, \dots, T_n$ for which $T_1 = T$, $T_n = T'$ and $T_i \rightarrow T_{i+1}$ is a *deduction rule*, i.e. an induced step, a generalised contraction or a generalised expansion.

A *lexicon* is a finite set of triples $w : T :: m$, where w is a word, T a type and m a meaning expression. The pairs $w : T$, called *entries*, are sufficient for syntactic analysis. Following [Lambek, 2008], such a set of pairs is called a *dictionary*.

Intuitively, the basic types represent grammatically important notions. For example, the basic types $\boldsymbol{s}$ and $\boldsymbol{n}$ stand for sentences and noun phrases respectively.

A string of words $w_1, \dots, w_n$ is a sentence, if there are entries $w_1 : T_1, \dots, w_n : T_n$ such that $T_1 \dots T_n \rightarrow \boldsymbol{s}$. Similarly, it is a noun phrase, if $T_1 \dots T_n \rightarrow \boldsymbol{n}$.

For example, the sentences *The boys left* and *The boys did not leave* are recognised by a pregroup grammar with a dictionary containing, among others,

the entries

all/the	: $\boldsymbol{n}_2\boldsymbol{c}_2{}^{\ell}$	*did*	: $\boldsymbol{n}_2{}^{r}\boldsymbol{s}\boldsymbol{i}^{\ell}\boldsymbol{d}$
boys	: $\boldsymbol{c}_2$	*not*	: $\boldsymbol{d}^{r}\boldsymbol{i}\boldsymbol{i}^{\ell}\boldsymbol{d}$
left	: $\boldsymbol{n}_2{}^{r}\boldsymbol{s}\boldsymbol{n}^{\ell}$	*leave*	: $\boldsymbol{d}^{r}\boldsymbol{i}\boldsymbol{n}^{\ell}$

The basic type $\boldsymbol{c}_1$ stands for the singular and $\boldsymbol{c}_2$ for the plural of count nouns, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ for the singular and the plural of noun phrases. Moreover, $\boldsymbol{n}$ is the type of the noun phrase when the number does not matter, $\boldsymbol{s}$ is the sentence type[1] and $\boldsymbol{i}$ the type of the infinitive. The type $\boldsymbol{d}$ is a 'dummy' type. It incorporates the indices of phrase structure grammars into the syntactic types of pregroup grammars. Finally, $\boldsymbol{c}_1 \leq \boldsymbol{n}_1$, $\boldsymbol{c}_2 \leq \boldsymbol{n}_2$ and $\boldsymbol{n}_2 \leq \boldsymbol{n}$ so that $\boldsymbol{c}_2\boldsymbol{n}_2{}^{r} \to \epsilon$, $\boldsymbol{n}^{\ell}\boldsymbol{c}_2 \to \epsilon$ etc. are generalised contractions.

Derivations are presented in the form of a graph, for example

$$r_0 = \quad \begin{array}{ccccc} \textit{all} & & \textit{boys} & & \textit{left} \\ \boldsymbol{n}_2 & \boldsymbol{c}_2{}^{\ell} & \boldsymbol{c}_2 & \boldsymbol{n}_2{}^{r} & \boldsymbol{s} \\ & \cup & \cup & \cup & \downarrow \\ & & & & \boldsymbol{s} \end{array} \tag{1}$$

$$r_1 = \quad \begin{array}{cccccccccccc} \textit{the} & & \textit{boys} & & \textit{did} & & & & \textit{not} & & & \textit{leave} \\ \boldsymbol{n}_2 & \boldsymbol{c}_2{}^{\ell} & \boldsymbol{c}_2 & \boldsymbol{n}^{r} & \boldsymbol{s} & \boldsymbol{i}^{\ell} & \boldsymbol{d} & \boldsymbol{d}^{r} & \boldsymbol{i} & \boldsymbol{i}^{\ell} & \boldsymbol{d} & \boldsymbol{d}^{r}\ \boldsymbol{i} \\ & \cup & \cup & \cup & \downarrow & \cup & \cup & \cup & & \cup & \cup & \cup \\ & & & & \boldsymbol{s} & & & & & & & \end{array} \tag{2}$$

The underlinks mark the generalised contractions. Derivations that use only generalised contractions are called *reductions*. It follows from a theorem in [Lambek, 1999] that every derivation to a basic type is equivalent to a reduction.

Dictionary entries $w : T$ are sufficient to recognise grammatical strings. To define semantics for pregroup grammars, a meaning expression has to be added to each dictionary entry.

The lexical entries below give two versions of the meaning expressions, namely vectors in the third column and functions in the fourth column. We recall that V and its dual space, V^*, are isomorphic and that the basis vectors of $W \otimes V^*$ are the vectors $b_i \otimes a_j$, where the b_i's vary over the basis vectors of W and the a_j's over those of V.[2] Moreover, $W \otimes V^*$ and $V^* \otimes W$ are isomorphic. A vector v in space V is identified with a linear map $\overline{v} : I \to V$, where I is a fixed one-dimensional space. A linear map $f : V \to W$ is identified with a matrix, i.e. a vector $\ulcorner f \urcorner \in W \otimes V^*$. A linear map $g : V_1 \otimes V_2 \to W$ is identified with the bilinear map $(v_1, v_2) \mapsto g(v_1 \otimes v_2) \in W$, which we also denote g. These identifications rewrite the vectors of the third column of (2) as vectors or maps in the fourth column. The isomorphism in Section 4 gives a more profound connection: The entries in the fourth column are two-sorted functions, namely functions that take elements and sets as arguments and return elements and

[1] We ignore the tense here.

[2] The 'tensor product' of two vectors is the same as the outer product of the vectors.

sets as values. In particular, a 'function' with no arguments can be either an element or a set. The connection between the two versions and how to get from one to the other is the subject of the rest of this paper.

$$
\begin{array}{lll}
\mathit{all} & : \boldsymbol{n}_2\,{\boldsymbol{c}_2}^{\ell} & :: I \xrightarrow{\overline{\texttt{all}}} E \otimes E^* \qquad \texttt{all}(x) \\
\mathit{boys} & : \boldsymbol{c}_2 & :: I \xrightarrow{\overline{\texttt{boy}}} E \qquad \texttt{boy} \\
\mathit{left} & : {\boldsymbol{n}_2}^{r}\boldsymbol{s} & :: I \xrightarrow{\overline{\texttt{leave}}} E^* \otimes S \qquad \texttt{leave}(z) \\
\mathit{did} & : {\boldsymbol{n}_2}^{r}\boldsymbol{s}\boldsymbol{i}^{\ell}\boldsymbol{d} & :: I \xrightarrow{\overline{\texttt{do}}} E^* \otimes S \otimes S^* \otimes E \qquad \texttt{do}(y_2)\ \mathit{id}(y_1) \\
\mathit{not} & : \boldsymbol{d}^{r}\boldsymbol{i}\boldsymbol{i}^{\ell}\boldsymbol{d} & :: I \xrightarrow{\overline{\texttt{not}}} E^* \otimes S \otimes S^* \otimes E \qquad \texttt{not}_\Omega(z_2)\ \mathit{id}(z_1) \\
\mathit{leave} & : \boldsymbol{d}^{r}\boldsymbol{i}\boldsymbol{n}^{\ell} & :: I \xrightarrow{\overline{\texttt{leave}}} E^* \otimes S \otimes E^* \qquad \texttt{leave}(x_1, x_2)
\end{array}
$$

Here, E and S are distinguished vector spaces. The space $E = V_U$ is the finite dimensional space generated by the set U. The elements of U, i.e. the basis vectors of E, are called *individuals*. The space $S = V_\Omega$ is two-dimensional. Its basis vectors, the elements of the set $\Omega = \{\top, \bot\}$, are called *truth-values*. The vector $\top = (1, 0)$ stands for *true*, $\bot = (0, 1)$ for *false*, $\overrightarrow{0} = (0, 0)$ stands for *no truth values* and the vector $\overrightarrow{1} = (1, 1) \in S$ for *partly true and partly false*.

