1 Introduction

This chapter introduces context algebras and demonstrates their application to combining logical and vector-based representations of meaning. Other chapters in this volume consider approaches that attempt to reproduce aspects of logical semantics within new frameworks. The approach we present here is different: We show how logical semantics can be embedded within a vector space framework, and use this to combine distributional semantics, in which the meanings of words are represented as vectors, with logical semantics, in which the meaning of a sentence is represented as a logical form.

The ideas discussed here are present (at least implicitly) in earlier work, however we have introduced some notions which allow the mathematics to be tidied considerably:

- When context algebras were introduced [3] they were applied only to functions from a free monoid $A^*$ to $\mathbb{R}$. In fact, this construction generalises to functions from $A^*$ to an arbitrary vector space $V$. The proof of the general case is identical to the specific one, and is reproduced here unchanged.

- This more general construction gives us an elegant way of embedding logical semantics within an algebraic framework. The embedding presented here follows similar lines to the thinking of [3], but uses the new, more general, context algebras.

- The method of combining logical semantics with vector-based lexical semantics is new, but follows similar lines to an approach suggested in [4].

1.1 Motivation

Like other work in this book, we are concerned with the question of how to compose vector-based representations of meaning so that phrases and sentences are also represented as vectors. We wish to preserve the wonderful flexibility and fine-grained distinctions of meaning that vector spaces allow, and which
have been so successful in lexical semantics, to build a complete framework for natural language semantics encompassing words, phrases, sentences and beyond. Unlike other work, in the approach presented here, we do not attempt to reconstruct logical semantics from scratch, instead embedding logical representations within a vector space. This has some benefits:

- Doing natural language semantics well is difficult, and a lot of work has gone into getting logical semantics for natural language right. It includes worrying about things like anaphora resolution, generalised quantifiers and negation, and reproducing this work from scratch in a vector-based framework is a mammoth task. Our approach allows us to reuse existing work while incorporating vector-based lexical semantics.

- There is the potential to reuse existing tools for natural language semantics, although computation in general is a problem with our approach.

The downside to our approach is that we don’t yet have an efficient way of computing with it, although we have ideas for how this may be achieved. Another potential criticism of this approach is that the flexibility in how vector representations are combined with logic may be hindered by requiring the wholesale adoption of existing formalisms, rather than the more tailored approaches of other work.

2 Theory of Meaning

We first recall some basic definitions:

**Definition 1** (Algebra over a field). An algebra over a field is a vector space $A$ over a field $K$ together with a binary operation $(a, b) \mapsto ab$ on $A$ that is bilinear,

\begin{align*}
(1) \quad &a(ab + bc) = aab + bac \\
(2) \quad &(aa + \beta b)c = aac + \beta bc
\end{align*}

for all $a, b, c \in A$ and all $\alpha, \beta \in K$. If we additionally have the property $(ab)c = a(bc)$ then $A$ is called associative. An algebra is called unital if it has a distinguished unity element $1$ satisfying $1x = x1 = x$ for all $x \in A$. We are generally only interested in real associative algebras, where $K$ is the field of real numbers, $\mathbb{R}$.

Examples of associative algebras are given by square matrices of order $n$ under normal matrix multiplication and entry-wise vector operations. The field of the algebra is the field of the elements of the matrices; so real valued matrices form a real associative algebra.

2.1 Meaning as Context

The distributional hypothesis of Harris [5] states that words will have similar meanings if and only if they occur in similar contexts. We formalise this idea, and examine the resultant mathematical properties.
Let \( A \) be some set, which we imagine to be the set of words of a natural language. If \( V \) is a vector space, we define a **general language for** \( V \) (or simply a **language** when there is no ambiguity) as a function from the free monoid \( A^* \) to \( V \). For each string \( x \in A^* \), we have associated with it a vector in \( V \) that may have several interpretations:

- \( V \) may simply be the real numbers \( \mathbb{R} \), and the language may describe a probability distribution over strings in \( A^* \), in which case we can view the language as a generative model of a natural language, describing the probability of observing each possible string as a sentence or a document.
- \( V \) may be a vector space describing the meaning of strings, for example a representation of model-theoretic semantics. In this case, the language attaches a meaning to each possible string in \( A^* \).

