Lambdas, Vectors, and Word Meaning in Context

Reinhard Muskens[1] and Mehrnoosh Sadrzadeh[2]

[1] Tilburg University
r.a.muskens@gmail.com
[2] Queen Mary University of London
mehrnoosh.sadrzadeh@qmul.ac.uk

Abstract

We provide Lambda Logical Forms, which we think of as a reasonably neutral interface between syntax and semantics, with two interpretations. One interpretation is very close to a standard Montagovian semantics, be it that it allows the extensions of certain terms to be dependent on constants denoting vectors. A second interpretation constrains the values of these constants so that a form of word sense disambiguation in context results.

1 Introduction

Formal semantics and distributional semantics have complementary virtues. One approach explains how linguistic expressions can describe features of our surroundings, how a sentence can be true in one situation, but false in another, and what it means for expressions to be in a relation of entailment. The other approach provides a model of how the meanings of words can depend on other words and comes with a notion of similarity of meaning that often corresponds well with human judgements. One theory has a convincing story about the composition of phrasal meanings from pre-given word meanings, while the other actually provides those word meanings. One framework excels in the treatment of functional, especially logical, words, the other in the treatment of content words. And so forth.

How can the two approaches be combined? In previous work (Muskens and Sadrzadeh [8, 7]) we have shown how a Montague-like framework can be used to provide linguistic expressions—more precisely, abstract lambda terms that can stand proxy for linguistic expressions—with a vector-based semantics. Since the set-up made it possible to also provide those abstract lambda terms with a standard truth-functional semantics, a combination was obtained. But there was no communication between the two forms of semantics, which is clearly not satisfactory. In this paper we will remedy this by providing a set-up in which a truth-functional and a distributional component communicate through shared constants denoting vectors associated with word meaning in context.

2 Abstract Lambda Terms and Object Lambda Terms

Let us first explain the technical context that we assume here. It is clear that a semantic theory must be associated with a theory of syntax in order to be able to make predictions about the form-meaning relation in language. But there are many syntactic theories on the market and, in order to be able to avoid a choice between them, we will make use of some techniques from Abstract Categorial Grammars.[1] These will allow us to abstract away from the details of syntax and the details of various proposals for providing syntactic structures with a semantics. We

[1] De Groote [4]; see also Muskens [9, 10]. De Groote's Abstract Categorial Grammars (ACGs) and Muskens' Lambda Grammars were independently conceived in 2001, but are very similar. While ACGs can be used as a theory of surface syntax, this will not be the way we will use them in this paper.

constants	type
JOHN$_k$, SUE$_k$, MARY$_k$, ...	$(DS)S$
WOMAN$_k$, BALL$_k$, PARTY$_k$, FLU$_k$, ...	N
TALL$_k$, RED$_k$, STONE$_k$, ...	NN
SMOKE$_k$, TALK$_k$, RUN$_k$, ...	DS
LOVE$_k$, THROW$_k$, CATCH$_k$, ...	DDS
BELIEVE$_k$, CLAIM$_k$, HOPE$_k$, ...	SDS
WHO	$(DS)NN$
EVERY, A, NO, THE, MOST, ...	$N(DS)S$
AND, OR	$(\vec{\alpha}S)(\vec{\alpha}S)\vec{\alpha}S$

Table 1: Abstract constants (for each $k \in \mathbb{N}$) used for generating Lambda Logical Forms. AND and OR are assigned to each type of the form $(\vec{\alpha}S)(\vec{\alpha}S)\vec{\alpha}S$, where $\vec{\alpha}$ is any sequence of types.

will work with a level of abstract lambda terms that can be associated with syntactic structures in any of the usual ways but can also have various more concrete interpretations, as will be explained shortly.

The typed lambda terms that will form this interface of abstract terms will be called Lambda Logical Forms (LLFs). The types of these LLFs are defined to be the smallest set of strings such that (a) D, N, and S are (basic) types[2] and (b) whenever α and β are types, $(\alpha\beta)$ is a type.[3] We consider lambda terms over this set of types.[4] A *combinator* will be a closed lambda term not containing constants and a lambda term is called *linear* if every binder λX in it binds exactly one X. In Table 1 we have given a collection of constants, many of which must be subscripted with some $k \in \mathbb{N}$. We are interested in *occurrences* of content words, and each distinct occurrence of a same content word will be associated with a constant carrying a unique subscript. The set of LLFs is defined as the smallest set such that the following hold.

- Every constant in Table 1 is an LLF of the type(s) it is associated with in that table;
- every typed linear combinator is an LLF;
- if M is an LLF of type $\alpha\beta$ and N is an LLF of type α, then (MN) is an LLF of type β, provided no subscript in M also occurs in N;
- if M is an LLF and M is λ-convertible to a linear term M', then M' is also an LLF.

As examples of LLFs, here are first some linear combinators. (We use ξ as a variable of type D, $\mathcal{P}$ as a variable of type DS, $\mathcal{R}$ as a variable of type DDS, and $\mathcal{Q}$ as a variable of type $(DS)S$).