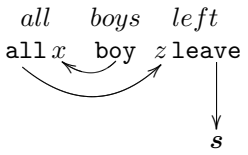
The syntactical type determines the space of the meaning vector. Each simple type corresponds to a factor of the tensor product and vice-versa. The basic types $\boldsymbol{c}_i, \boldsymbol{n}_i, \boldsymbol{d}$ correspond to the space E and the types $\boldsymbol{s}, \boldsymbol{i}$ to S. Right and left adjoints of basic types correspond to the dual spaces, for example

$$
\begin{array}{ccccccccccccc}
\boldsymbol{n}_2 & & {\boldsymbol{c}_2}^{\ell} & & {\boldsymbol{n}_2}^{r} & & \boldsymbol{s} & & \boldsymbol{i}^{\ell} & & \boldsymbol{d} & \boldsymbol{c}_2 & {\boldsymbol{n}_2}^{r} & & \boldsymbol{s} \\
E & \otimes & E^* & & E^* & \otimes & S & \otimes & S^* & \otimes & E & E & E^* & \otimes & S
\end{array}
$$

Suppose the meaning expressions of words are given as functions. The function symbols `leave`, `all`, $\texttt{not}_\Omega$, *id* and `boy` correspond to basic types, the variables to right or left adjoints of basic types

$$
\begin{array}{cccccc}
\boldsymbol{n}_2 & {\boldsymbol{c}_2}^{\ell} & & \boldsymbol{c}_2 & {\boldsymbol{n}_2}^{r} & \boldsymbol{s} \\
\texttt{all} & x & & \texttt{boy} & z & \texttt{leave}\,.
\end{array}
$$

In order to compute the meaning expression of a sentence replace every simple type in the reduction by the corresponding function symbol and substitute according to the links. For example,



defines the substitutions $[z|\,\texttt{all}(x)]$ and $[x|\,\texttt{boy}]$. The expression obtained is

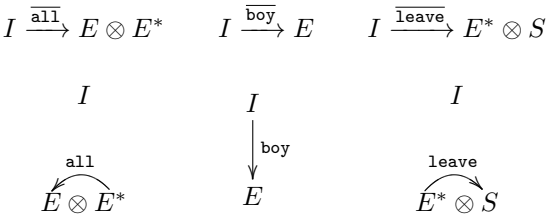
$$\texttt{leave}(z)[z|\texttt{all}(x)][x|\texttt{boy}] = \texttt{leave}(\texttt{all}(\texttt{boy}))\,.$$

Apply the same method to the sentence *The boys did not leave.* Substitution according to the links results in the compound expression

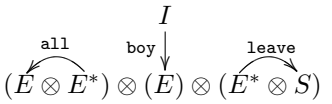
$$\texttt{do}(\texttt{not}_\Omega(\texttt{leave}(\texttt{the}(\texttt{boy})))\,. \tag{3}$$

On the other hand, to construct the vector representing a sentence, form the tensor product of the word vectors and compose it with the reduction.[3] The composition is legitimate, because reductions define *2*-cells in the free compact *2*-category and in particular linear maps of vector spaces. The meaning expressions in vector notation identify with linear maps of domain I. Therefore they are morphisms of a compact closed category and as such are representable by graphs. Compute the meaning expression of a sentence by the following instructions

- replace the morphisms by their graphs

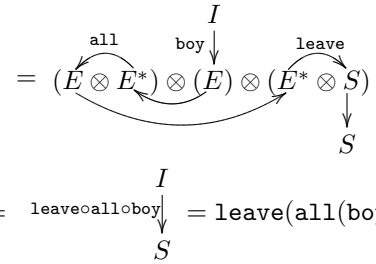


- join the graphs by the tensor product



- compose this product with the linear map r'_0 defined by the reduction (1)

$$m(all\, boys\, left) = r'_0 \circ (\overline{\texttt{all}} \otimes \overline{\texttt{boy}} \otimes \overline{\texttt{leave}})$$



The logical content of *all* is captured by the postulate

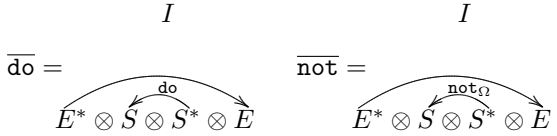
$$\texttt{all} = id_E\ ,$$

[3] In a first approach, we could take the meaning of a sentence to be the tensor product of the meanings. But then every string of words would have a meaning.

from which it follows that

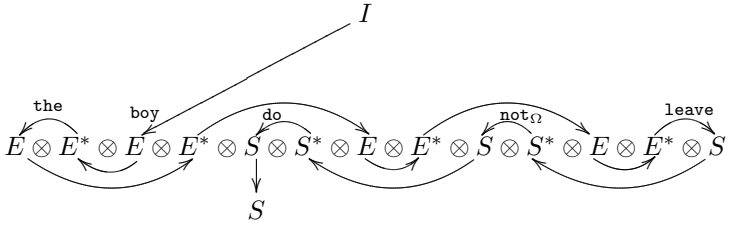
$$\texttt{leave}(\texttt{all}(\texttt{boy})) = \texttt{leave}(\texttt{boy})\,. \tag{4}$$

The graph of $\overline{\texttt{the}}$ is that of $\overline{\texttt{all}}$ except for the label. The graphical representation of the vectors $\overline{\texttt{do}} : I \to E^* \otimes S \otimes S^* \otimes E$ and $\overline{\texttt{not}} : I \to E^* \otimes S \otimes S^* \otimes E$ are



Again, we form the tensor product and compose it with the linear map corresponding to the reduction (2)

$$r'_1 \circ (\overline{\texttt{the}} \otimes \overline{\texttt{boy}} \otimes \overline{\texttt{do}} \otimes \overline{\texttt{not}} \otimes \overline{\texttt{leave}}) =$$



$$= \texttt{do}(\texttt{not}_\Omega(\texttt{leave}(\texttt{the}(\texttt{boy}))))$$

The righthand expression in the equality above coincides with the expression (3) computed by substitution, confirming the slogan 'substitution in logic is composition in categories'.

The auxiliary *do* in the negated sentence has only a syntactic purpose, it does not influence the meaning. This is expressed by postulate

$$\texttt{do} = id_S\,. \tag{5}$$

$$\texttt{do}(\texttt{not}_\Omega(\texttt{leave}(\texttt{the}(\texttt{boy})))) = \texttt{not}_\Omega(\texttt{leave}(\texttt{the}(\texttt{boy})))\,. \tag{6}$$

Similarly, the logical content of the word *not* is captured by requiring that $\texttt{not}_\Omega : S \to S$ is the negation of truth-values

$$\texttt{not}_\Omega(\top) = \bot \quad \texttt{not}_\Omega(\bot) = \top\,. \tag{7}$$

The semantics is clearly compositional in both versions. The vector approach does not require variables and is invariant under change of order. The latter is guaranteed by the following lemma

Lemma 1. *The meaning vector of a sentence is strictly invariant under permutation of the word meanings.*

Proof. Indeed, the tensor product is symmetric up to isomorphism. If we compose the tensor product of the word vectors with the permutation σ, we must compose the inverse σ^{-1} with the reduction r to obtain a reduction of the permuted tensor product. Hence

$$r \circ (\overline{\mathtt{w}}_1 \otimes \ldots \otimes \overline{\mathtt{w}}_n) = (r \circ \sigma^{-1}) \circ (\sigma \circ (\overline{\mathtt{w}}_1 \otimes \ldots \otimes \overline{\mathtt{w}}_n)) = (r \circ \sigma^{-1}) \circ (\overline{\mathtt{w}}_{\sigma(1)} \otimes \ldots \otimes \overline{\mathtt{w}}_{\sigma(n)}) \,.$$

□

Variable-free functional expressions and invariance of the expression under change of order in the arguments are the first two postulates concerning compositional semantics formulated in [Kracht, 2007]. The third and last postulate, which says that arguments may not be repeated, is also satisfied. The example of the relative pronoun *who* in Section 7 illustrates how multiple occurrences are avoided.

Examples

The value of the meaning-expression of a sentence depends on the value of the meaning expressions of the words. Consider two examples and M_2. The former represents a situation where *all* of the *boys* present in the room *left*, whereas in the latter, *the boys did not leave.* Let $\mathtt{boy} = \{a_1, \ldots, a_m\}$ be the set of boys in question. The interpretations of `leave` differ in M_1 and M_2, namely

$$\begin{array}{ll} M_1 & M_2 \\ \mathtt{leave}_1(a_i) = \top & \mathtt{leave}_2(a_i) = \bot \qquad 1 \leq i \leq m \,. \end{array} \tag{8}$$

We refer the reader to Section 7 to see that

$$\begin{array}{ll} \mathtt{leave}_1(\mathtt{boy}) = \top & \mathtt{not}_\Omega(\mathtt{leave}_1(\mathtt{boy})) = \bot \\ \mathtt{leave}_2(\mathtt{boy}) = \bot & \mathtt{not}_\Omega(\mathtt{leave}_2(\mathtt{boy})) = \top \,. \end{array}$$

In a third situation M_3, only the boys a_1 and a_2 left and the others stayed behind.

$$\begin{array}{ll} M_3 & \\ \mathtt{leave}_3(a_i) = \top & \mathtt{leave}_3(a_i) = \bot \\ 1 \leq i \leq 2 & 3 \leq i \leq m \,. \end{array}$$

In this case,

$$\mathtt{leave}_3(\mathtt{boy}) = \{\top, \bot\} \text{ and } \mathtt{not}_\Omega(\mathtt{leave}_3(\mathtt{boy})) = \{\top, \bot\} \,.$$

3 An algorithm from functional semantics to vector semantics

The algorithm translates function expressions to vector expressions, preserving the meaning of grammatical strings of words. It decomposes into five routines, called 'Map', 'Glue', 'Diagonal', 'Name', 'Rearrange'. They are executed in

that order, but the routines 'Glue', 'Diagonal' and 'Rearrange' do not always apply. The first example concerning sentence negation does not use the routine 'Diagonal'. The second example uses all five routines and concerns the relative pronoun.