Given a general language \( L \) we define the context vector \( \hat{x} \) of a string \( x \) as a function from \( A^* \times A^* \) to \( V \):

\[
\hat{x}(y,z) = L(yxz)
\]

Thus, as in the study of formal languages, we consider the context of a string to be the pair of strings surrounding it. We think of \( \hat{x} \) as an element of the vector space \( V^{A^* \times A^*} \), the space of functions from \( A^* \times A^* \) to \( V \). This is a vector space with operations defined point-wise, i.e. if \( f, g \in V^{A^* \times A^*} \) and \( \alpha \in K \) where \( K \) is the field of \( V \) then \( (\alpha f)(x,y) = \alpha f(x,y) \) and \( (f + g)(x,y) = f(x,y) + g(x,y) \) for all \( x, y \in A^* \).

**Definition 2** (Generated Subspace \( A \)). The subspace \( A \) of \( V^{A^* \times A^*} \) is the set defined by

\[
A = \{ a : a = \sum_{x \in A^*} \alpha_x \hat{x} \text{ for some } \alpha_x \in \mathbb{R} \}
\]

In other words, it is the space of all vectors formed from linear combinations of context vectors.

Given this definition, we can define multiplication on \( A \), by assuming linearity, and making the multiplication compatible with the underlying multiplication of \( A^* \). That is, we want to define a product \( \cdot \) on \( A \) such that \( \hat{x} \cdot \hat{y} = \hat{xy} \) for all \( x, y \in A^* \). However, in general there is more than one basis for \( A \) formed from elements \( \hat{x} \), for \( x \in A^* \). We need to confirm that multiplication will be the same, regardless of which basis we choose.

**Proposition 1** (Context Algebra). Multiplication on \( A \) is the same irrespective of the choice of basis \( B \).

**Proof.** We say \( B \subseteq A^* \) defines a basis \( B \) for \( A \) when \( B \) is a basis such that \( B = \{ \hat{x} : x \in B \} \). Assume there are two sets \( B_1, B_2 \subseteq A^* \) that define corresponding bases \( B_1 \) and \( B_2 \) for \( A \). We will show that multiplication in basis \( B_1 \) is the same as in the basis \( B_2 \).
We represent two basis elements \( \hat{u}_1 \) and \( \hat{u}_2 \) of \( B_1 \) in terms of basis elements of \( B_2 \):

\[
\hat{u}_1 = \sum_i \alpha_i \hat{v}_i \quad \text{and} \quad \hat{u}_2 = \sum_j \beta_j \hat{v}_j
\]

for some \( u_i \in B_1, v_j \in B_2 \) and \( \alpha_i, \beta_j \in \mathbb{R} \). First consider multiplication in the basis \( B_1 \). Note that \( \hat{u}_1 = \sum_i \alpha_i \hat{v}_i \) means that \( L(xu_1y) = \sum_i \alpha_i L(xv_iy) \) for all \( x, y \in A^* \). This includes the special case where \( y = u_2y' \) so

\[
L(xu_1u_2y') = \sum_i \alpha_i L(xv_iu_2y')
\]

for all \( x, y' \in A^* \). Similarly, we have \( L(xu_2y) = \sum_j \beta_j L(xv_jy) \) for all \( x, y \in A^* \) which includes the special case \( x = x'v_i \), so \( L(x'v_iu_2y) = \sum_j \beta_j L(x'v_iv_jy) \) for all \( x', y \in A^* \). Inserting this into the above expression yields

\[
L(xu_1u_2y) = \sum_{i,j} \alpha_i \beta_j (\hat{v}_i \cdot \hat{v}_j)
\]

for all \( x, y \in A^* \) which we can rewrite as

\[
\hat{u}_1 \cdot \hat{u}_2 = \hat{u}_1u_2 = \sum_{i,j} \alpha_i \beta_j (\hat{v}_i \cdot \hat{v}_j) = \sum_{i,j} \alpha_i \beta_j \hat{v}_i \hat{v}_j
\]

Conversely, the product of \( u_1 \) and \( u_2 \) using the basis \( B_2 \) is

\[
\hat{u}_1 \cdot \hat{u}_2 = \sum_i \alpha_i \hat{v}_i \cdot \sum_j \beta_j \hat{v}_j = \sum_{i,j} \alpha_i \beta_j (\hat{v}_i \cdot \hat{v}_j)
\]

thus showing that multiplication is defined independently of what we choose as the basis.