[2] D will be the type of names, N the type of nominal phrases, and S the type of sentences.

[3] This gives types with lots of parentheses in them, but outer parentheses will never be written and nor will parentheses be written if they can be recovered by the rule that association is to the right (so that DDS, the type of transitive verbs, is short for $(D(DS))$, for example).

[4] We will use the notation for lambda terms that is standard in formal work (see Barendrecht [1], for example), but not, alas, in linguistic applications. This means we will write (MN) (not $M(N)$) for the result of applying M to N and use $(\lambda X.M)$ (not $\lambda X(M)$) for lambda-abstraction. Outer parentheses can be removed and MNO is short for $(MN)O$ (i.e. association is to the left). But we will often refrain from removing (all) parentheses. The reason is that the structure of lambda terms in official notation is often quite close to that of the linguistic expressions they formalise, much closer in fact than the linguistic notation. This concerns hierarchical (dominance) aspects of the terms, not aspects to do with the order in which constants appear (linear precedence).

(1) a. $\lambda\xi\lambda\mathcal{P}.\mathcal{P}\xi$

b. $\lambda\mathcal{R}\lambda\mathcal{Q}_o\lambda\xi_s.\mathcal{Q}_o(\lambda\xi_o.\mathcal{R}\xi_o\xi_s)$

c. $\lambda\mathcal{R}\lambda\mathcal{Q}_o\lambda\mathcal{Q}_s.\mathcal{Q}_o(\lambda\xi_o.\mathcal{Q}_s(\mathcal{R}\xi_o))$

d. $\lambda\mathcal{R}\lambda\xi_1\lambda\xi_2.\mathcal{R}\xi_2\xi_1$

The reader will recognise (1a) as 'Montague Raising' (applied to, say, J of type D, it will give $\lambda\mathcal{P}.\mathcal{P}J$ of type $(DS)S$[5]), while (1b) and (1c) are forms of 'Argument Raising' (Hendriks [5]). Not all linear combinators are meaning preserving type raisers, however. (1d), for example, will change a relation to its converse. If the language of LLFs is used as an arbitrary interface with syntax, i.e. as an interface that virtually any syntactic theory can connect with, there is no guarantee that application of linear combinators will be meaning-preserving.[6]

In (2) some examples of LLFs of type S are given.

(2) a. Every man loves a woman
$((\textsc{every man}_0)(\lambda\xi_s.((\textsc{a woman}_1)(\lambda\xi_o.\textsc{love}_2\ \xi_o\xi_s))))$

b. Every man loves a woman
$((\textsc{a woman}_0)\lambda\xi.((\textsc{every man}_1)(\textsc{love}_2\ \xi)))$

c. Every tall woman smokes
$((\textsc{every}(\textsc{tall}_0\ \textsc{woman}_1))\textsc{smoke}_2)$

d. Sue loves and admires a stockbroker
$(\textsc{a stockbroker}_0)\lambda\xi.\textsc{sue}_1(\textsc{and admire}_2\ \textsc{love}_3\ \xi)$

e. Bill admires but Anna despises every cop
$(\textsc{every cop}_0)\textsc{and}(\lambda\xi.\textsc{anna}_1(\textsc{despise}_2\ \xi))(\lambda\xi.\textsc{bill}_3(\textsc{admire}_4\ \xi))$

f. The witch who Bill claims Anna saw disappeared
$\textsc{the}(\textsc{who}(\lambda\xi.\textsc{bill}_0(\textsc{claim}_5(\textsc{anna}_1(\textsc{see}_2\ \xi))))\textsc{witch}_3)\textsc{disappear}_4$

The reader will hopefully agree that any syntactic theory that comes with a way to associate syntactic structures with some form of lambda-based semantics, can also be coupled with LLFs.

But LLFs must be given a further interpretation in order to be useful for semantics. We explain how such a further interpretation can be given with the help of *type* and *term homomorphisms* (De Groote [4]). If $\mathcal{B}$ is some set of basic types, we write $TYP(\mathcal{B})$ for the smallest set containing $\mathcal{B}$ such that $(\alpha\beta) \in TYP(\mathcal{B})$ whenever $\alpha, \beta \in TYP(\mathcal{B})$. A function η from types to types is said to be a *type homomorphism* if $\eta(AB) = (\eta(A)\eta(B))$, whenever $\eta(AB)$ is defined. It is clear that a type homomorphism η with domain $TYP(\mathcal{B})$ is completely determined by the values of η for types $\alpha \in \mathcal{B}$. For example, let $\mathcal{B} = \{D, N, S\}$, the set of basic types of our LLFs, and let γ be the type homomorphism with domain $TYP(\{D, N, S\})$ such that $\gamma(D) = e$, $\gamma(N) = est$, and $\gamma(S) = st$ (as usual, e is for entities, t is for truth values, and s is for possible worlds). Then $\gamma(NN) = (est)est$, $\gamma(DS) = est$, $\gamma(DDS) = eest$, $\gamma(SDS) = (st)est$, $\gamma(N(DS)S) = (est)(est)st$, etc.