3.1 The routines 'Map', 'Glue' , 'Name', 'Rearrange'

Recall the entries

$$\begin{array}{ll} \mathit{all} : \boldsymbol{n}_2\boldsymbol{c}_2{}^{\ell} :: \mathtt{all}(x) & \mathit{not} : \boldsymbol{d}^r\boldsymbol{i}\boldsymbol{i}^{\ell}\boldsymbol{d} :: \mathtt{not}_\Omega(z_2)\ \mathit{id}(z_1) \\ \mathit{did} : \boldsymbol{n}^r\boldsymbol{s}\boldsymbol{i}^{\ell}\boldsymbol{d} :: \mathtt{do}(y_2)\ \mathit{id}(y_1) & \mathit{leave} : \boldsymbol{d}^r\boldsymbol{i} :: \mathtt{leave}(z) \end{array}$$

and the correspondence of simple types and symbols

$\boldsymbol{n}_2$	$\boldsymbol{c}_2{}^{\ell}$	$\boldsymbol{n}_2{}^r$	$\boldsymbol{s}$	$\boldsymbol{i}^{\ell}$	$\boldsymbol{d}$	$\boldsymbol{d}^r$	$\boldsymbol{i}$	$\boldsymbol{i}^{\ell}$	$\boldsymbol{d}$	$\boldsymbol{d}^r$	$\boldsymbol{i}$
all	x	y_1	do	y_2	id	z_1	$\mathtt{not}_\Omega$	z_2	id	z	leave

Routine 'Map'

- read each function as map with domain and codomain, respecting the correspondence of simple types and spaces

$$\begin{array}{cc} \boldsymbol{c}_2{}^{\ell}\ \ \boldsymbol{n}_2 & \boldsymbol{i}^{\ell}\ \ \boldsymbol{s} \\ x\ \ \mathtt{all} & y_2\ \ \mathtt{do} \\ I \xrightarrow{\mathtt{all}} E & S \xrightarrow{\mathtt{do}} S \end{array} \qquad \begin{array}{cc} \boldsymbol{n}_2{}^r\ \ \boldsymbol{d} \\ y_1\ \ \mathit{id} \\ E \xrightarrow{\mathit{id}} E \end{array}$$

$$\begin{array}{cc} \boldsymbol{i}^{\ell}\ \ \boldsymbol{i} & \boldsymbol{d}^r\ \ \boldsymbol{d} \\ z_2\ \ \mathtt{not}_\Omega & z_1\ \ \mathit{id} \\ S \xrightarrow{\mathtt{not}_\Omega} S & E \xrightarrow{\mathit{id}} E \end{array} \qquad \begin{array}{cc} \boldsymbol{d}^r\ \ \boldsymbol{i} \\ x\ \ \mathtt{leave} \\ E \xrightarrow{\mathtt{leave}} S \end{array}$$

Routine 'Glue'

- glue the maps arising in the same lexical entry together with the help of the tensor product, recopying the correspondence of simple types and spaces

$$\begin{array}{ccc} S \otimes E & \xrightarrow{\mathtt{do}\otimes \mathit{id}} & S \otimes E \\ \boldsymbol{i}^{\ell}\ \ \boldsymbol{n}_2{}^r & & \boldsymbol{s}\ \ \boldsymbol{d} \end{array} \qquad \begin{array}{ccc} S \otimes E & \xrightarrow{\mathtt{not}_\Omega\otimes \mathit{id}} & S \otimes E \\ \boldsymbol{i}^{\ell}\ \ \boldsymbol{d}^r & & \boldsymbol{i}\ \ \boldsymbol{d} \end{array}$$

Routine 'Name'

- replace the linear maps by their names, that is to say the vector designating the matrix associated to a linear map. Keep track of the correspondence between factors and simple types

$$\begin{array}{ll} I \xrightarrow{\ulcorner \mathtt{boy} \urcorner = \mathtt{boy}} & E \\ & \boldsymbol{c}_2 \\ I \xrightarrow{\ulcorner \mathtt{leave} \urcorner} & E^* \otimes S \\ & \boldsymbol{d}^r\ \ \boldsymbol{i} \end{array} \qquad \begin{array}{ll} I \xrightarrow{\ulcorner \mathtt{do}\otimes \mathit{id} \urcorner} & E^* \otimes S^* \otimes S \otimes E \\ & \boldsymbol{n}_2{}^r\ \ \boldsymbol{i}^{\ell}\ \ \boldsymbol{s}\ \ \boldsymbol{d} \\ I \xrightarrow{\ulcorner \mathtt{not}_\Omega\otimes \mathit{id} \urcorner} & E^* \otimes S^* \otimes S \otimes E \\ & \boldsymbol{d}^r\ \ \boldsymbol{i}^{\ell}\ \ \boldsymbol{i}\ \ \boldsymbol{d} \end{array}$$

Routine 'Rearrange'

- rearrange the factors in the order in which the simple types occur in the syntactical type. To do so, compose the name with the appropriate permutation

$$\begin{array}{lll} \overline{\texttt{all}} & = c \circ \ulcorner \texttt{all} \urcorner : I \to E^* \otimes E & \\ & & \boldsymbol{c_2}^{\ell} \quad \boldsymbol{n_2} \\ \overline{\texttt{do}} & = c' \circ \ulcorner \texttt{do} \otimes id \urcorner : I \to E^* \otimes S \otimes S^* \otimes E & \\ & & \boldsymbol{n_2}^{r} \quad \boldsymbol{s} \quad \boldsymbol{i}^{\ell} \quad \boldsymbol{d} \\ \overline{\texttt{not}} & = c' \circ \ulcorner \texttt{not}_\Omega \otimes id \urcorner : I \to E^* \otimes S \otimes S^* \otimes E & \\ & & \boldsymbol{d}^{r} \quad \boldsymbol{i} \quad \boldsymbol{i}^{\ell} \quad \boldsymbol{d} \end{array}$$

where $c : E \otimes E^* \to E^* \otimes E$ permutes the two factors and $c' : E^* \otimes S^* \otimes S \otimes E \to E^* \otimes S \otimes S^* \otimes E$ switches the second and the third factor.

3.2 The routine 'Diagonal'

The next example involves the relative pronoun *who* and uses the routine 'Diagonal'. The lexical entries concerning the noun-phrase *the boys who left* are

$$\begin{array}{llll} \textit{the} & : \boldsymbol{n_2 c_2}^{\ell} :: \texttt{the}(x) & \textit{who} : \boldsymbol{c_2}^{r} \boldsymbol{c_2}\, \boldsymbol{s}^{\ell} \boldsymbol{n_2} :: \texttt{who}(x_1, x_2)\ id(x_1) \\ \textit{boys} & : \boldsymbol{c_2} \quad :: \texttt{boy} & \textit{left} \ : \boldsymbol{n_2}^{r} \boldsymbol{s} \quad :: \texttt{leave}(z) \end{array} .$$

Note that the interpretation of $\textit{who} : \boldsymbol{c_2}^{r} \boldsymbol{c_2}\, \boldsymbol{s}^{\ell} \boldsymbol{n_2}$ has a property not encountered previously. A variable, namely x_1, occurs in more than one functional expression. The algorithm constructs the corresponding vector

$$\overline{\texttt{who}} : I \to E^* \otimes E \otimes S^* \otimes E$$

with the help of the new routine 'Diagonal' below.