Multiplication as defined above makes \( \mathcal{A} \) an algebra, moreover it is easy to see that it is associative since the multiplication on \( A^* \) is associative. It has a unity, which is given by \( \hat{\epsilon} \), where \( \epsilon \) is the empty string.

### 2.2 Entailment

Our notion of entailment is founded on the idea of **distributional generality** [8]. This is the idea that the distribution over contexts has implications not only for similarity of meaning, but can also describe how general a meaning is. A term \( t_1 \) is considered distributionally more general than another term \( t_2 \) if \( t_1 \) occurs in a wider range of contexts than \( t_2 \). It is proposed that distributional generality may be connected to semantic generality. For example, we may expect the term **animal** to occur in a wider range of contexts than the term **cat** since the first is semantically more general.

We translate this to a mathematical definition by making use of an implicit partial ordering on the vector space:
Definition 3 (Partially ordered vector space). A partially ordered vector space \( V \) is a real vector space together with a partial ordering \( \leq \) such that:

- if \( x \leq y \) then \( x + z \leq y + z \)
- if \( x \leq y \) then \( \alpha x \leq \alpha y \)

for all \( x, y, z \in V \), and for all \( \alpha \geq 0 \). Such a partial ordering is called a vector space order on \( V \). An element \( u \) of \( V \) satisfying \( u \geq 0 \) is called a positive element; the set of all positive elements of \( V \) is denoted \( V^+ \). If \( \leq \) defines a lattice on \( V \) then the space is called a vector lattice or Riesz space.

If \( V \) is a vector lattice, then the vector space of contexts, \( V^{A^* \times A^*} \), is a vector lattice, where the lattice operations are defined component-wise: \( (u \land v)(x, y) = u(x, y) \land v(x, y) \) and \( (u \lor v)(x, y) = u(x, y) \lor v(x, y) \). For example, \( \mathbb{R} \) is a vector lattice with meet as the min operation and join as max, so \( \mathbb{R}^{A^* \times A^*} \) is also a vector lattice. In this case, where the value attached to a context is an indication of its frequency of occurrence, \( \hat{x} \leq \hat{y} \) means that \( y \) occurs at least as frequently as \( x \) in every context.

Note that, unlike the vector operations, the lattice operations are dependent on the basis: a different basis gives different operations. This makes sense in the linguistic setting, since there is nearly always a distinguished basis, originating in the contexts from which the vector space is formed.

3 From logical forms to algebra

Model-theoretic approaches generally deal with a subset of all possible strings, the language under consideration, translating sequences in the language to a logical form, expressed in another, logical language. Relationships between logical forms are expressed by an entailment relation on this logical language.

This section is about the algebraic representation of logical languages. Representing logical languages in terms of an algebra will allow us to incorporate statistical information about language into the representations. For example, if we have multiple parses for a sentence, each with a certain probability, we will be able to represent the meaning of the sentence as a probabilistic sum of the representations of its individual parses.

By a logical language we mean a language \( \Lambda \subseteq A^* \) for some alphabet \( A' \), together with a relation \( \vdash \) on \( \Lambda \) that is reflexive and transitive; this relation is interpreted as entailment on the logical language. We will show how each element \( u \in \Lambda \) can be associated with a projection on a vector space; it is these projections that define the algebra. Later we will show how this can be related to strings in the natural language \( \lambda \) that we are interested in.