A function ϑ from lambda terms to lambda terms is a *term homomorphism based on* η if η is a type homomorphism and, whenever M is in the domain of ϑ:

[5] In fact, in Table 1 we have categorised constants such as $\textsc{john}_k$ directly in the type $(DS)S$.

[6] See Muskens [10] for a set-up in which application of linear combinators does not lead to form-meaning mismatches, because permutations in syntax and semantics always occur in tandem.

constant c	type c	c°	type c°
JOHN_k	$(DS)S$	$\lambda P.P\mathbf{j}$	$(est)st$
WOMAN_k	N	$woman$	est
RED_k	NN	$\lambda P\lambda x.red\ x \wedge Px$	$(est)est$
RUN_k	DS	run	est
THROW_k	DDS	$throw$	$eest$
BELIEVE_k	SDS	$\lambda p\lambda x\lambda w.\forall w'(Bxww' \rightarrow pw')$	$(st)est$
WHO	$(DS)NN$	$\lambda P'\lambda P\lambda x\lambda w.P'xw \wedge Pxw$	$(est)(est)est$
EVERY	$N(DS)S$	$\lambda P'\lambda P\lambda w.\forall x(P'xw \rightarrow Pxw)$	$(est)(est)st$
A	$N(DS)S$	$\lambda P'\lambda P\lambda w.\exists x(P'xw \wedge Pxw)$	$(est)(est)st$
AND	$(\vec{\alpha}S)(\vec{\alpha}S)\vec{\alpha}S$	$\lambda R'\lambda R\lambda\vec{X}\lambda w.R'\vec{X}w \wedge R\vec{X}w$	
OR	$(\vec{\alpha}S)(\vec{\alpha}S)\vec{\alpha}S$	$\lambda R'\lambda R\lambda\vec{X}\lambda w.R'\vec{X}w \vee R\vec{X}w$	

Table 2: A term homomorphism $(\cdot)^\circ$ sending (some) LLFs to object terms.

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

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

Clearly, a term homomorphism ϑ with domain the set of LLFs is completely determined by the values $\vartheta(c)$ for LLF constants c.

Here is an example of how this can be used. In Table 2 we have (partially) defined a term homomorphism $(\cdot)^\circ$ based on γ sending LLF constants to certain translations. (The types of variables and constants used in this table are given in a footnote.[7]) For the moment, and since this is merely an example, the translation is not sensitive to the values of subscripts on constants at all.

If this definition is extended to all LLF constants, we will automatically also get translations of all complex LLFs. Consider the LLF in (2a), for example. It's translation image under $(\cdot)^\circ$ in (3a) is easily seen to be equal to (3b), since $(\cdot)^\circ$ is defined to be a term homomorphisms. But, in view of the translations of constants in Table 2 (and similar ones that we leave to the reader), this is equal to (3c), which reduces to (3d) in the usual way.

(3) a. $((\text{EVERY MAN}_0)\lambda\xi_s.(\text{A WOMAN}_1)(\lambda\xi_o.\text{LOVE}_2\ \xi_o\xi_s))^\circ$

b. $(\text{EVERY}^\circ\ \text{MAN}_0^\circ)\lambda x.(\text{A}^\circ\ \text{WOMAN}_1^\circ)(\lambda y.\text{LOVE}_2^\circ\ yx)$

c. $((\lambda P'\lambda P\lambda w.\forall x(P'xw \rightarrow Pxw))man)$
$\lambda x.((\lambda P'\lambda P\lambda w.\exists x(P'xw \wedge Pxw))woman)(\lambda y.love\, yx)$

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

[7] Using $A : \tau$ as shorthand for 'term A is of type τ', we have, for variables: $x, y, z : e$, $P : est$, $p : st$, $w : s$, $R : \vec{\alpha}S$, $\vec{X} : \vec{\alpha}$. For constants: $\mathbf{j} : e$, $woman$, red, $run : est$, $throw : eest$, $B : esst$.

constant c	type c	$c^\bullet$	type $c^\bullet$
JOHN$_k$	$(DS)S$	$\lambda Z.Z\mathbf{john}$	$(VV)V$
WOMAN$_k$	N	$\mathbf{woman}$	V
RED$_k$	NN	$\lambda v.(\mathbf{red} \times v)$	VV
RUN$_k$	DS	$\lambda v.(\mathbf{run} \times v)$	VV
THROW$_k$	DDS	$\lambda uv.(\mathbf{throw} \times_2 u) \times v$	VVV
BELIEVE$_k$	SDS	$\lambda uv.(\mathbf{believe} \times_2 u) \times v$	VVV
WHO	$(DS)NN$	$\lambda Zv.v \dotplus (Zv)$	$(VV)VV$
EVERY	$N(DS)S$	$\lambda vZ.Z(\mathbf{every} \times v)$	$V(VV)V$
A	$N(DS)S$	$\lambda vZ.Z(\mathbf{a} \times v)$	$V(VV)V$
AND	$(\vec{\alpha}S)(\vec{\alpha}S)(\vec{\alpha}S)$	$\lambda R'\lambda R\lambda\vec{X}.R\vec{X} \dotplus R'\vec{X}$	
OR	$(\vec{\alpha}S)(\vec{\alpha}S)(\vec{\alpha}S)$	$\lambda R'\lambda R\lambda\vec{X}.R\vec{X} + R'\vec{X}$	

Table 3: A term homomorphism $(\cdot)^\bullet$ sending LLFs of type S to terms denoting vectors.