The correspondence between symbols and simple types is

$$\begin{array}{cccccccccc} \texttt{the} & x & \texttt{boy} & x_1 & \texttt{who} & id & x_2 & z & \texttt{leave} \\ \boldsymbol{n_2} & \boldsymbol{c_2}^{\ell} & \boldsymbol{c_2} & \boldsymbol{c_2}^{r} & \boldsymbol{c_2} & \boldsymbol{n_2} & \boldsymbol{s}^{\ell} & \boldsymbol{n_2}^{r} & \boldsymbol{s} \end{array} .$$

using the reduction, compute the meaning expression of the noun-phrase *the boys who left*

$$\texttt{the}(\texttt{who}(\texttt{boy}, \texttt{leave}(id(\texttt{boy})))) = \texttt{the}(\texttt{who}(\texttt{boy}, \texttt{leave}(\texttt{boy}))) .$$

The translation algorithm first executes

'Map' 'Glue'

$$\begin{array}{cccccccccccc} E \otimes S & \xrightarrow{\texttt{who}} & E & E & \xrightarrow{id} & E & E & \otimes E & \otimes S & \xrightarrow{id_E \otimes \texttt{who}} & E & \otimes E \\ \boldsymbol{c_2}^{r} & \boldsymbol{s}^{\ell} & \boldsymbol{c_2} & \boldsymbol{c_2}^{r} & & \boldsymbol{n_2} & \boldsymbol{c_2}^{r} & \boldsymbol{c_2}^{r} & \boldsymbol{s}^{\ell} & & \boldsymbol{n_2} & \boldsymbol{c_2} \end{array} .$$

The simple type $\boldsymbol{c_2}^{r}$ occurs twice in the tensor product, whereas it has only one occurrence in the lexical entry. Equivalently, the variable x_1 'lives' in two different factors of the argument space where there should be only one. This is

[Kracht, 2007]'s requirement that arguments may not be repeated. The solution here is the *diagonal*, i.e. the linear map $d_E : E \to E \otimes E$ that maps each basis vector a to $a \otimes a$.

Routine 'Diagonal'
- form the tensor product of the diagonal d_E with the identity of the other factor(s) so that the 'adjusted' the diagonal can be composed with $id_E \otimes \texttt{who}$

$$\begin{array}{c} E \otimes S \\ \boldsymbol{c}_2{}^r\ \boldsymbol{s}^\ell \end{array} \xrightarrow{d_E \otimes id_S} \begin{array}{c} E \otimes E \otimes S \\ \boldsymbol{c}_2{}^r\ \ \boldsymbol{c}_2{}^r\ \ \boldsymbol{s}^\ell \end{array} \qquad \begin{array}{c} E \otimes S \\ \boldsymbol{c}_2{}^r\ \boldsymbol{s}^\ell \end{array} \xrightarrow{(id_E \otimes \texttt{who}) \circ (d_E \otimes id_S)} \begin{array}{c} E \otimes E \\ \boldsymbol{c}_2\ \ \boldsymbol{n}_2 \end{array} .$$

The next two routines are 'Name' and 'Rearrange', namely

$$I \xrightarrow{\ulcorner (id \otimes \texttt{who}) \circ (d_E \otimes id_S) \urcorner} \begin{array}{c} E^* \otimes S^* \otimes E \otimes E \\ \boldsymbol{c}_2{}^r\ \ \boldsymbol{s}^\ell\ \ \boldsymbol{n}_2\ \ \boldsymbol{c}_2 \end{array}$$

and

$$\overline{\texttt{who}} = c \circ \ulcorner (id \otimes \texttt{who}) \circ (d_E \otimes id_{S \otimes E}) \urcorner : I \to E^* \otimes E \otimes S^* \otimes E\,, \qquad (9)$$

where

$$c : \begin{array}{c} E^* \otimes S^* \otimes E \otimes E \\ \boldsymbol{c}_2{}^r\ \ \boldsymbol{s}^\ell\ \ \boldsymbol{n}_2\ \ \boldsymbol{c}_2 \end{array} \to \begin{array}{c} E^* \otimes E \otimes S^* \otimes E \\ \boldsymbol{c}_2{}^r\ \ \boldsymbol{c}_2\ \ \boldsymbol{s}^\ell\ \ \boldsymbol{n}_2 \end{array} .$$

makes two exchanges, namely first between the last two factors and then between the third and the second factor.

3.3 Noun-phrases formed with the relative pronoun

The strings *boys who left* and *the boys who left* have both a reduction to a basic type, namely

$$r_3 = \begin{array}{ccccccccc} & \textit{boys} & & & \textit{who} & & & \textit{left} & \\ (& \boldsymbol{c}_2 &) & (\boldsymbol{c}_2{}^r & \boldsymbol{c}_2 & \boldsymbol{s}^\ell & \boldsymbol{n}_2) & (\boldsymbol{n}_2{}^r & \boldsymbol{s}) \\ & & & & \downarrow & & & & \\ & & & & \boldsymbol{c}_2 & & & & \end{array}$$

and

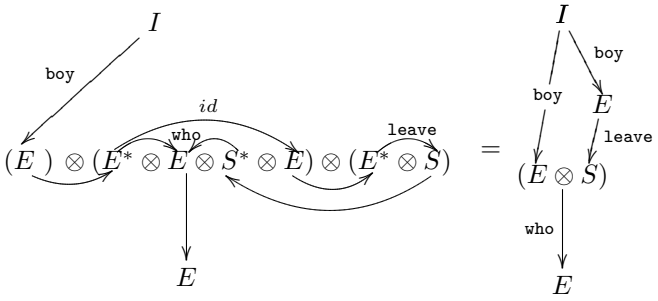
$$r_4 = \begin{array}{ccccccccccc} & \textit{the} & & \textit{boys} & & & \textit{who} & & \textit{left} & & \\ (\boldsymbol{n}_2 & \boldsymbol{c}_2{}^\ell) & (& \boldsymbol{c}_2 &) & (\boldsymbol{c}_2{}^r\ \boldsymbol{c}_2 & \boldsymbol{s}^\ell & \boldsymbol{n}_2) & (\boldsymbol{n}_2{}^r & \boldsymbol{s}) \\ \downarrow & & & & & & & & & & \\ \boldsymbol{n}_2 & & & & & & & & & & \end{array} .$$

Finally, the equality (9) above defines the representation of $\overline{\texttt{who}}$ by the graph

$$\overline{\texttt{who}} : I \to E^* \otimes E \otimes S^* \otimes E = \begin{array}{c} I \\ id \\ \texttt{who} \\ E^* \otimes E \otimes S^* \otimes E \end{array} .$$

Compose the tensor product of the word vectors with the reduction to obtain

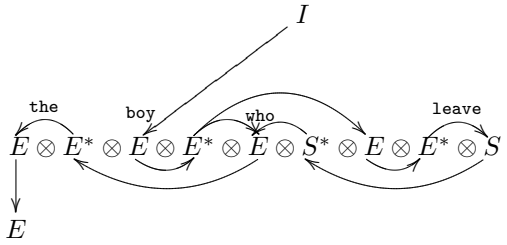
$$m(\textit{boys who left}) = r'_3 \circ (\overline{\texttt{boy}} \otimes \overline{\texttt{who}} \otimes \overline{\texttt{leave}}) \;=$$



$$= \texttt{who} \circ (\texttt{boy} \otimes (\texttt{leave} \circ \texttt{boy})) = \texttt{who}(\texttt{boy}, \texttt{leave}(\texttt{boy}))$$

and

$$m(\textit{the boys who left}) = r'_4 \circ (\overline{\texttt{the}} \otimes \overline{\texttt{boy}} \otimes \overline{\texttt{who}} \otimes \overline{\texttt{leave}}) =$$



$$= \texttt{the}(\texttt{who}(\texttt{boy}, \texttt{leave}(\texttt{boy}))$$

4 The isomorphism

In Information Retrieval Systems, words are represented by vectors of some fixed finite dimensional space. The coordinates of the vector are in general obtained by a frequency count of occurrences of words in documents. Hence, without loss of generality, we may assume that the coordinates belong to the real interval $I = [0, 1]$. The linear order of real numbers induces a distributive and implication-complemented lattice structure on I

$$\begin{gathered}\alpha \vee \beta = \max\{\alpha, \beta\} \\ \alpha \wedge \beta = \min\{\alpha, \beta\} \\ \alpha \to \beta = \sup\{\gamma \in I : \; \alpha \wedge \gamma \leq \beta\} \\ \neg\alpha = \alpha \to 0\,.\end{gathered}$$

Clearly, $\alpha \to \beta = 1$ if and only if $\alpha \leq \beta$. I becomes a semi-ring with the operations

$$\begin{aligned}\alpha + \beta &= \alpha \vee \beta \\ \alpha \cdot \beta &= \alpha \wedge \beta\,.\end{aligned}$$

An *I-space* is an arbitrary finite-dimensional semi-module over I. The semi-ring I is identified with a one-dimensional space, also denoted I. The sum of vectors

$v = \alpha_1 a_1 + \cdots + \alpha_n a_n \in V$ and $w = \beta_1 a_1 + \cdots + \beta_n a_n \in V$ and the multiplication with a scalar $\alpha \in I$ are defined as usual, namely

$$v + w = (\alpha_1 + \beta_1)a_1 + \cdots + (\alpha_n + \beta_n)a_n$$
$$\alpha w = (\alpha \cdot \beta_1)a_1 + \cdots + (\alpha \cdot \beta_n)a_n \,.$$

In particular, $\alpha(\alpha v) = (\alpha \cdot \alpha)v = \alpha v = (\alpha + \alpha)v = \alpha v + \alpha v$. An easy computation shows that every I-space V has a unique set A of basis vectors. We write $V = V_A$.

Orthogonality, inner and outer product of vectors are defined like in a vector space over the field of real numbers. As all coordinates are non-negative, vectors are orthogonal in the semi-module over I if and only if they are orthogonal in the vector space over the field of real numbers. The norm[4] of a vector is given by

$$\|\alpha_1 a_1 + \cdots + \alpha_n a_n\| = \sup\{|\alpha_i|,\ 1 \leq i \leq n\} = \sum_{i=1}^{n} \alpha_i \,.$$

The definitions of linear maps, matrix representation of linear maps and multiplication of matrices also remain unchanged. For example, a linear map $f : V_A \to V_B$ satisfies

$$\begin{array}{rcl} f(\overrightarrow{0}) & = & \overrightarrow{0} \\ f(\sum_{i=1}^{n} \alpha_i a_i) & = & \sum_{i=1}^{n} \alpha_i f(a_i) \,. \end{array}$$

Clearly, every linear map is defined by its values on the basis vectors. Its matrix is identified with a vector $\ulcorner f \urcorner \in V_A^* \otimes V_B$, the tensor product of the dual space V_A^* of V_A and V_B. The coordinates of $\ulcorner f \urcorner$, i.e. the entries of the matrix, are determined as usual. Indeed, let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$. The dual space V_A^* is isomorphic to V_A and therefore the vectors $a_i \otimes b_j,\ 1 \leq i \leq n,\ 1 \leq j \leq m$ form a basis of $V_A^* \otimes V_B$. Then

$$\ulcorner f \urcorner = \sum_{i,j} \gamma_{i,j}(a_i \otimes b_j) \,,$$

where $\gamma_{i,j}$ is the j-th coordinate of the vector $f(a_i)$.

Examples of linear maps are the identity id_{V_A} and the *diagonal* $d_{V_A} : V_A \to V_A \otimes V_A$, namely the linear map that sends a basis vector to the outer product with itself

$$d_{V_A}(a) = a \otimes a, \text{ for } a \in A \,.$$

Note that $d_{V_A}(v) \neq v \otimes v$ if v is not a basis vector. Indeed,

$$d_{V_A}(\sum_{i=1}^{n} \alpha_i a_i) = \sum_{i=1}^{n} \alpha_i (a_i \otimes a_i)$$

whereas

$$(\sum_{i=1}^{n} \alpha_i a_i) \otimes (\sum_{i=1}^{n} \alpha_i a_i) = \sum_{1 \leq i,j \leq n} (\alpha_i \alpha_j)(a_i \otimes a_j) \,.$$

[4] norm advocated for context-theoretical semantics in [Clarke, 2007]

Linear maps of I-spaces are closely related to the functions interpreting lexical entries of pregroup grammars in two-sorted first order predicate logic.

Definition 1. *A function $g : A \to B$ is* two-sorted *if it maps elements and subsets of A to elements or subsets of B and satisfies*

$$\begin{array}{rcl} g(\{a\}) & = & g(a) \text{ for } a \in A \\ g(\emptyset) & = & \emptyset \\ g(X \cup Y) & = & g(X) \cup g(Y) \text{ for } X, Y \subseteq A\,. \end{array} \tag{10}$$

Obviously, a two-sorted function defined on a finite set is determined by the values on elements. Examples are

two-sorted diagonal	*two-sorted identity*
$d(a) = \langle a, a\rangle$	$id(a) = a,$
$d(A) = \{\langle a, a\rangle : a \in A\}$	$id(A) = A\,.$

Given an I-space V_A, generated by a set of basis vectors $A = \{a_1, \ldots, a_n\}$, define a map $\mathcal{I}_A$ from V_A to the set of subsets of A by

$$\mathcal{I}_A(\beta_1 a_1 + \cdots + \beta_n a_n) = \{a_i \in A \,:\, \beta_i \neq 0\}\ .$$

From the definition follows that

$$\begin{array}{rcl} \mathcal{I}_A(v + w) & = & \mathcal{I}_A(v) \cup \mathcal{I}_A(w) \\ \mathcal{I}_A(\overrightarrow{0}) & = & \emptyset\,. \end{array}$$

Moreover, define

$$\mathcal{I}(V_A) = A$$

and, for every linear map $f : V_A \to V_B$, the two-sorted function $\mathcal{I}(f) : \mathcal{I}(V_A) \to \mathcal{I}(V_B)$ determined by

$$\mathcal{I}(f)(a) = \mathcal{I}_B(f(a)), \text{ for all } a \in A\,. \tag{11}$$

Conversely, given a set $A = \{a_1, \ldots, a_m\}$, define a map $\mathcal{J}_A$ from the set of subsets of A to the space V_A thus

$$\mathcal{J}_A(X) = \sum_{i=1}^{n} \alpha_i a_i, \text{ where } \alpha_i = 1 \text{ if } a_i \ \in X \text{ and } \alpha_i = 0 \text{ else}\,.$$

Clearly, for $X, Y \subseteq A$

$$\begin{array}{rcl} \mathcal{J}_A(X \cup Y) & = & \mathcal{J}_A(X) + \mathcal{J}_A(Y) \\ \mathcal{J}_A(\emptyset) & = & \overrightarrow{0}\,. \end{array}$$

Finally, extend $\mathcal{J}_A$ to elements by

$$\mathcal{J}_A(a) = \mathcal{J}_A(\{a\})\,.$$

Given a set A and a two-sorted function $g : A \to B$, let

$$\mathcal{J}(A) = V_A$$

and $\mathcal{J}(g) : \mathcal{J}(A) \to \mathcal{J}(B)$ be the linear map defined on the basis A by

$$\mathcal{J}(g)(a) = \mathcal{J}_B(g(a)), \text{ for all } a \in A\,. \tag{12}$$

Lemma 2. *For every subset $X \subseteq A$ and every two-sorted map $g : A \to B$ the following holds*

$$\begin{array}{rcl} \mathcal{I}_A \circ \mathcal{J}_A(X) & = & X \\ (\mathcal{I} \circ \mathcal{J})(g) & = & g\,. \end{array}$$

Moreover, if the coordinates of $v \in V_A$ are 0 or 1 respectively the entries of the matrix of $f : V_A \to V_B$ are 0 or 1, then

$$\begin{array}{rcl} \mathcal{J}_A \circ \mathcal{I}_A(v) & = & v \\ (\mathcal{J} \circ \mathcal{I})(f) & = & f\,. \end{array}$$

Proof. To see the first equality, let $\mathcal{J}_A(X) = \sum_{i=1}^{n} \alpha_i a_i$. By definition of $\mathcal{I}_A$, $a_i \in \mathcal{I}_A(\mathcal{J}_A(X))$ if and only if $\alpha_i \neq 0$. By definition of $\mathcal{J}_A$, the latter holds if and only if $a_i \in X$.

It suffices to show the second equality for elements, because its left and right hand expressions are two-sorted functions. Let $a \in A$. Then

$$\begin{array}{lll} (\mathcal{I} \circ \mathcal{J})(g)(a) & = \mathcal{I}(\mathcal{J}(g))(a) & \\ & = \mathcal{I}_B(\mathcal{J}(g)(a)) & \text{by (11)} \\ & = \mathcal{I}_B(\mathcal{J}_B(g(a)) & \text{by (12)} \\ & = (\mathcal{I}_B \circ \mathcal{J}_B)(g(a)) & \\ & = g(a) & \text{by the first equality}\,. \end{array}$$

The other equalities are proved similarly. □

The equalities $\mathcal{J}_A \circ \mathcal{I}_A(v) = v$ and $(\mathcal{J} \circ \mathcal{I})(f) = f$ do not hold in general. Vectors with coordinates different from 0 and 1 are 'lost' by the map $\mathcal{J}_A \circ \mathcal{I}_A$. Only the restriction of $\mathcal{I}_A$ to vectors with coordinates in $\{0, 1\}$ is a one-one map onto the set of subsets of A.

Theorem 1. *$\mathcal{J}$ and $\mathcal{I}$ preserve identities, commute with composition, identify unions of sets with sums of spaces, products of sets with tensor products of spaces, the two-sorted diagonal with the linear diagonal and graphs of functions with matrices of linear maps.*

The proof is straightforward, but notationally unpleasant, because there are only vectors in vector spaces, whereas the image of a space needs elements and subsets. The next section introduces a better notation for two-sorted functions and we come back to the proof then. In fact, natural language, where two-sorted functions are ubiquitous, also uses them in a 'sloppy' way compared to set-theory.

5 Logic of concepts

I-spaces accommodate both the functional logic of sentences and the propositional logic of semantic vector models. The link between the two is quantum logic of semi-modules over the lattice of real numbers. The point of view presented is taken from [Preller and Prince, 2010].

According to [Rijsbergen, 2004], quantum logic for Information Retrieval Systems refers to the lattice structure of subspaces of a space over the field of real numbers. Every subspace is identified with a projector, namely the linear map that takes its values in the subspace and leaves every vector of the subspace invariant. Subspaces stand for words or combinations of words constructed with the help of the propositional connectives given by

$$\neg E = E^{\perp},\ E \vee F = E \oplus F,\ E \wedge F = E \circ F,\ E \to F = \{v\ :\ F(E(v)) = E(v)\}\ .$$

In general, the lattice structure is not distributive.

The same definitions in an I-space $E = V_A$ result in a distributive lattice of subspaces. Moreover, the order of the real interval I introduces a lattice structure on the vectors of V_A, isomorphic to the lattice structure of subspaces of V_A. The isomorphism $\mathcal{J}$ maps the Boolean algebra of subsets of A to the lattice of subspaces generated by subsets of A. Hence the lattice of subspaces for an arbitrary I-space includes a Boolean sublattice.

In a semantic vector model, words are represented by vectors of a finite dimensional space over the real numbers. One may assume without loss of generality that the coordinates belong to the interval $I = [0, 1]$, because the number of words is finite. There are two kinds of semantic vector models. In one, the basis vectors are identified with words, in the other one with strings of words. In the first, the coordinates of the vectors result from a frequency count in context windows. In the second, they are given by a probability distribution on a set of strings. Both are used for detecting similarity of words or for choosing a particular meaning among the possible meanings of a word.

The string approach seems more appropriate for compositional semantics. In this case, arbitrary vectors can be viewed as concepts and the quantum connectives define the logic of concepts.

Indeed, the logical properties of quantum logic render faithfully the intuitive logical properties of words in the particular case where the I-space is a tensor product of two-dimensional spaces. Words are classified by a thesaurus, be it the mental thesaurus of the speaker or an explicit thesaurus. For a given set of key-words or *basic concepts* $P = \{p_1, \ldots, p_n\}$, the vectors of the tensor product

$$C(P) = V_{\{p_{1\top},\ p_{1\perp}\}} \otimes \ldots \otimes V_{\{p_{n\top},\ p_{n\perp}\}}$$

represent concepts. In particular, the words classified by the thesaurus are Boolean combinations of basic concepts. The sublattice of subspaces generated by basis vectors is isomorphic to the free Boolean algebra $B(P)$ generated by P. The geometrical properties of $C(P)$ reflect faithfully the logical properties of words. Concept spaces are the linguistic analogue to the tensor products

of two-dimensional Hilbert spaces representing compound systems in quantum mechanics.

Note that $V_{\{p_i\top, p_i\bot\}}$ is a notational variant of V_Ω. Moreover, Ω is identified with the two-element Boolean algebra $B(\emptyset)$ of truth values. Hence the isomorphism transforms the Boolean connectives to linear maps. For example, the *truth-value negation*

$$\mathcal{J}(\neg) = \mathtt{not}_\Omega : V_\Omega \to V_\Omega\,,$$

which satisfies

$$\mathtt{not}_\Omega(\overrightarrow{0}) = \overrightarrow{0},\ \mathtt{not}_\Omega(\top) = \bot,\ \mathtt{not}_\Omega(\bot) = \top,\ \mathtt{not}_\Omega(\top + \bot) = \bot + \top\,.$$

Thus, the compact closed category of I-spaces includes both the functional models based on 2-sorted logic and the semantic vector models based on quantum logic.

6 Natural language in two-sorted first order logic

It follows from Section 4 that the functional expressions in a pregroup lexicon are two-sorted functions. Functions in two-sorted first order logic were used in [Preller, 2007] for pregroup semantics. Their relevance for natural language, however, goes back to the fact that two-sorted first order logic is equivalent to second order logic with general models, established in [Benthem and Doets, 1983].

Two-sorted first order logic, 2-FOL, has two sorts, one for elements and one for sets, a primitive relational symbol $\in$ requiring elements on the left and sets on the right, and an equality symbol $=$ accepting either sets on both sides or elements on both sides. Formulae are defined as usual, except that there are two sorts of quantifiers, namely $\exists_x, \forall_x$ for elements and $\exists_X, \forall_X$ for sets. The function(al symbol)s take elements and subsets as arguments and return elements or subsets. The two-sorts are reflect the fact that verbs may take singulars and plurals as arguments. 2-FOL provides a meta-language in which to define and prove the properties of the two-sorted functions interpreting words and grammatical expressions.

We introduce several operators on sets and elements and justify them by their use in natural language.

- The first operation is the *two-sorted product* defining both the ordered pair of elements and the Cartesian product of sets

$$\begin{array}{ll} \mathit{prod}(a,b) & = \langle a,b\rangle \\ \mathit{prod}(a,B) & = \{a\}\times B \\ \mathit{prod}(A,b) & = A\times\{b\} \\ \mathit{prod}(A,B) & = A\times B \end{array}$$

The two-sorted product is the counterpart of the tensor product, also a single operator defined both for vectors and spaces. The two-sorted product provides the argument set for transitive verbs.

- The second operator is the *two-sorted union*. It combines the set-theoretical operators 'unordered pair of elements' and 'union of sets'

$$\begin{aligned}
&\mathtt{and}(a,b) = \{a,b\}\\
&\mathtt{and}(a,B) = \{a\} \cup B\\
&\mathtt{and}(A,b) = A \cup \{b\}\\
&\mathtt{and}(A,B) = A \cup B\,.
\end{aligned}$$

The single notation `and` stems from the meaning of the word *and* in *Eva and Sue* respectively *boys and girls*. It occurs in the lexical entries

$$\mathit{and} : \boldsymbol{n}_1{}^r \boldsymbol{n}_2 \boldsymbol{n}_1{}^\ell \;::\; \mathtt{and}(x_1,x_2), \quad \mathit{and} : \boldsymbol{n}_2{}^r \boldsymbol{n}_2 \boldsymbol{n}_2{}^\ell \;::\; \mathtt{and}(x_1,x_2)\,.$$

Note that $\mathtt{and}(a,a) = \{a\}$ and $\mathtt{and}(A,A) = A$. Therefore $\mathtt{and}(A,A) \neq \{A\}$. In fact, closure under the two-sorted union does not imply that a set exists which has the set A as element.

The operation of two-sorted union is the companion to the sum of vectors and spaces. Hence the property of commuting with two-sorted unions is the companion to the property of commuting the the sum in vector spaces. The established one-sorted set-theoretical notation is cumbersome when it comes to two-sorted logic. Here is an example what 'commuting with two-sorted unions' looks like in one-sorted notation.

Lemma 3. *A function f that maps elements and sets to elements or sets commutes with the two-sorted union* `and` *if and only if it satisfies*

$$\begin{aligned}
&f(\{a\}) = \{f(a)\} \;\textit{if } f(a) \textit{ is an element}\\
&f(\{a\}) = f(a) \;\textit{if } f(a) \textit{ is a set}\\
&f(A \cup B) = f(A) \cup f(B) \;\textit{if } f(A) \textit{ and } f(B) \textit{ are sets}\\
&f(A \cup B) = \{f(A)\} \cup f(B) \;\textit{if } f(A) \textit{ is an element and } f(B) \textit{ a set}\\
&f(A \cup B) = f(A) \cup \{f(B)\} \;\textit{if } f(A) \textit{ is a set and } f(B) \textit{ an element}\\
&f(A \cup B) = \{f(A)\} \cup \{f(B)\} \;\textit{if } f(A) \textit{ and } f(B) \textit{ are elements}
\end{aligned}$$

Proof. $f(\mathtt{and}(a,a)) = f(\{a\})$. On the other hand, $\mathtt{and}(f(a),f(a))$ is equal to $\{f(a)\}$ if $f(a)$ is an element and to $f(a)$ if it is a set. Hence, if f commutes with `and` the first two equalities hold. Similarly, $f(\mathtt{and}(A,B)) = f(A \cup B)$. Again, it suffices to express $\mathtt{and}(f(A),f(B))$ in terms of union and unordered pair formation to see that the last four equalities hold if f commutes with `and`. □

The problem is that $f(a)$ can be an element or a set. Without the notational distinction between $f(a)$ and $\{f(a)\}$ the six equalities above are equivalent to the two equalities $f(\{a\}) = f(a)$ and $f(A \cup B) = f(A) \cup f(B)$. We resolutely write $f(\{a\}) = f(a)$ from now on, even if $f(a)$ is an element. Hence the two-sorted functions of Definition (10) are exactly the functions which map the empty set to itself and commute with the two-sorted union.

The relative pronoun *who* is an example of a two-sorted function in natural language. The noun phrase *the boys who entered the room* may be refer to nobody, a single boy or several boys. Moreover, *the boys and girls who entered*

the room is the union of *the boys who entered the room* and *the girls who entered the room*. We take the point of view that all meaning expressions in a pregroup lexicon refer either to elements or to sets or to two-sorted functions and illustrate this with some in particular, the quantifiers and determiners in the next section.

Before we can prove Theorem 4, we need an operation which transforms two-sorted function $g : A \to B$ into a set $\ulcorner g \urcorner$, called the *graph* of g

$$\ulcorner g \urcorner = \{\langle a, b\rangle \in A \times B : \ a \in A, \ b \in B \text{ and either } b = g(a) \text{ or } b \in g(a)\}\,.$$

With the understanding that 'union' means 'two-sorted union' and 'product' means 'two-sorted product' in Theorem 1, its proof is indeed straight forward. As an example, we show that $\mathcal{I}$ maps matrices to graphs:

Assume that $f : V_A \to V_B$ is linear. Let $f(a_i) = \sum_{j=1}^{m} \beta_{ij} b_j$. From $\mathcal{I}(f)(a_i) = \mathcal{I}_B(f(a_i))$ follows that $\langle a_i, b_j\rangle \in \ulcorner \mathcal{I}(f) \urcorner$ if and only if $\beta_{ij} \neq 0$. Hence $\mathcal{I}_{A^* \otimes B}(\ulcorner f \urcorner) = \ulcorner \mathcal{I}(f) \urcorner$. After identification of vectors in $A^* \otimes B$ with linear maps from I to $A^* \otimes B$, this equality becomes

$$\mathcal{I}(\ulcorner f \urcorner) = \ulcorner \mathcal{I}(f) \urcorner\,.$$

Two-sorted functions are closed under composition. They include the constant functions, the unary functions obtained from the two-sorted union by making one argument equal to the empty set, and the projections of a product set onto the factors.

Remark: If A is finite every function that maps elements of A to elements of B extends uniquely into a two-sorted function from A to B. In particular, every relation of first order logic can be viewed as a two-sorted function that sends elements to truth-values, more precisely

Definition 2. *A* predicate *is a two-sorted function* $f : A \to \Omega$ *such that*

$$\forall_x (x \in A \Longrightarrow f(x) \in \Omega)\,.$$

Note that the value of a predicate on a subset with more than two elements may be a set, namely $\{\top, \bot\}$. Moreover, predicates are closed under composition.

We conclude this section with a fundamental property of predicates. The interpretation of quantifiers by two-sorted functions is based on this property.

Lemma 4. [**Fundamental Lemma**] *Assume that f maps elements to elements. Then the following holds*
Fundamental Property
For all finite $X \subseteq A$ and all $b \in f(A)$

$$f(X) = b \Leftrightarrow (X \neq \emptyset \text{ and } f(a) = b \text{ for all } a \in X)\,. \tag{13}$$

Proof. If $f(X) = b$ for some $b \in f(A)$ then X is not the empty set, because two-sorted functions map the empty set to the empty set. Assume that $X = \{a_1, \ldots, a_n\}$. Then $f(X) = f(\{a_1\} \cup \cdots \cup \{a_n\}) = \{f(a_1)\} \cup \cdots \cup \{f(a_n)\}$, because f commutes with finite unions. By assumption, no $f(a_i)$ is the empty set. Hence the equivalence follows. □

7 The logic of sentences

The Fundamental Property provides the explanation how quantifiers can be captured by two-sorted functions. We explain this for universal quantifiers. A detailed development of the behaviour of quantifiers and determiners is beyond the scope of this paper. We use functional notation when discussing the meaning of sentences, vector notation when computing it.

Definition 3. *A sentence $w_1 \dots w_n$ is* true in the interior logic *if its meaning expression $m(w_1 \dots w_n)$ satisfies*

$$m(w_1 \dots w_n) = \top \textit{ holds in 2-FOL.}$$

Note that $\texttt{leave}(\texttt{boy}) \neq \top$ is not equivalent to $\texttt{leave}(\texttt{boy}) = \bot$, because $\{\top, \bot\}$ is another possible value. For all elements $a \in \texttt{boy}$, however, $\texttt{leave}(a) \neq \top$ is equivalent to $\texttt{leave}(a) = \bot$. Consider the sentences

S_1	*All boys left* *The boys left*	S_2	*The boys did not leave* *No boys left*.

The first and second sentence have the same meaning and so do the third and fourth. Obviously, the meaning of the latter two sentences is not the logical negation of the meaning of the first two. Accordingly, the first order formulas rendering S_1 and S_2 are not equivalent.

$$\begin{array}{ll} S_1 & \forall a(a \in \textit{boy} \Longrightarrow a \in \textit{leave}) \\ S_2 & \forall a(a \in \textit{boy} \Longrightarrow a \notin \textit{leave})\,. \end{array}$$

In our formalisation, relations are represented by functions with values in Ω

$$\begin{array}{ll} S_1 & \forall a(a \in \texttt{boy} \Longrightarrow \texttt{leave}(a)) = \top) \\ S_2 & \forall a(a \in \texttt{boy} \Longrightarrow \texttt{leave}(a)) = \bot)\,. \end{array}$$

Recall the postulate $\texttt{all} = id_E$. Similarly, we require

$$\texttt{the} = id_E\,. \tag{14}$$

Accordingly, the meaning expression $m(\textit{All boys left})$ computes to

$$\texttt{leave}(\texttt{all}(\texttt{boy})) = \texttt{leave}(\texttt{boy}) = \texttt{leave}(\texttt{the}(\texttt{boy}))\,.$$

Under the assumption that the set boy is not empty, the equivalence below is an instance of the Fundamental Property and therefore holds in 2-FOL

$$\texttt{leave}(\texttt{boy}) = \top \Longleftrightarrow \forall a(a \in \texttt{boy} \Longrightarrow \texttt{leave}(a)) = \top)\,.$$

The assumption $\texttt{boy} \neq \emptyset$ cannot be proved. Telling someone that *All/The boys left* in a situation where there are no boys would provoke the question *Which boys?*. We ignore the problems arising when there is no obvious referent for the count-noun.

The sentences *The boys do not leave* and *No boys left* also have the same intuitive meaning. They are rendered by the first order formula

$$\forall a(a \in \mathtt{boy} \Longrightarrow \mathtt{leave}(a) = \bot)\,.$$

The meaning expression of *The boys did not leave* is given by (6) in Section 2 as

$$\mathtt{not}_\Omega(\mathtt{leave}(\mathtt{the}(\mathtt{boy})))\,.$$

From (14) follows

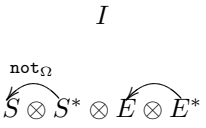
$$m(\textit{The boys did not leave}) = \mathtt{not}_\Omega(\mathtt{leave}(\mathtt{boy}))\,.$$

The lexical entry for the word *no* also involves the universal quantifier *id* and the sentence negation $\mathtt{not}_\Omega$

$$\begin{aligned} &\textit{no}\colon \boldsymbol{s}\boldsymbol{s}^\ell\boldsymbol{n}_1\boldsymbol{c}_1{}^\ell :: I \xrightarrow{\overline{\mathtt{no}}} S \otimes S^* \otimes E \otimes E^* \quad \mathtt{not}_\Omega(z_1)\ id(z_2)\\ &\textit{no}\colon \boldsymbol{s}\boldsymbol{s}^\ell\boldsymbol{n}_2\boldsymbol{c}_2{}^\ell :: I \xrightarrow{\overline{\mathtt{no}}} E^* \otimes S \otimes S^* \otimes E \quad \mathtt{not}_\Omega(z_1)\ id(z_2)\,. \end{aligned}$$

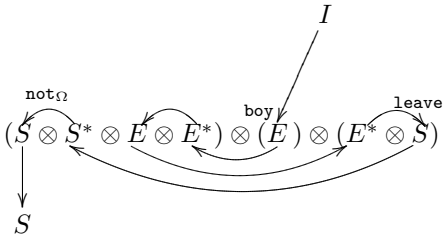
The graph of the vector $\overline{\mathtt{no}}$ is

$$\overline{\mathtt{no}} : I \to S \otimes S^* \otimes E \otimes E^* =$$



The meaning expression of the sentence *No boys left* is computed by the following graph, where the underlinks are given by the reduction

$$m(\textit{No boys left}) = r_2' \circ (\overline{\mathtt{no}} \otimes \overline{\mathtt{boy}} \otimes \overline{\mathtt{leave}}) =$$



$$= \mathtt{not}_\Omega(\mathtt{leave}(\mathtt{boy}))\,.$$

We prove the equivalence

$$\mathtt{not}_\Omega(\mathtt{leave}(\mathtt{boy})) = \top \Leftrightarrow \forall a(a \in \mathtt{boy} \Longrightarrow \mathtt{leave}(a) = \bot)\,.$$

Consider the lefthand equality. It is equivalent to $\mathtt{leave}(\mathtt{boy})) = \bot$, because $\top$ is the only element or subset of Ω mapped by $\mathtt{not}_\Omega$ to $\bot$ by (7). To conclude, it suffices to remark that the Fundamental Property (13) implies

$$\mathtt{leave}(\mathtt{boy})) = \bot \Longleftrightarrow \forall a(a \in \mathtt{boy} \Longrightarrow \mathtt{leave}(a) = \bot)\,.$$

Examples, continued

Return to the Examples (8) in Section 2. The statement *All boys left* holds in the interior logic of the model M_1, whereas the statement *The boys did not leave* holds in M_2. Indeed,

$$\texttt{leave}_1(\texttt{boy}) = \texttt{leave}_1(\sum_{1\leq i\leq m} a_i) = \top\,.$$

Similarly,

$$\texttt{leave}_2(\texttt{boy}) = \bot$$

and therefore

$$\texttt{not}_\Omega(\texttt{leave}_2(\texttt{boy}) = \top\,.$$

In the third example M_3, only the boys a_1 and a_2 left. In this case,

$$\texttt{leave}_3(\texttt{boy}) = \{\top, \bot\} \text{ and } \texttt{not}_\Omega(\texttt{leave}_3(\texttt{boy})) = \{\top, \bot\}\ .$$

Indeed, the two-sorted functions $\texttt{leave}$ and $\texttt{not}_\Omega$ commute with unions

$$\begin{aligned}\texttt{leave}_3(\texttt{boy}) &= \texttt{leave}_3((\{a_1, a_2\} \cup \{a_3, \ldots, a_m\})\\ &= \texttt{leave}_3(\{a_1, a_2\}) \cup \texttt{leave}_3(\{a_3, \ldots, a_m\})\\ &= \{\top, \bot\}\ .\end{aligned}$$

Therefore

$$\texttt{not}_\Omega(\texttt{leave}_3(\texttt{boy})) = \texttt{not}_\Omega(\{\top, \bot\}) = \{\texttt{not}_\Omega(\top)\}, \{\texttt{not}_\Omega(\bot)\} = \{\bot, \top\}\ .$$

The equality partitions $\texttt{boy}$ in two non-empty sets, namely the part for which $\texttt{leave}_3$ is true and another part for which it is false. In fact, these parts are constructed by the two-sorted function (linear map) that interprets the relative pronoun *who*.

Comprehension

The logical meaning of the relative pronoun *who* is captured by the axiom

$$\texttt{who}(a, b) = \begin{cases} a & \text{if } b = \top \\ \emptyset & \text{if } b = \bot \end{cases}$$

which defines the two-sorted map $\texttt{who} : E \times S \to E$ entirely.

Lemma 5. *For every predicate $f : E \to S$, the following equivalence holds*

$$\forall x\ (x \in \texttt{who}(A, f(A)) \iff (x \in A \text{ and } f(x) = \top))\,. \tag{15}$$

Proof. The functions $\texttt{who}$ and f commute with two-sorted unions and so does the composed function

$$x \mapsto \texttt{who}(x, f(x)) = (\texttt{who} \circ (id_E \times f) \circ d_E)(x).$$

By definition,

$$\begin{array}{ll} \texttt{who}(a, f(a)) & = a \text{ if } f(a) = \top \\ \texttt{who}(a, f(a)) & = \emptyset \text{ else} \end{array} .$$

Let $A = \{a_1, \ldots, a_n\} = \bigcup_{i=1}^{n} \{a_i\}$. Then

$$\begin{array}{ll} \texttt{who}(A, f(A)) & = (\texttt{who} \circ (id_E \times f) \circ d_E)(\bigcup_{i=1}^{n} a_i) \\ & = (\texttt{who} \circ (id_E \times f))(\bigcup_{i=1}^{n} \langle a_i, a_i \rangle) \\ & = \texttt{who}(\bigcup_{i=1}^{n} \langle a_i, f(a_i) \rangle) \\ & = \bigcup_{i=1}^{n} \texttt{who}(a_i, f(a_i)) . \end{array}$$

It follows that $x \in \bigcup_{i=1}^{n} \texttt{who}(a_i, f(a_i))$ if and only if $x = a_i$ for some $a_i \in A$ such that $f(a_i) = \top$. □

For example in M_3

$$\begin{array}{l} \texttt{who}(\texttt{boy}, \texttt{leave}_3(\texttt{boy})) = \{a_1, a_2\} \\ \texttt{who}(\texttt{boy}, \texttt{not}_\Omega(\texttt{leave}_3(\texttt{boy}))) = \{a_3, \ldots, a_m\} . \end{array}$$

The property expressed in (15) can be reformulated thus

Theorem 2. *The comprehension schema holds for predicates*

Proof. Indeed, $\texttt{who}(A, f(A))$ is a well defined set. Hence it suffices to put

$$\{x \in A : f(x) = \top\} = \texttt{who}(A, f(A)) .$$

□

8 Conclusion

The semantics proposed here is compositional. Its vectorial version satisfies the postulates of [Kracht, 2007], namely meanings do not involve variables and are invariant under change of order or multiplicity. Moreover, the meaning of a statement is true in the interior logic if and only if the corresponding closed first-order formula holds in two-sorted first order logic.

Compact closed categories unite the distributional semantic vector models based on concepts and the functional semantics based on individuals in a common frame. Work on the relation between the vectors given by pregroup semantics and the vectors given by a probability distribution shows promising results in [Clark et al., 2010].

The algorithm that computes the meaning of a string of words from the meanings of the words is proportional to the length of the string. The reduction intervening in the computation is provided by the cubic parsing algorithm for pregroup grammars. Hence, pregroup lexicons that list not only the syntactic types of words, but also the meaning expressions provide an efficient tool for transforming text to formal semantical expressions commonly used in knowledge representation. Admittedly, the fragments of natural language covered so far are insufficient for applications. However, meta-rules in the style of [Lambek, 2008] make the creation of comprehensive lexicons quite efficient.

9 Acknowledgement

The authors would like to thank two anonymous referees for constructive comments.

References

Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425, 2004.

Johan van Benthem and Kees Doets. *Handbook of Philosophical Logic*, chapter Higher-Order Logic, pages 275–329. Reidel Publishing Company, Dordrecht, 1983.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. A compositional distributional model of meaning. In W. Lawless P. Bruza and J. van Rijsbergen, editors, *Proceedings of Conference on Quantum Interactions*. University of Oxford, College Publications, 2008.

Stephen Clark, B. Coecke, E. Grefenstette, S. Pulman, and M. Sadrzadeh. Concrete compositional sentence spaces. In *Compositionality and Distributional Semantic Models, ESSLLI 2010*, 2010.

Daoud Clarke. *Context-theoretic Semantics for Natural Language, an algebraic framework*. PhD thesis, University of Sussex U.K., 2007.

Marcus Kracht. The emergence of syntactical structure. *Linguist. Philos.*, 30: 47–95, 2007.

Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

Joachim Lambek. *Substructural Logics*, chapter From categorial grammar to bilinear logic, pages 207–237. Oxford University Press, 1993.

Joachim Lambek. Type grammar revisited. In Alain et al. Lecomte, editor, *Logical Aspects of Computational Linguistics*, volume 1582 of *LNAI*, pages 1–27, Heidelberg, 1999. Springer.

Joachim Lambek. *From word to sentence*. Polimetrica, Milano, Italia, 2008.

Anne Preller. Category theoretical semantics for pregroup grammars. In Philippe Blache and Edward Stabler, editors, *Logical Aspects of Computational Linguistics*, volume 3492 of *Lecture Notes in Artificial Intelligence*, pages 254–270, 2005.

Anne Preller. Toward discourse representation via pregroup grammars. *JoLLI*, 16:173–194, 2007. doi: http://dx.doi.org/10.1007/s10849-006-9033-y.

Anne Preller and Violaine Prince. Quantum logic unites compositional functional semantics and distributional semantic models. In *Compositionality and Distributional Semantic Models, ESSLLI 2010*, 2010. URL `http://hal-lirmm.ccsd.cnrs.fr/lirmm-00495559`.

Anne Preller and Mehrnoosh Sadrzadeh. Bell states and negated sentences in the distributed model of meaning. *Electronic Notes in Theoretical Computer Science*, 270:141–153, 2011. doi: http://dx.doi.org/10.1016/j.entcs.2011.01.028.

C.J. van Rijsbergen. *The Geometry of Information Retrieval*. Cambridge University Press, 2004.

Mehrnoosh Sadrzadeh. High-level quantum structures in linguistics and multiagent systems. In *AAAI spring symposium on quantum interactions*, 2007.