For a subset \( T \) of a set \( S \), we define the projection \( P_T \) on \( L^\infty(S) \) (the set of all bounded real-valued functions on \( S \)) by

\[
P_T e_s = \begin{cases} 
  e_s & \text{if } s \in T \\
  0 & \text{otherwise}
\end{cases}
\]
Where $e_s$ is the basis element of $L^\infty(S)$ corresponding to the element $s \in S$. Given $u \in \Lambda$, define $\downarrow u = \{ v : v \vdash u \}$.

As a shorthand we write $P_u$ for the projection $P_{\downarrow u}$ on the space $L^\infty(\Lambda)$. The projection $P_u$ can be thought of as projecting onto the space of logical statements that entail $u$. This is made formal in the following proposition:

**Proposition 2.** $P_u \leq P_v$ if and only if $u \vdash v$.

**Proof.** Recall that the partial ordering on projections is defined by $P_u \leq P_v$ if and only if $P_u P_v = P_v P_u = P_u$ \[2\]. Clearly

$$P_u P_v e_w = \begin{cases} 
  e_w & \text{if } w \vdash u \text{ and } w \vdash v \\
  0 & \text{otherwise}
\end{cases}$$

so if $u \vdash v$ then since $\vdash$ is transitive, if $w \vdash u$ then $w \vdash v$ so we must have $P_u P_v = P_v P_u = P_u$.

Conversely, if $P_u P_v = P_u$ then it must be the case that $w \vdash u$ implies $w \vdash v$ for all $w \in \Lambda$, including $w = u$. Since $\vdash$ is reflexive, we have $u \vdash u$, so $u \vdash v$ which completes the proof. \[Q.E.D.\]

To help us understand this representation better, we will show that it is closely connected to the ideal completion of partial orders. Define a relation $\equiv$ on $\Lambda$ by $u \equiv v$ if and only if $u \vdash v$ and $v \vdash u$. Clearly $\equiv$ is an equivalence relation; we denote the equivalence class of $u$ by $[u]$. Equivalence classes are then partially ordered by $[u] \leq [v]$ if and only if $u \vdash v$. Then note that $\bigcup \downarrow ([u]) = \downarrow u$, thus $P_u$ projects onto the space generated by the basis vectors corresponding to the elements $\bigcup \downarrow ([u])$, the ideal completion representation of the partially ordered equivalence classes.

What we have shown here is that logical forms can be viewed as projections on a vector space. Since projections are operators on a vector space, they are themselves vectors; viewing logical representations in this way allows us to treat them as vectors, and we have all the flexibility that comes with vector spaces: we can add them, subtract them and multiply them by scalars; since the vector space is also a vector lattice, we also have the lattice operations of meet and join. As we will see in the next section, in some special cases such as that of the propositional calculus, the lattice meet and join coincide with logical conjunction and disjunction.

### 3.1 Example: Propositional Calculus

In this section we apply the ideas of the previous section to an important special case: that of the propositional calculus. We choose as our logical language $\Lambda$ the language of a propositional calculus with the usual connectives $\lor$, $\land$ and $\neg$, the logical constants $\top$ and $\bot$ representing “true” and “false” respectively, with
$u \vdash v$ meaning “infer $v$ from $u$”, behaving in the usual way. Then:

\[
\begin{align*}
P_{u \land v} &= P_u P_v \\
P_{\neg u} &= 1 - P_u + P_\bot \\
P_{u \lor v} &= P_u + P_v - P_u P_v \\
P_\top &= 1
\end{align*}
\]

To see this, note that the equivalence classes of $\vdash$ form a Boolean algebra under the partial ordering induced by $\vdash$, with

\[
\begin{align*}
[u \land v] &= [u] \land [v] \\
[u \lor v] &= [u] \lor [v] \\
[\neg u] &= \neg [u].
\end{align*}
\]

Note that while the symbols $\land$, $\lor$ and $\neg$ refer to logical operations on the left hand side, on the right hand side they are the operations of the Boolean algebra of equivalence classes; they are completely determined by the partial ordering associated with $\vdash$.