Applied to a constant $\mathbf{w}$ (the 'actual world') the term in (3d) provides the desired truth conditions: $\forall x\,(man\,x\mathbf{w} \to \exists y\,(woman\,y\mathbf{w} \wedge love\,yx\mathbf{w}))$. The reader may find it amusing to translate other LLFs in a similar fashion and to experiment with alternative definitions of LLFs and of the type and term homomorphisms used. Our main point for the moment is that, given a collection of LLFs, the only thing needed for providing them with a semantics is to define a type homomorphism, plus a term homomorphism based on it.

3 Vectors in Type Logic

Since we want to combine truth-conditional semantics with vector semantics and use lambdas for composition, we must have a type theory that is able to talk about vectors over some field. For this field we choose the reals, as is usual. In order to have the latter available, we need a basic type $\mathbb{R}$ and constrain the models under consideration in such a way that the objects of type $\mathbb{R}$ are real numbers. Additionally, constants such as $0 : \mathbb{R}$, $1 : \mathbb{R}$, $+ : \mathbb{RRR}$, $\cdot : \mathbb{RRR}$, and $< : \mathbb{RR}t$[8] must have their usual interpretation. Fortunately, the problem of axiomatising the reals has already been solved for us by Alfred Tarski, who in [11] discusses two sets of (second-order) axioms for the real numbers. Adopting Tarski's axioms will ensure that the domain $D_\mathbb{R}$ of type $\mathbb{R}$ will equal the reals in full models.[9]

Vectors can now be introduced as objects of type $I\mathbb{R}$, where I is interpreted as some finite index set. Think of I as a set of words; if a phrase is associated with a vector $\mathbf{v} : I\mathbb{R}$, $\mathbf{v}$ assigns a real to each word, which gives information about the company the phrase keeps.[10] We abstract from the order present in vectors here. Since $I\mathbb{R}$ will be used often, we will abbreviate it as V. Note that $II\mathbb{R}$, abbreviated as M, can be associated with the type of *matrices* and $III\mathbb{R}$, abbreviated as C, with the type of *cubes*. In general, $I^n\mathbb{R}$ will be the type of tensors of rank n.

We need a toolkit of functions combining vectors, matrices, cubes, etc. Here are some definitions. The following typographical conventions are used for variables: r is of type $\mathbb{R}$; v and u are of type V; i, j, and ℓ are of type I; and m and c are of types M and C respectively.

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

[9] In generalised models that are not full the domain $D_\mathbb{R}$ will contain nonstandard reals, but will still satisfy the first-order theory of the reals.

[10] For exposition, we will work with a single index type I. Alternatively, several index types might be considered, so that phrases of distinct categories are allowed to live in their own space.

Indices are written as subscripts—v_i is syntactic sugar for vi.

$$
\begin{aligned}
* &:= \lambda rvi.r \cdot v_i : \mathbb{R}VV \\
\dot{+} &:= \lambda vui.v_i + u_i : VVV \\
\odot &:= \lambda vui.v_i \cdot u_i : VVV \\
\times &:= \lambda mvi.\sum_j m_{ij} \cdot v_j : MVV \\
\times_2 &:= \lambda cvij.\sum_\ell m_{ij\ell} \cdot v_\ell : CVM
\end{aligned}
$$

The reader will recognise $*$ as scalar multiplication, $\dot{+}$ as pointwise addition, $\odot$ as pointwise multiplication, $\times$ as matrix-vector and $\times_2$ as cube-vector multiplication. Other relevant operations are easily defined.

4 An Aside: Vectors All the Way Up

As a further example of how a set of LLFs can be provided with a semantics, we provide our fragment with a vector semantics in which phrases of all categories are associated with vectors, or vector-based functions. The type homomorphism we employ will be the function γ_1, defined by $\gamma_1(D) = \gamma_1(N) = \gamma_1(S) = V$, i.e. names, common nouns, and sentences all go to vectors. A term homomorphism $(\cdot)^\bullet$ based on γ_1 is given in Table 3. In this table the constants **john**, and **woman** denote vectors (type V), while **red**, **run**, **every**, and **a** denote matrices (type M), while **throw** and **believe** denote cubes (type C).[11] The variable Z is of type VV here.

We now find consequences of our translation such as the ones in (4).