Since the partial ordering carries over to the ideal completion we must have

\[
\begin{align*}
\downarrow [u \land v] &= \downarrow [u] \cap \downarrow [v] \\
\downarrow [u \lor v] &= \downarrow [u] \cup \downarrow [v]
\end{align*}
\]

Since $u \vdash \top$ for all $u \in \Lambda$, it must be the case that $\downarrow [\top]$ contains all sets in the ideal completion. However the Boolean algebra of subsets in the ideal completion is larger than the Boolean algebra of equivalence classes; the latter is embedded as a Boolean sub-algebra of the former. Specifically, the least element in the completion is the empty set, whereas the least element in the equivalence class is represented as $\downarrow [\bot]$. Thus negation carries over with respect to this least element:

\[
[\neg u] = ([\top] - [u]) \cup [\bot].
\]

We are now in a position to prove the original statements:

- Since $\downarrow [\top]$ contains all sets in the completion, $\bigcup \downarrow [\top] = \downarrow \vdash (\top) = \Lambda$, and $P_\top$ must project onto the whole space, that is $P_\top = 1$.

- Using the above expression for $\downarrow [u \land v]$, taking unions of the disjoint sets in the equivalence classes we have $\downarrow \vdash (u \land v) = \downarrow \vdash (u) \cap \downarrow \vdash (v)$. Making use of the equation in the proof to Proposition 2, we have $P_{u \land v} = P_u P_v$.

- In the above expression for $\downarrow [\neg u]$, note that $\downarrow [\top] \subseteq \downarrow [u] \subseteq \downarrow [\bot]$. This allows us to write, after taking unions and converting to projections, $P_{\neg u} = 1 - P_u + P_\bot$, since $P_\top = 1$.

\footnote{In the context of model theory, the Boolean algebra of equivalence classes of sentences of some theory $T$ is called the Lindenbaum-Tarski algebra of $T$.}
Finally, we know that $u \lor v \equiv \neg(\neg u \land \neg v)$, and since equivalent elements in $\Lambda$ have the same projections we have

$$
P_{u \lor v} = 1 - P_{\neg u \land \neg v} + P_\perp
= 1 - P_{\neg u}P_{\neg v} + P_\perp
= 1 - (1 - P_u + P_\perp)(1 - P_v + P_\perp) + P_\perp
= P_u + P_v - P_uP_v - 2P_\perp + P_\perpP_u + P_\perpP_v
= P_u + P_v - P_uP_v
$$

It is also worth noting that in terms of the vector lattice operations $\lor$ and $\land$ on the space of operators on $L^\infty(\Lambda)$, we have $P_{u \lor v} = P_u \lor P_v$ and $P_u \land v = P_u \land P_v$.

### 3.2 From logic to context algebras

In the simplest case, we may be able to assign to each natural language sentence a single sentence in the logical language (its interpretation). Let $\Gamma \subseteq \mathcal{A}^*$ be a formal language consisting of natural language sentences $x$ with a corresponding interpretation in the logical language $\Lambda$, which we denote $\rho(x)$. The function $\rho$ maps natural language sentences to their interpretations, and may incorporate tasks such as word-sense disambiguation, anaphora resolution and semantic disambiguation.

We can now define a general language to represent this situation. We take as our vector space $V$ the space generated by projections $\{P_u : u \in \Lambda\}$ on $L^\infty(\Lambda)$. For $x \in \mathcal{A}^*$ we define

$$L(x) = \begin{cases} P_{\rho(x)} & \text{if } x \in \Gamma \\ 0 & \text{otherwise} \end{cases}$$

Given the discussions in the preceding sections, it is clear that for $x, y \in \Gamma$, $L(x) \preceq L(y)$ if and only if $\rho(x) \vdash \rho(y)$, so the partial ordering of the vector space encodes the entailment relation of the logical language.

The context algebra constructed from $L$ gives meaning to any substring of elements of $\Gamma$, so any natural language expression which is a substring of a sentence with a logical interpretation has a corresponding non-zero element in the algebra. If $\Gamma$ has the property that no element of $\Gamma$ is a substring of any other element of $\Gamma$ (for example, $\Gamma$ consists of natural language sentences starting and ending with a unique symbol), then we will also have $\hat{x} \leq \hat{y}$ if and only if $\rho(x) \vdash \rho(y)$, for all $x, y \in \Gamma$. In this case, the only context which maps to a non-zero vector is the pair of empty strings, $(\epsilon, \epsilon)$.