(4) a. $((\text{A WOMAN}_0)\lambda\xi.((\text{EVERY MAN}_1)(\text{LOVE}_2\ \xi)))^\bullet =$
$(\mathbf{love} \times_2 (\mathbf{a} \times \mathbf{woman})) \times (\mathbf{every} \times \mathbf{man})$

b. $(((\text{EVERY}(\text{TALL}_0\ \text{WOMAN}_1))\text{SMOKE}_2))^\bullet = \mathbf{smoke} \times (\mathbf{every} \times (\mathbf{tall} \times \mathbf{woman}))$

c. $((\text{A STOCKBROKER}_0)\lambda\xi.\text{SUE}_1(\text{AND ADMIRE}_2\ \text{LOVE}_3\ \xi))^\bullet =$
$((\mathbf{love} \times_2 (\mathbf{a} \times \mathbf{stockbroker})) \times \mathbf{sue}) \dot{+} ((\mathbf{admire} \times_2 (\mathbf{a} \times \mathbf{stockbroker})) \times \mathbf{sue})$

d. $((\text{EVERY COP}_0)\text{AND}(\lambda\xi.\text{ANNA}_1(\text{DESPISE}_2\ \xi))(\lambda\xi.\text{BILL}_3(\text{ADMIRE}_4\ \xi)))^\bullet =$
$((\mathbf{admire} \times_2 (\mathbf{every} \times \mathbf{cop})) \times \mathbf{bill}) \dot{+} ((\mathbf{despise} \times_2 (\mathbf{every} \times \mathbf{cop})) \times \mathbf{anna})$

e. $(\text{THE}(\text{WHO}(\lambda\xi.\text{BILL}_0(\text{CLAIM}_5(\text{ANNA}_1(\text{SEE}_2\ \xi))))\text{WITCH}_3)\text{DISAPPEAR}_4)^\bullet =$
$\mathbf{disappear} \times (\mathbf{the} \times (\mathbf{witch} \dot{+} ((\mathbf{claim} \times_2 ((\mathbf{see} \times_2 \mathbf{witch}) \times \mathbf{anna})) \times \mathbf{bill})))$

It is of course the question whether it will be possible to harvest all the vectors, matrices and cubes that are necessary to make such translations more than a theoretical exercise. And, if so, it will still be the case that only empirical testing can answer the question how well a model such as this one will actually do on given tasks (say predicting perceived similarity of sentences). But, given that LLFs can form the output of many syntactic frameworks (and many parsers), there is at least no *theoretical* hurdle that must be overcome if we want to associate linguistic structures with a vector semantics.

[11] Compare Baroni and Zamparelli [2].

constant c	type c	$c^\dagger$	type $c^\dagger$
JOHN$_k$	$(DS)S$	$\lambda P.P(\mathbf{jv}^k)$	$(est)st$
WOMAN$_k$	N	$woman\,\mathbf{v}^k$	est
RED$_k$	NN	$\lambda P\lambda x.red\,\mathbf{v}^k x \wedge Px$	$(est)est$
RUN$_k$	DS	$run\,\mathbf{v}^k$	est
THROW$_k$	DDS	$throw\,\mathbf{v}^k$	$eest$
BELIEVE$_k$	SDS	$\lambda p\lambda x\lambda w.\forall w'(Bxww' \rightarrow pw')$	$(st)est$
WHO	$(DS)NN$	$\lambda P'\lambda P\lambda x\lambda w.P'xw \wedge Pxw$	$(est)(est)est$
EVERY	$N(DS)S$	$\lambda P'\lambda P\lambda w.\forall x(P'xw \rightarrow Pxw)$	$(est)(est)st$
A	$N(DS)S$	$\lambda P'\lambda P\lambda w.\exists x(P'xw \wedge Pxw)$	$(est)(est)st$
AND	$(\vec{\alpha}S)(\vec{\alpha}S)\vec{\alpha}S$	$\lambda R'\lambda R\lambda\vec{X}\lambda w.R'\vec{X}w \wedge R\vec{X}w$	
OR	$(\vec{\alpha}S)(\vec{\alpha}S)\vec{\alpha}S$	$\lambda R'\lambda R\lambda\vec{X}\lambda w.R'\vec{X}w \vee R\vec{X}w$	

Table 4: Term homomorphism $(\cdot)^\dagger$. As $(\cdot)^\circ$, but with additional vector constants.

5 Vectors in Truth-conditional Semantics

The fragment given in the previous section provides linguistic phrases with a vector semantics, but not with one that could easily be used to make predictions about truth conditions or entailment. It can be combined with an interpretation such as $(\cdot)^\circ$, and then, for each LLF Λ, Λ will come with a truth-conditional interpretation Λ° and a vector interpretation $\Lambda^\bullet$, but there is no interaction between the two.

We now want to present a model in which two interpretations, a distributional and a truth-conditional one, interact. The distributional interpretation will help to constrain word senses in context, while the truth-functional homomorphism will work on the basis of the senses thus constrained. Our inspiration for the distributional part has been Erk and Padó [3], but we replace their *structured vector space model* with a term homomorphism and their direct computation of word senses by a constraint-based approach.

The idea is as follows. Many words come, not with one, but with a whole collection of possible meanings. For example, the internet version of Merriam-Webster's dictionary gives no less than 18 different senses of the word *throw* (*throw a party*, *throw a ball*, *throw a tantrum*, ...). We assume that words typically come with a finite number of senses, each represented by a prototypical vector (see Kartsaklis et al. [6] for a closely related idea). While the senses of a word are often obviously related, two senses of the same word may well be associated with entirely different extensions.

The linguistic question is now how language users choose between these senses and the technological question is how a machine could be persuaded to do it. We focus on the linguistic question. Part of its answer must be that words do not only come with senses, but that senses also come with selectional preferences for other senses and that there is some mechanism for satisfying these preferences in the best possible way.

Let us first provide a truth-conditional set-up with vector senses, to be constrained shortly. In Table 4 we have defined a term homomorphism $(\cdot)^\dagger$, based on the type homomorphism γ that $(\cdot)^\circ$ was also based on. In fact, $(\cdot)^\dagger$ is very close to $(\cdot)^\circ$, but we now make use of the subscripts on some of the abstract constants, using them as superscripts on certain vector constants (the constants $\mathbf{v}^k$, of type V) in the translation. Constants such as *woman*, and *red* now have one extra argument place in order to provide for these constants. In particular, $\mathbf{j}$ is of type Ve, *woman* and *red* are of type $Vest$, and *throw* is of type $Veest$. Applied to constants of type V,

constant c	$c^\ddagger$	type $c^\ddagger$
JOHN$_k$	$\lambda\mathcal{Z}.\mathcal{Z}\lambda u.(\mathsf{john}\,\mathbf{v}^k \wedge u = \mathbf{v}^k)$	$(bt)t$
GRAPEFRUIT$_k$	$\lambda u.(\mathsf{grapefruit}\,\mathbf{v}^k \wedge u = \mathbf{v}^k)$	b
RED$_k$	$\lambda\rho\lambda u.(\mathsf{red}\,\mathbf{v}^k \wedge \rho u \wedge d(u, \mathbf{h} \times \mathbf{v}^k) < \varepsilon_k)$	bb
RUN$_k$	$\lambda\rho.\exists u(\mathsf{run}\,\mathbf{v}^k \wedge \rho u \wedge d(u, \mathbf{s} \times \mathbf{v}^k) < \varepsilon_k)$	bt
THROW$_k$	$\lambda\rho_1\rho_2.\exists uu'(\mathsf{throw}\,\mathbf{v}^k \wedge \rho_1 u \wedge \rho_2 u'$ $\wedge\, d(u, \mathbf{o} \times \mathbf{v}^k) < \varepsilon_k \wedge d(u', \mathbf{s} \times \mathbf{v}^k) < \varepsilon'_k)$	bbt
BELIEVE$_k$	$\lambda q\rho.\exists u(q \wedge \mathsf{believe}\,\mathbf{v}^k \wedge \rho u \wedge d(u, \mathbf{s} \times \mathbf{v}^k) < \varepsilon_k)$	tbt
WHO	$\lambda\mathcal{Z}\rho u.\mathcal{Z}\rho \wedge \rho u$	$(bt)bb$
EVERY, A, THE	$\lambda\rho\mathcal{Z}.\mathcal{Z}\rho$	$b(bt)t$
AND	$\lambda\mathcal{R}'\mathcal{R}\vec{X}.\mathcal{R}\vec{X} \wedge \mathcal{R}'\vec{X}$	
OR	$\lambda\mathcal{R}'\mathcal{R}\vec{X}.\mathcal{R}\vec{X} \vee \mathcal{R}'\vec{X}$	

Table 5: A term homomorphism $(\cdot)^\ddagger$ sending LLFs of type S to statements describing vectors.

they result in terms of the types they were given previously, and the rest of the set-up is as in the $(\cdot)^\circ$ homomorphism.

We get identities such as the one in (5), with vector constants helping to denote senses. An expression such as $run\mathbf{v}^2$ is intended to point at a particular sense of running (is it water running? an animal running? are we running out of time?).

(5) a. $(\text{JOHN}_1\ \text{RUN}_2)^\dagger = run\mathbf{v}^2(\mathbf{jv}^1)$ $\quad [=_\eta \lambda w.run\mathbf{v}^2(\mathbf{jv}^1)w]$

b. $((\text{A PARTY}_1)(\lambda\xi.\text{MARY}_2(\text{THROW}^3\ \xi))^\dagger =$
$\lambda w.\exists y\,(party\,\mathbf{v}^1 yw \wedge \rightarrow throw\,\mathbf{v}^3 y(\mathbf{mv}^2)w))$

c. $((\text{A UNICORN}_1)(\lambda\xi.(\text{EVERY DOG}_2)(\text{CHASE}_3\ \xi)))^\dagger =$
$\lambda w.\exists y\,(unicorn\,\mathbf{v}^1 yw \wedge \forall x\,(dog\,\mathbf{v}^2 xw \rightarrow chase\,\mathbf{v}^3 yxw))$

In order to make this work, the denotations of the $\mathbf{v}^k$ must be constrained and we do this with the help of a second homomorphism $(\cdot)^\ddagger$, defined in Table 5, and based on a type homomorphism γ' with $\gamma'(S) = t$ and $\gamma'(D) = \gamma'(N) = Vt$. Since Vt (sets of vectors) will be used often, we abbreviate it as b. $(\cdot)^\ddagger$ does not generate the usual kind of semantic objects, its sole purpose is to associate each LLF of type S with a conjunction of two kinds of statements:

- Statements saying that a certain vector belongs to the set of prototypes associated with a certain word. Examples: $\mathsf{run}\,\mathbf{v}^2$, $\mathsf{john}\,\mathbf{v}^1$, etc. Here the first constant is always of type Vt (i.e. b) and the second of type V. [Warning: $\mathsf{run}\,\mathbf{v}^2$ should well be distinguished from $run\,\mathbf{v}^2$.]

- Statements expressing that the cosine distance between two vectors is less than a given value. Example: $d(\mathbf{v}^1, \mathbf{s} \times \mathbf{v}^2) < \varepsilon_2$, which can be glossed as 'the distance between $\mathbf{v}^1$ and $\mathbf{s} \times \mathbf{v}^2$, the subject vector of $\mathbf{v}^2$, is sufficiently small. Here $\mathbf{s}$ is a matrix (type M).

The entries in Table 5 also mention other conjuncts, such as applicatons ρu and identities $u = \mathbf{v}^k$, but these only have intermediate importance in derivations.

Here is a simple example.

(6) $(\text{JOHN}_1\ \text{RUN}_2)^\ddagger = \mathsf{run}\,\mathbf{v}^2 \wedge \mathsf{john}\,\mathbf{v}^1 \wedge d(\mathbf{v}^1, \mathbf{s} \times \mathbf{v}^2) < \varepsilon_2$

We leave it to the reader to verify that the statement in (6) is correct. Note that in a last derivation step the predicate logical fact can be used that $\exists u(u = \mathbf{v} \wedge \varphi)$ is equivalent to the result of substituting $\mathbf{v}$ for u in φ. This is a general way of getting rid of existential quantifiers and identity statements that will recur often in derivations on the basis of $(\cdot)^\ddagger$.

The description generated as the right-hand side of (6) states that $\mathbf{v}^1$ is a vector associated with John, $\mathbf{v}^2$ is one of the prototype vectors associated with running, and that the distance between the John vector and a things-that-can-run vector associated with $\mathbf{v}^2$ is small. The idea that words come with vectors standing for their selectional preferences is taken from Erk and Padó [3], who "compute the selectional preference vector for word b and relation r as the weighted centroid of seen filler vectors $\vec{v}_a$". Here we associate the selectional preference vector not just with the word, but with each of the word senses.

Note that the statement $d(\mathbf{v}^1, \mathbf{s} \times \mathbf{v}^2) < \varepsilon_2$ in the right-hand side of (6) puts a *mutual* constraint on the vectors $\mathbf{v}^1$ and $\mathbf{v}^2$. Depending on the value of ε_2, certain combinations of values for $\mathbf{v}^1$ and $\mathbf{v}^2$ may well be excluded. This may be used to exclude certain models from consideration.

Let us have a look at some more images of LLFs under $(\cdot)^\ddagger$.

(7) a. $((\text{A}(\text{RED}_1\ \text{GRAPEFRUIT}_2))(\lambda\xi.\text{MARY}_3(\text{THROW}^4\ \xi)))^\ddagger =$
$\mathsf{throw}\,\mathbf{v}^4 \wedge \mathsf{red}\,\mathbf{v}^1 \wedge \mathsf{grapefruit}\,\mathbf{v}^2 \wedge d(\mathbf{v}^2, \mathbf{h} \times \mathbf{v}^1) < \varepsilon_1 \wedge \mathsf{mary}\,\mathbf{v}^3$
$\wedge\, d(\mathbf{v}^2, \mathbf{o} \times \mathbf{v}^4) < \varepsilon_4 \wedge d(\mathbf{v}^3, \mathbf{s} \times \mathbf{v}^4) < \varepsilon'_4$

b. $((\text{A UNICORN}_1)(\lambda\xi.(\text{EVERY DOG}_2)(\text{CHASE}_3\ \xi)))^\ddagger =$
$\mathsf{chase}\,\mathbf{v}^3 \wedge \mathsf{unicorn}\,\mathbf{v}^1 \wedge \mathsf{dog}\,\mathbf{v}^2 \wedge d(\mathbf{v}^1, \mathbf{o} \times \mathbf{v}^3) < \varepsilon_3 \wedge d(\mathbf{v}^2, \mathbf{s} \times \mathbf{v}^3) < \varepsilon'_3$

c. $(\text{THE}(\text{WHO}(\lambda\xi.\text{BILL}_0(\text{CLAIM}_5(\text{ANNA}_1(\text{SEE}_2\ \xi))))\text{WITCH}_3)\text{DISAPPEAR}_4)^\ddagger =$
$\mathsf{disappear}\,\mathbf{v}^4 \wedge \mathsf{see}\,\mathbf{v}^2 \wedge \mathsf{witch}\,\mathbf{v}^3 \wedge \mathsf{anna}\,\mathbf{v}^1 \wedge d(\mathbf{v}^3, \mathbf{o} \times \mathbf{v}^2) < \varepsilon_2 \wedge d(\mathbf{v}^1, \mathbf{s} \times \mathbf{v}^2) < \varepsilon'_2$
$\wedge\, \mathsf{claim}\,\mathbf{v}^5 \wedge \mathsf{bill}\,\mathbf{v}^0 \wedge d(\mathbf{v}^0, \mathbf{s} \times \mathbf{v}^5) < \varepsilon_5 \wedge d(\mathbf{v}^3, \mathbf{s} \times \mathbf{v}^4) < \varepsilon_4$

Note that in (7a) it is the vector connected with *grapefruit* that is constrained not to be too far from the object vector of *throw*. This is because the entry for RED_k in Table 5 'picks up' the value for u from its head, after which it can be picked up by further functors. In (7a) the vector for *witch* is not only constrained by the object vector of *see*, but also by the subject vector of *disappear*.

How can we define a notion of consequence on the basis of $(\cdot)^\dagger$ and $(\cdot)^\ddagger$? If Λ_1 and Λ_2 are LLFs, when does Λ_2 follow from Λ_1? We define a notion of entailment that is based on models that minimise the sum of the ε_k and ε'_k occurring in $(\Lambda_1)^\ddagger$ and $(\Lambda_2)^\ddagger$. Let us say that a model M is a δ-*model* if, (a) for no k, $\varepsilon_k > 1$ or $\varepsilon'_k > 1$ holds in M, (b) for only a finite number of k, $\varepsilon_k \neq 0$ or $\varepsilon'_k \neq 0$ holds in M, and (c) the sum of the nonzero values for ε_k and ε'_k in M is δ. Λ_2 *follows from* Λ_1 if there is a δ such that the following conditions hold.

- There is a δ-model M satisfying both $(\Lambda_1)^\ddagger$ and $(\Lambda_2)^\ddagger$;
- for every $\delta' \leq \delta$ and every δ'-model M, if M satisfies $(\Lambda_1)^\ddagger$, $(\Lambda_2)^\ddagger$, and $(\Lambda_1)^\dagger\mathbf{w}$, then M satisfies $(\Lambda_2)^\dagger\mathbf{w}$.

In the last clause, $\mathbf{w}$ is a fixed but arbitrary constant of type s that may be thought of as the actual world.

The idea here is that entailment holds if it there is transmission of truth in all models where the sum of the ε_k and ε'_k values are sufficiently low. As many vectors as possible need to be excluded as values for the $\mathbf{v}^k$. There may be ties, of course, in the sense that not all $\mathbf{v}^k$ are provided with a unique value, even in the best models (those with lowest δ), in which case there may be true ambiguity.

6 Conclusion

We have provided Lambda Logical Forms, with two interpretations. One interpretation is very close to a standard Montagovian semantics, be it that it allows the extensions of certain terms to be dependent on constants denoting vectors. The vectors can be thought of as prototypes associated with word senses. A second interpretation constrains the values of these constants so that a form of word sense disambiguation in context results. Since cosine distance must be minimised, many potential word senses are discarded. The disambiguation can be made sensitive to linguistic structure and is not restricted to local linguistic context.

7 Acknowledgments

We wish to thank the referees of Amsterdam Colloquium 2017 for their useful and encouraging comments.

References

[1] H. Barendrecht. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, 1981.

[2] Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in NLP*, EMNLP '10, pages 1183–1193. ACL, 2010.

[3] K. Erk and S. Padó. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906, 2008.

[4] Ph. de Groote. Towards Abstract Categorial Grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155, Toulouse, France, 2001. ACL.

[5] H.L.W.H. Hendriks. *Studied Flexibility: Categories and Types in Syntax and Semantics*. PhD thesis, University of Amsterdam, 1993.

[6] Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. Separating disambiguation from composition in distributional semantics. In *Proceedings of 17th Conference on Natural Language Learning (CoNLL)*. Sofia Bulgaria, 2013.

[7] R. Muskens and M. Sadrzadeh. Context Update for Lambdas and Vectors. In M. Amblard, Ph. de Groote, S. Pogodalla, and C. Retoré, editors, *Logical Aspects of Computational Linguistics: 9th International Conference*, pages 247–254. Springer, 2016.

[8] R. Muskens and M. Sadrzadeh. Lambdas and Vectors. Abstract presented at the ESSLLI 2016 workshop on Distributional Semantics and Linguistic Theory (DSALT), 2016.

[9] R. A. Muskens. Categorial Grammar and Lexical-Functional Grammar. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG01 Conference, University of Hong Kong*, pages 259–279, Stanford CA, 2001. CSLI Publications. http://cslipublications.stanford.edu/LFG/6/lfg01.html.

[10] R. A. Muskens. Language, Lambdas, and Logic. In Geert-Jan Kruijff and Richard Oehrle, editors, *Resource Sensitivity in Binding and Anaphora*, Studies in Linguistics and Philosophy, pages 23–54. Kluwer, 2003.

[11] Alfred Tarski. *Introduction to Logic and to the Methodology of Deductive Sciences*. Dover Publications, 1946.