### 3.3 Incorporating Word Vectors

The construction of the preceding section is not very useful on its own, as it merely encodes the logical reasoning within a vector space framework. However, we will show how this construction can be used to incorporate vector-based representations of lexical semantics.
In the general case, we assume that associated with each word \( a \in A \) there is a vector \( \psi(a) \) in some finite-dimensional vector space \( L^\infty(S) \) that represents its lexical semantics, perhaps obtained using latent semantic analysis or some other technique. \( S \) is a finite set indexing the lexical semantic vector space which we interpret as containing aspects of meaning. This set may be partitioned into subsets containing aspects for different parts of speech, for example, we may wish that the vector for a verb is always disjoint to the vector for a noun. Similarly, there may be a single aspect for each closed class or function word in the natural language, to allow these words not to have a vector nature.

Instead of mapping directly from strings of the natural language to the logical language, we map from strings of aspects of meaning. Let \( \Delta \subset S^* \) be a formal language consisting of all meaningful strings of aspects, i.e. those with a logical interpretation. As before, we assume a function \( \rho \) from \( \Delta \) to a logical language \( \Lambda \). The corresponding context algebra \( A \) describes composition of aspects of meaning.

We can now describe the representation of the meaning \( \tilde{a} \) of a word \( a \in A \) in terms of elements of \( A \):

\[
\tilde{a} = \sum_{s \in S} \psi(a)_s \hat{s}
\]

thus a term is represented as a weighted sum of the context vectors for its aspects. Composition of \( A \) together with the distributivity of the algebra is then enough to define vectors for any string in \( A^* \). For \( x \in A^* \), we define \( \tilde{x} \) by

\[
\tilde{x} = \tilde{x}_1 \cdot \tilde{x}_2 \cdots \tilde{x}_n
\]

where \( x = x_1x_2 \cdots x_n \) for \( x_n \in A \).

This construction achieves the goal of combining vector space representations of lexical semantics with existing logical formalisms. It has the following properties:

- The entailment relation between the logical expressions associated with sentences is encoded in the partial ordering of the vector space.
- The vector space representation of a sentence includes a sum over all possible logical sentences, where each word has been represented by one of its aspects.

Discussion

This second property is actually inconsistent with the idea that distributional generality determines semantic generality. To see why, consider the sentences

1. *No animal likes cheese*
2. *No cat likes cheese*

Although the first sentence entails the second, the term *animal* is more general than *cat*; the quantifier *no* has reversed the direction of entailment. The distributivity of the algebra means that if the term *animal* occurs in a wider range
of contexts than *cat* (i.e. it is distributionally more general), then this generality must persist through to the sentence level as terms are multiplied.

There are two ways of viewing this inconsistency:

1. The construction is incorrect because the distributional hypothesis does not hold universally at the lexical level. In fact, this idea is justified by our example above, in which the more general term, *animal*, would be expected to occur in a smaller range of contexts than *cat* when preceded by the quantifier *no*. Under this view, the construction needs to be altered so that quantifiers such as *no* can reverse the direction of entailment. In fact, for this to be the case, we would need to dispense with using an algebra altogether, since distributivity is a fundamental property of the algebra.

2. The construction is correct as long as the “aspects” of words are really their senses, so the vector space nature only represents semantic ambiguity and not semantic generality. In reality, aspects may be more subtle than what is normally considered a word sense, and the vector space representation would capture this subtlety. In this view, distributional generality is correlated with semantic generality at the sentence level, but not necessarily the word level. Vector representations describe only semantic ambiguity and not generality, which should be incorporated into the associated logical representation. This would mean that the algorithms used for obtaining vectors for terms would have to have more in common with automatic word sense induction than the more general semantic induction associated with the distributional hypothesis.

An alternative solution for this type of quantifier (another example is *all*) and negation in general, is to use a construction similar to that proposed in [7], which uses Bell states to swap dimensions in a similar manner to qubit operators.

### 3.4 Partial Entailment

In general, it is unlikely that any two strings \( x, y \in A^* \) which humans would judge as entailing would have vectors such that \( \tilde{x} \leq \tilde{y} \) because of the nature of the automatically obtained vectors. Instead, it makes sense to consider a *degree* of entailment between strings. In this case, we assume we have a linear functional \( \phi \) on \( V^{A^*} \times A^* \). The degree to which \( x \) entails \( y \) is then given defined as

\[
\frac{\phi(\tilde{x} \land \tilde{y})}{\phi(\tilde{x})}
\]

This has many of the properties we would expect from a degree of entailment. In certain cases \( \phi \) can be used to make the space an Abstract Lebesgue space [1], in which case we can interpret the above definition as a conditional probability. This idea, and methods of defining \( \phi \) are discussed in detail in [3].
For the vector space $V$ generated by projections on $L^\infty(\Lambda)$, one possible definition of $\phi$ would be given by

$$\phi(u) = \sum_{l \in \Lambda} P(l) \|u(\epsilon, \epsilon)e_l\|$$

where $P(l)$ is some probability distribution over elements of $\Lambda$. The interpretation of this is: all contexts except those consisting of a pair of empty strings are ignored (so only strings that have logical interpretations make a contribution to the value of the linear functional); the value is then given by summing over all strings of the logical language and multiplying the probability of each string $l$ by the size of the vector resulting from the action of the operator on the basis element corresponding to $l$. The probability distribution over $\Lambda$ needs to be estimated, this could perhaps be done using machine learning:

- Given a corpus, build a model for each sentence in the corpus and its negation using a model builder
- Train a support vector machine using the models for the sentence and its negation as the two classes. This requires defining a kernel on models.
- Given a new string $l \in \Lambda$, build a model, then use the probability estimate of the support vector machine for the model belonging to the positive class, together with a normalisation function.

### 3.5 Computational Issues

In general it will be very hard to compute the degree of entailment between two strings using the preceding definitions. The number of logical interpretations that need to be considered increases exponentially with the length of the string. One possible way of tackling this would be to use Monte-Carlo techniques, for example, sampling dimensions of the vector space and computing degrees of entailment for the sample.

A more principled approach would be to exploit symmetries in the mapping $\rho$ from strings of aspects to the logical language. A further possibility is that a deeper analysis of the algebraic properties of context algebras leads to a simpler method of computation. Further work is undoubtedly necessary to tackle this problem.

### 4 Conclusion

We have introduced a more general definition of context algebras, and shown how they can be used to combine vector-based lexical semantics with logic based semantics at the sentence level. Whilst computational issues remain to be resolved, our approach allows the reuse of the abundance of work in logic-based natural language semantics.
References

[1] Yuri A. Abramovich and Charalambos D. Aliprantis. *An Invitation to Operator Theory*. American Mathematical Society, 2002.

[2] Charalambos D. Aliprantis and Owen Burkinshaw. *Positive Operators*. Academic Press, 1985.

[3] Daoud Clarke. *Context-theoretic Semantics for Natural Language: an Algebraic Framework*. PhD thesis, Department of Informatics, University of Sussex, 2007.

[4] Daoud Clarke, David Weir, and Rudi Lutz. Algebraic approaches to compositional distributional semantics. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS-2011)*, pages 325–329. Association for Computational Linguistics, 2011.

[5] Zellig Harris. *Mathematical Structures of Language*. Wiley, New York, 1968.

[6] P. Hinman. *Fundamentals of Mathematical Logic*. A. K. Peters, 2005.

[7] Anne Preller and Mehrnoosh Sadrzadeh. Bell states and negative sentences in the distributed model of meaning. *Electronic Notes in Theoretical Computer Science*, 270(2):141–153, 2011. Proceedings of the 6th International Workshop on Quantum Physics and Logic (QPL 2009).

[8] Julie Weeds, David Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *Proceedings of Coling 2004*, pages 1015–1021, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING.