Types, Meanings and Co-composition in Lexical Semantics

Nicholas Asher, Marta Abrusan, Tim van de Cruys, CNRS, Institut de Recherche en Informatique de Toulouse, France

1. Introduction

This paper investigates co-composition and its implications for type-theoretic approaches to semantics. While various type theories [1, 3, 17, 18, 19] have proposed rich and interesting theories of composition using types to account for phenomena like *coercion* and co-predication in which linguists have observed meaning shifts, it turns out that composition can affect the meanings of individual words in much more mundane ways. In fact as we will see, there is evidence from the computational approach to meaning known as *distributional semantics* (DS) that shifts in meaning of a predicate or an argument during composition are almost ubiquitous. We call this shifting of meaning in composition *co-composition*, following [27].

The reason why we're interested in co-composition is that a satisfactory theory of co-composition has important ramifications for how we think of types. While formal type theories are good at analyzing coercions, it's much less clear how they model the more mundane shifts in meaning. The data from distributional semantics leads us to suppose that types are not as we suppose in type theory. Instead, each basic type is a vector in some space $\mathcal{V}$ whose dimensions are syntactic/semantic constructs or more abstract features. Given this, the space of basic types is continuous in principle, with every vector in the vector space constituting a type. Only very few of these possible types are lexicalized as simple words in a given corpus or language. General types, or type presuppositions of the sort one of us has discussed in [1], can be found automatically on the basis of distributional generalizations involving simple types. This has a number of advantages. First, by using an appropriate algorithm, we are able to come up with type characterizations at different levels of granularity. Second, type characterizations might vary according to the context they appear in. Third, simple types are never plainly of one higher type or not. Since the space of types is continuous, there is no clean cutoff between those that are subtypes of a higher type and those that are not. This predicts, correctly, that we might find graded judgments concerning examples that involve category mistakes (cf. [20]).

The organization of our paper is follows. In the next section, we introduce the notion of co-composition, and in the third we look at co-composition from the viewpoint of distributional semantics. In the fourth section, we present one attempt to model co-composition in formal type theory using the treatment of coercion as a point of departure, and we then return in section 5 to the question of how this affects our thinking about the core of type theory, what types are. Then in a final section, we conclude.

2. Co-composition

Co-compositions come in different flavors. One type of co-composition is easily explained using type theories. An ambiguous word may be made less ambiguous by combining with other words. Consider for instance the word *suit*. It's ambiguous at least between the senses of denoting a set of legal documents or a set of articles of clothing.

(1) a. Julie filed a suit.
b. Julie wore a suit.

Here the predicate introduced by the verb, in particular the selectional restrictions this predicate imposes on its second or direct object argument disambiguates *suit*. Type theories that can represent the different senses of ambiguous words with disjoint types can model this sort of disambiguation by an inference that is logically sound: the predicate, by selecting one of the disjoint types as its selectional restriction, can confer a more specialized meaning to the argument.

But the composition of a predicate and its argument can exhibit other sorts of meaning shifts as well, which pose challenges for type theories. Consider the following adjectival common noun compositions:

(2) a. flat surface
b. flat country
c. flat tire
d. flat water
e. flat beer

In these examples the head noun affects the meaning of the modifier. If these data are well-known, the models of analysis for them are not. We could assume that adjectives are wildly ambiguous, roughly one sense for each noun with which they can combine. But that would miss certain logical relations. For instance, a mathematically flat surface does have something in common with a flat tire, and even with flat beer. We could say that adjectival types are polymorphic, but that simply labels the problem and doesn't solve the problem of how to draw the right inferences. There is a sense of flat that might be taken as fundamental, that of a flat surface, which then shifts to a different but related meaning when the adjective combines with nouns of type other than SURFACE. A theory of composition should tell us how the adjectival sense shifts when combined with a particular noun, and it should tell us whether that shifted sense holds of the lambda bound argument x in λx Composition(Adj-Noun)(x), which is the general logical form of an adjective-noun combination, whatever method one chooses to combine that adjective and noun.

The vast majority of adjectives in most languages are subsective; i.e., if x is a Adj N, it is an N. Those few adjectives that aren't subsective, like *former prisoner*, support inferences that subsective adjectives do under some sort of modal or temporal operator. For example, former prisoners were once prisoners, future candidates are candidates at some future point in time, possible difficulties are difficulties in some epistemic alternative, and fake guns and stone lions appear to be or look like guns and lions.[1] This has an important moral for composition; compositions should in general be *decomposable* in the sense that an Adj N combination should entail that there is an object of type N and that it has some properties given by the modified sense of the adjective. A theory of composition should also explain why this doesn't hold in the comparatively few cases of adjective-noun combination where the adjective isn't subsective.

Such "shifty" co-compositions occur also with verbs and their arguments.

(3) a. Mary directed a best grossing movie last year.
b. Mary directed John to take a seat.
c. Sam swallowed the wine.
d. Sam swallowed her anger.
e. The car ran into a wall.
f. The group ran into a problem.

There is at least a family resemblance between the meanings of *direct* in (3a) and (3b); in both examples Mary is ordering people to do things, but directing a movie involves a long complex interrelated sequence of actions involving many different people, whereas directing someone to take a seat does not. So it would seem here as though the object argument shifts the verb's meaning to a related though distinct sense. But the verb isn't simply ambiguous. As with the examples in (2), the senses of *direct* in (3a,b) or *run* in (3d,e) are related, whereas the meanings of *suit* are not so related. A theory of co-composition should explain the relatedness of these readings.

In addition, we see a requirement of decomposability for verb-argument composition: to swallow the wine or to swallow one's anger entails that there was something one swallowed, albeit in different senses. In addition, our semantics should entail that there was an event of swallowing something. In fact, this sort of decomposability seems to be a general feature of composition in the vast majority of cases. Of course, most formal methods of composition allow for forms of decomposability, in particular the proof-theoretic semantics that comes with most type-theoretic approaches to semantics.

[2] proposed a model for adjectival noun composition that grew out of an analysis of coercion [1]. Like co-composition, coercion happens during composition when the senses of the words to be

[1] [1] and many other approaches—e.g., [16, 24]—adopt this view for adjectives and modifiers.

combined or the sense of the predication itself shifts as a result of the composition. Originally developed for use in simply typed programming languages (see, e.g. [22]), coercions have been widely employed in linguistic semantics [25, 27, 26]. Very roughly, a coercion is a function from one semantic value or one type to another that is employed when some problem arises in the construction of meaning. Recently, coercions have been studied using typed theories of lexical semantics—e.g., the work of [1, 3] using the Type Composition Logic (TCL) and also in Modern Type Theory (MTT) [17, 18, 19, 2]. Coercions take place against a background of a theory of lexical meaning and meaning composition that takes selectional restrictions seriously. Normally a predication involving a predicate whose arguments do not meet its standard selectional restrictions will not result in a felicitous meaning. However, with *coercion* a predicate whose standard selectional restrictions are not met by its argument may still convey a coherent linguistic meaning because either one of the terms or the predication relation between predicate and argument is adjusted in some way so that the composition process may succeed. In a type-theoretic framework, this means that coercions occur only when there is a type incompatibility between a predicate's selectional restrictions and the type of its argument.

Adjective-noun composition and verb-argument composition also exploit mechanisms of meaning shift. But these differ from coercions, because there is no manifest type incompatibility between the selectional restrictions of *direct* and the different arguments of the verb in (3a,b). So then what triggers the shifts in co-composition? The theory of [2] is largely mute on this point, as is [1]. This is a central question for us in this paper. Before beginning on our journey to answer it though, let's look at some more motivation for co-compositionality, this time from the perspective of distributional semantics.

3. Co-compositionality from a distributional point of view

3.1. Vector space model of meaning

Distributional semantics looks at lexical semantics through a different but ultimately, we will argue, related lens to that used by type-theoretic semantics. Distributional approaches start from the assumption that a compact representation of the use of a word will suffice as a representation of its lexical meaning, a philosophical point of view that has roots in the later Wittgenstein [30], and was developed in some detail by Michael Dummett [12] and Robert Brandom [6].

Distributional semantics aims to automatically extract the meaning of words by looking at their distribution in text, and comparing these distributions within a mathematical model. The unsupervised nature of distributional semantics, and consequently its ability to process large volumes of text in a fully automatic way, make it an attractive method for a data-driven and broad-coverage model of lexical semantics. This section provides a succinct summary of the model; for an elaborated overview of semantic word space models, the interested reader is referred to [28]. Other influential articles on the subject are [5, 21, 10, 9],

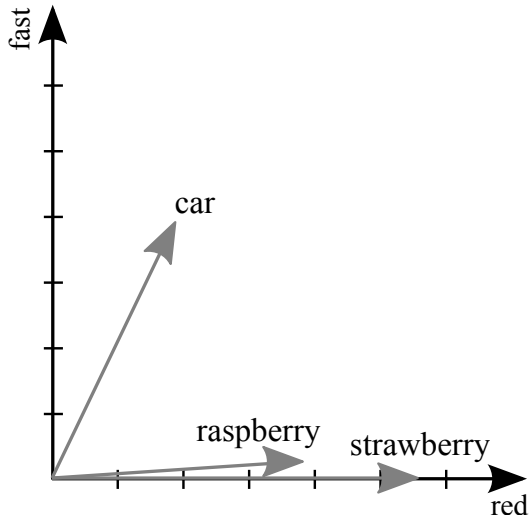

Figure 1: A graphical representation of a word vector space. The angle between the vectors for *strawberry* and *raspberry* is much smaller than the angle between the vectors for *strawberry* and *car*.

Within distributional semantics, the meaning of a word w is represented as a vector, such that each of the vector's dimensions captures the frequency of a particular context with which the word appears. By constructing words as vectors within a vector space model, it becomes straightforward to compute a similarity value for them.[2] A graphical representation of the process is given in figure 1.

There are many proposals as to what should be taken as the vector's context features: the documents that w appears in, other words within a limited distance of w, or particular syntactic constructions in which w occurs.[3] Note that context is a determining factor in the nature of the semantic similarity that is induced. A broad context window (e.g. a paragraph or document) yields broad, topical similarity, whereas a small context window or syntactic constructions yield tight, synonym-like similarity.

All the word vectors combined together make up a word matrix. Similarity calculations may be carried out using the matrix as such, but in many cases word space models undergo further mathematical processing, with the aim of extracting more salient semantic data. One typically makes use of dimensionality reduction or factorization algorithms, that aim to express an abundant number of overlapping feature dimensions (that make up the original word vectors) in terms of

[2] e.g. using the cosine measure, which computes the angle between two vectors.

[3] One of the syntactic features of the word *apple* might e.g. be eat_{dobj}, since *apple* appears as a direct object of the verb *eat*. Note that these syntactic contexts – also called dependency features – are generally automatically extracted using a dependency-based grammatical parser.

a reduced, limited number of meaningful dimensions. One well-known dimensionality reduction technique is singular value decomposition (SVD). SVD decomposes the original matrix into three other matrices, such that the most important dimensions captured by these matrices are said to represent latent semantic dimensions, according to which words and contexts can be represented more efficiently. In our research, we also make use of non-negative matrix factorization (NMF). The key idea is that a non-negative matrix **A** is factorized into two other non-negative matrices, **W** and **H**

$$\mathbf{A}_{i\times j} \approx \mathbf{W}_{i\times k}\mathbf{H}_{k\times j} \tag{4}$$

where k is much smaller than i, j so that both instances and features are expressed in terms of a few components. The main difference with SVD is that non-negative matrix factorization enforces the constraint that all three matrices must be non-negative, so all elements must be greater than or equal to zero. The non-negative constraint enforces the resulting representation to be *parts-based*, which is generally useful for language data. Using the minimization of the Kullback-Leibler divergence (an information-theoretic measure) as an objective function, we want to find the matrices **W** and **H** for which the Kullback-Leibler divergence between **A** and **WH** (the multiplication of **W** and **H**) is the smallest. This factorization is carried out through the iterative application of update rules. When the factorization is finished, both words and context are linked to a limited number of latent factors, which represent meaningful, latent semantic dimensions.

Finally, note that, due to their non-negative nature, the NMF matrices can be interpreted probabilistically: from the original matrix **A**, we are able to infer $p(\mathbf{d}|\mathbf{w})$ — the conditional probability of a dependency feature given a word, from matrix **W**, we are able to infer $p(\mathbf{w}|\mathbf{z})$ – the conditional probability of a word given a latent dimension, and from matrix **H**, we are able to infer $p(\mathbf{d}|\mathbf{z})$ – the conditional probability of a context given a latent dimension. Using Bayes' rule, we can equally compute the inverse conditional probabilities.

3.2. Computing meaning shifts

Of course, having simply a representation of the use of the word is not enough to yield an adequate theory of meaning. One has to say how these word meanings combine to make meanings of more complex constituents, and how both primitive and derived meanings support inference. Using adjective-noun composition as an example, we present a distributional framework that is able to compute the required representations.

In order to compute the meaning shifts that take place for adjectives and nouns when they appear in a compositional expression, we make use of the *latent vector weighting* approach of [29]. The main idea is that the adjective (noun) that appears with a particular noun (adjective) defines

a distribution over latent semantic factors, which is subsequently used to adapt the general vector representation of the noun (adjective), shifting the vector towards the correct meaning. As a first step, a factorization model is constructed in which words, together with their window-based context words and their dependency features, are linked to latent dimensions. The factorization model then allows us to determine which dimensions are important for a particular expression, and adapt the dependency-based feature vector of the word accordingly.

Using the results of our model, we can now adapt a word's feature vector according to the compositional expression it appears in. Intuitively, a modifier that takes part in a compositional expression with the target word (e.g. an adjectival modifier that appears with a target noun) pinpoints the important, relevant semantic dimensions of the target word, creating a probability distribution over latent factors,

$$p(\mathbf{z}|d_j) \tag{5}$$

where d_j is the dependency feature that represents the target word's modifier in the compositional expression.

The resulting probability distribution over latent factors can be interpreted as a semantic fingerprint according to which the target word needs to be interpreted. Using this fingerprint, we can now determine a new probability distribution over dependency features given the context.

$$p(\mathbf{d}|d_j) = p(\mathbf{z}|d_j)p(\mathbf{d}|\mathbf{z}) \tag{6}$$

The last step is to weight the original probability vector of the word according to the probability vector of the dependency features given the word's context, by taking the pointwise multiplication of probability vectors $p(\mathbf{d}|w_i)$ and $p(\mathbf{d}|d_j)$.

$$p(\mathbf{d}|w_i, d_j) \sim p(\mathbf{d}|w_i) \cdot p(\mathbf{d}|d_j) \tag{7}$$

This final step is a crucial one in our approach. We do not just build a model based on latent factors, but we use the latent factors to determine which of the features in the original word vector are the salient ones given a particular context. This allows us to compute an accurate adaptation of the original word vector in context. For more information on the model, as well as implementational details, we refer the interested reader to [29].

3.3. Example

Let us exemplify the procedure with an example. Say we want to compute the distributionally similar words to the noun *device* in the context of example expressions 8 and 9.

(8) explosive device

(9) electrical device

First, we determine the contextual dependency feature for both instances, in this case $d_1 = \{explosive_A\}$ for example 8, and $d_2 = \{electrical_A\}$ for example 9, and we extract their respective probability distributions over latent factors — $p(\mathbf{z}|d_1)$ and $p(\mathbf{z}|d_2)$. Using these probability distributions over latent factors, we can now determine the probability of each possible dependency feature given the particular dependency feature used in the expressions – $p(\mathbf{d}|d_1)$ and $p(\mathbf{d}|d_2)$.

The former step yields a general probability distribution over dependency features that tells us how likely a dependency feature is given the argument that our target word appears with. Our last step is now to weight the original probability vector of the target word (the aggregate of dependency-based context features over all contexts of the target word) according to the new distribution given the argument that the target word appears with. For the first sentence, features associated with the argument *explosive* (or more specifically, the dependency features associated with latent factors that are related to the feature $\{explosive_A\}$) will be emphasized, while features associated with unrelated latent factors are leveled out. For the second sentence, features that are associated with the adjective *electrical* (dependency features associated with latent factors that are related to the feature $\{electrical_A\}$) are emphasized, while unrelated features are played down.

We can now return to our original matrix **A** and compute the top similar words for the two adapted vectors of *device* given the different arguments, which yields the results presented below.

1. **device**$_N$, d_1: *device*, *ammunition*, *firearm*, *weapon*, *missile*
2. **device**$_N$, d_2: *device*, *equipment*, *sensor*, *system*, *technology*

3.4. Exploring co-composition

Using the method outlined above, we evaluated how DS composition methods might reflect the shift that the adjective and noun undergo during co-composition and how we might translate this result back. To do so, we compare the vectors for the unmodified adjective and unmodified noun with the vectors for the adjective and noun weighted according to their composition.

Consider, e.g., the shift of the adjective *heavy* when it modifies the noun *traffic*. The first listing gives the 10 most similar adjectives for the unmodified adjective *heavy*. The second listing shows the 10 most similar adjectives relative for *heavy* modified by *traffic*, as computed by our method.

1. **heavy**$_A$: *heavy*$_A$ (1.000), *torrential*$_A$ (.149), *light*$_A$ (.140), *thick*$_A$ (.127), *massive*$_A$ (.118), *excessive*$_A$ (.115), *soft*$_A$ (.107), *large*$_A$ (.107), *huge*$_A$ (.104), *big*$_A$ (.103)

2. **heavy**$_A$, traffic$_N$: *heavy*$_A$ (.293), *motorised*$_A$ (.231), *vehicular*$_A$ (.229), *peak*$_A$ (.181), *one-way*$_A$ (.181), *horse-drawn*$_A$ (.175), *fast-moving*$_A$ (.164), *articulated*$_A$ (.158), *calming*$_A$ (.156), *horrendous*$_A$ (.146)

There is an evident shift in the composed meaning of *heavy* relative to its original meaning in our example. We observe a shift in the quantitative measure of cosine similarity; for instance, sim_{cos} between the original vector for *heavy* $\vec{v}_0$ and the modified vector for *heavy* $\vec{v}_1$ as modified by its predicational context: $sim_{cos}(\vec{v}_0, \vec{v}_1) = .29$. We also see it in the fact that the set of most similar words with respect to sim_{cos} shifts to a set that is disjoint for the set of words most similar to the unmodified adjective. The composition model is attuned to the fact that the modifier has to apply in some way to vehicles. We ran the method on over 200 adjective-noun combinations and replicated similar findings [4]. Overall, $sim_{cos}(\vec{v}_0, \vec{v}_1)$ for all the adjectives varied between $.25$ and $.77$ with the vast majority of adjectives exhibiting $sim_{cos}(\vec{v}_0, \vec{v}_1) < .5$. The mean overlap between the shifted set of similar words and the original set of 20 most similar words is 6, whereas for nouns it's 10. $sim_{cos}(\vec{v}_0, \vec{v}_1)$ for all the nouns varied between .3 and .8, and the mean was .5. These numbers indicate that co-composition was the norm for adjectives in our test set. Nouns shifted less, something that we would expect from the fact that the vast majority of adjectives in most languages are subsective.

4. From coercions to co-composition in type theory

Linguists working on word meaning developed a rich and complex typology of different sorts of entities that could affect semantic composition. Predicates come with *sortal restrictions* on their arguments; and if arguments meet those restrictions, then the predication is semantically felicitous. If they do not, then often the predications fail, as in (10):

(10) A prime number is soft.

The predicate *is soft* cannot felicitously apply to its argument *a prime number*, unless one of the terms is redefined or acquires a very idiosyncratic meaning in context. A natural explanatory hypothesis for the behavior of selectional restrictions is that selectional restrictions are *type* restrictions; the type of entity that satisfies the predicate *is soft* is a subtype of the type of entities in formal

semantics (the type E), and this subtype is incompatible with the type assigned to numbers.[4] Work in foundations of mathematics and computer science uses strongly typed languages and a system of type checking or consistent type assignments to terms to assess the well-formedness of formulas or programs. [1] uses these tools in an analysis of semantic well-formedness, selectional restrictions and coercion, which we will follow here. We note that work in MTT [17, 18, 19] also uses types in dealing with coercion.

Let's begin with a very productive sort of coercion. In the following examples, we see two types of entities that a single expression may give rise to.

(11) a. John brought a bottle. It had a nice label/It was yummy.

b. John brought a bottle. It had a nice label and was yummy.

c. John touched the bottle, which had been so yummy.

(11a,b) provide two different continuations for the first sentence, each containing pronouns that refer back to two different sorts of objects. If we analyze pronominal reference across sentences using Discourse Representation Theory or some other dynamic semantic formalism, we see something interesting. Depending on the continuation, one could infer that the first sentence of (11) makes available a discourse referent for the bottle or one for its contents that can be linked to the anaphoric pronoun in the continuation; but as (11bc) show, the first sentence makes available discourse referents for *both* the bottle and its contents.

Let's now examine the examples in (11) from the perspective of selectional restrictions. *Bottle* intuitively types the entities satisfying the predicate as being of a type having to do with containers. Predicates like *have a nice label* apply to physical objects with stable surfaces, *inter alia* containers. So in the first continuation given in (11a), the pronoun picks up the discourse referent of the type CONTAINER. Assuming that anaphoric binding preserves type identity, then the discourse referent introduced by *it* that is the argument of *have a nice label* is of the right type to meet the selectional restrictions of the predicate. However, in the second continuation, the predicate *is yummy* requires its argument to be edible or drinkable, let's assume. For simplicity, let's assume that BOTTLE is not in the type system a subtype of comestible foodstuff (though this is a simplification—the bottle might be made of chocolate). In that case, we have a case of coercion: in order for the predication to succeed, the predication must license the introduction of a discourse referent or variable that refers to the contents of the bottle and it is this referent that is the argument of the predicate *is yummy*. This discourse referent with the type CONTENTS satisfies the selectional restrictions of the predicate. Importantly, with this kind of coercion both the "coerced" denotation and the original denotation of the argument seem equally available.

There are other well-known examples of coercion—for example those involving aspectual verbs like *start*, *begin* and *finish*, as well as verbs like *enjoy* in English, where this is not the case. For instance, (12) is equivalent in meaning to (13):

[4] [1] argues for this thesis in detail.

(12) Julie enjoyed (started/finished) a book.

(13) Julie enjoyed doing something with (e.g., reading, writing, ...) a book.

enjoy requires an event as its direct object as in *enjoy the spectacle*, *enjoy the view*. This also happens when *enjoy* takes a free relative clause as its complement, as in *enjoy (hearing) what he said*. When the direct object of a transitive use of *enjoy* in a predication does not denote an event, the predication introduces some sort of eventuality that can serve to satisfy the selectional restrictions of the verb.

We must do several things to apply type coercions to examples like (11) or (12). First, we must extend the standard type system used in Montague Grammar, which includes only two atomic types E (the type of entity) and T (the type of truth values), "downwards" to include appropriate subtypes of E. We have to have subtypes like CONTAINER and CONTENTS and PHYSICAL OBJECT and EVENTUALITY as inputs to the type shifting mechanism, whatever form this may take. Second, if we take, as we do, coercions to be licensed by conflicts between the requirements of predicates like *be yummy* or *have a nice label*, we have to assign these predicates types or lexical meanings such that they require one of these types as arguments.

A difficulty for spelling out a story of coercion detailed at length in [1, 3, 2] is to account properly for the "shift in meaning" that people feel goes on in coercion. Simply shifting the meaning of the predicate or of its argument does not do justice to the data. TCL [1] but also [13, 11] argue that you have to change the relation of predication that holds between the predicate and the argument. A clash between the type of the argument and the type presupposition of the predicate induces, not a type shift in either the argument or the predicate, but rather a type shift on the predication relation itself. This shift has effects in spelling out the logical form or associated formula: in TCL and within the framework of subsumptive subtyping, the shift introduces a functor that takes the core part of the verb or predicate meaning and returns a formula in which an existentially bound fresh variable with a type appropriate to the selectional restrictions of the predicate is now the argument of the predicate and is related in some way to the variable introduced by the predicate's syntactically given argument. To make matters more concrete, suppose that a verb like enjoy has the form

$$\lambda\Phi\lambda\Psi\Psi(\lambda x\Phi(\lambda y\text{enjoy}(x,y))) \tag{14}$$

where Φ and Ψ range over noun phrases. Now suppose that the type EVENT assigned to y by *enjoy* is incompatible with that introduced by Φ (say P (physical object) on its referential argument (the argument that is fed to the verb). Let P be a variable over $\lambda y\text{enjoy}(x,y)$ in (14). Then, by abstraction (14) is equivalent to:

$$\lambda P\lambda\Phi\lambda\Psi\Psi(\lambda x\Phi(P))[\lambda y\text{enjoy}(x,y)] \tag{15}$$

Given that *enjoy* allows a type shift P $\rightarrow$ EVT, the shift is reflected in logical form by applying the following functor to the term λyenjoy(x, y):

$$\mathfrak{f}\colon \lambda y \text{ enjoy}(x, y) \rightarrow \lambda u \exists v\ (R(v, u) \wedge \lambda y\text{enjoy}(x, y)[v]) \tag{16}$$

where v: EVT and u: P. Now composition can proceed as the type mismatch has been cleared up. Further, the meaning of the argument does not shift—it remains what it always was; the predicate also retains its original meaning. What changes is the "glue" that links them together.

How can we use the techniques introduced to deal with coercion for co-composition? Proponents of distributional semantics often use the metaphor that the meaning of an individual word is a cloud of possible specifications that depends for its precisification on a determinate predicational and discourse context. TCL's type-theoretic approach is also amenable to this view by making use of underspecification in the type language [1]. Another way to make this metaphor more precise type-theoretically is to construe a basic fine-grained type as a disjunctive type of its possible precisifications. But with no constraints on what these disjoined types might be or what specifications of an underspecified type are, we can't predict the important difference between the two unrelated senses of *suit* in our examples (1a,b) and the related but shifted meanings, say in the examples in (2).

The functor approach of TCL to coercion provides a more promising model for co-composition. TCL already explains coercions involving adjectives and nouns as in *quick cigarette* along the lines of verbal coercions, so it appears sensible to try to extend this approach. We illustrate with adjective-noun combinations, in which composition has this general form, where M is the adjective, or more generally the modifier, and N is the noun:

$$\lambda x\ (\mathcal{O}_M(N(x)) \wedge \mathcal{O}_N(M(x))) \tag{17}$$

The functors $\mathcal{O}_M$ and $\mathcal{O}_N$, like our functor $\mathfrak{f}$, can encode observed shifts of meanings of the noun and modifier in adjective-noun combinations. Co-compositions, however, do not involve a repair of a type mismatch, so we shouldn't expect that these operators will, when spelled out, necessarily introduce new variables. So what causes the shifts?

To understand this form of meaning combination better, we have to specify the functor $\mathcal{O}_M$, and the one that gives rise to $\mathcal{O}_N$. For subsective adjectives, $\mathcal{O}_M$ must give rise to a functor whose denotation is at least a subset of the common noun phrase N. But beyond that formal linguistics doesn't tell us much. Further, how do non-subsective adjectives behave? For temporal non-subsective adjectives like *former, future* and modal adjectives like *possible, likely*, it's pretty clear what $\mathcal{O}_M$ should be. It will be a temporal or modal operator.[5]

[5]Some cases are much less obvious, however. Consider *capital* vs. *cultural capital*. *Cultural* does not seem to be subsective, but it gives some sort of modal frame—within the cultural worlds, X is the capital of the country/world...

For material modifiers like *stone, brass, clay* that at least sometimes are non-subsective, the answer is less obvious. These can function subsectively but also non-subsectively; it depends on what the head noun is that they combine with. Brass pots are pots, and stone pots are pots too. But a stone lion can't be a lion (it's not a living animal). In fact, it is essentially or necessarily not a lion. And here there may be a sort of coercion story, for the shift in meaning on the noun occurs when there is a type conflict. This is borne out by preliminary experiments over a large corpus using vector models for meanings like those found in[21]. [7] calculated a vector for *lion en pierre* using the matrix approach to adjectives advocated by [5] over a parsed version of the French FrWak corpus; using the standard cosine distance measure of similarity for vectors, the closest vectors were those of expressions like *statue*, *sculpture*, *stone*, while the vector for *lion* had a much larger cosine distance from *stone lion*. However, a stone lion looks much more like a lion than any of the alternatives that *lion* suggests—these associated types are common hyponyms of LION's lowest common super type [1]. Following the analysis of [1], we have the following entry for a generic material modifier *Mat*:

$$\lambda P\ \lambda x \lambda \pi\ (P(\pi * \text{MADE OF}(\text{HD}(\text{MAT}))(x) \wedge \exists u(\text{Mat}(u) \wedge \text{made-of}(u, x))) \tag{18}$$

To understand this formula, we need to explain several things. [1] uses a dynamic parameter to pass type requirements from predicates to their arguments; π is a sequence of type requirements, where sequences of types are built up using the concatenation operation $*$, and $\pi * \text{MADE OF}(\text{HD}(\text{MAT})$ above instructs P that its argument whose specific type must satisfy the type that it is made of the most specific type associated with *Mat*. Since the λ abstracted variable P ranges over first order properties, (18) can combine with the entry for a noun in TCL or with a simple predicate like $\lambda u\ lion(u)$ to yield a λ term for an adjective-noun combination. [1] argues that in case the head type of the noun, $\text{HD}(N)$, cannot have as a material that given by the modifier, then the predication has to be understood loosely. Although a stone lion isn't a real lion, it shares many more features with lions than it does with other animals (shape, typical posture and so on). That is, a stone lion is so called because it is more a lion than the other alternatives evoked by the expression.

Following [2], we take a different approach here and code the notion of loose speaking into a particular sort of functor with a well-defined semantics. The reason is that although loose talk is a fact of life, it is difficult to integrate it properly into formal semantic theories. Although the details get a bit complicated, in principle we can specify O_M when the types $\text{MAT} \sqcap \text{HD}(P) = \bot$. The type combination specifies the operator O_M to something like *it appears that* or *resembles* as in (19), since loose talk typically makes use of superficial properties that serve to infer that something is of a particular type. Leaving aside the type parameters π for ease of readability, we spell this out in (20). The formula $(Gen\ z(P(z), Q(z))$ says that normal Ps normally have the property Q. We could further restrict X to explicitly compare the properties that our target shares with respect to lions compared to the properties it shares with the other alternatives:

$$O_{\text{MADE OF(MAT)}} := \lambda P \lambda x\ \text{looks-like}(P(x)) \tag{19}$$

$$O_{\text{MADE OF(MAT)}} := \lambda P \lambda x\ \text{Most} Q\ (X(Q) \rightarrow\ (Gen\ z(P(z), Q(z)) \rightarrow Q(x))) \tag{20}$$

When applied to a common noun phrase like *lion*, this functor takes $\lambda x\text{lion}(x)$ and transforms it into the property of things that have most of the superficial properties that lions typically or normally have. This is in effect a kind of coercion, though not one that would fit under traditional analyses of coercion.[6] The restriction on properties X restricts the property quantification to range over those superficial properties that lions or the alternatives have. For material modifiers, on the other hand, the functor $\mathcal{O}_N$ from schema (17) is just the identity functor: stone lions really are stone, or made of stone, as are stone jars, stone houses, etc. This follows from theses about essential materialism as detailed in [1]. Putting these observations together and exploiting the schema (17), we now have the form for the predication involving a material modifier and a noun like *stone lion* with our specifications for O_M in (19) and $\mathcal{O}_N$ as the identity functor:

$$\lambda x\ \text{Most}Q\ (X(Q) \rightarrow\ ((Gen\ z(\text{lion}(z), Q(z)) \rightarrow Q(x)) \wedge \exists u(\text{stone}(u) \wedge \text{made-of}(u, x)))) \quad (21)$$

Of course, the case of material modification that we have examined here is in fact a kind of coercion. The type of matter given by the material modifier is essentially incompatible with the type of the noun. So let's now consider the question of the determination of the functor $\mathcal{O}_N$ for subsective adjectives, where we'll simplify and assume the functor $\mathcal{O}_M$ is the identity functor. Let's consider once again the intersective adjective like *flat*. A flat country is closer to a mathematically flat surface than one that is not, at the scale relevant for measuring the surface of countries. A flat tire has a portion that is closer to a flat surface than any portion of a normally inflated tire. Flat water and flat beer have still unperturbed surfaces that are flat by comparison to bubbly ones. In each one of these cases, the way the adjective is applied changes slightly in virtue of the type of the head noun, but there is also always a common reference to a standard mathematical notion of flatness (0 curvature). Unlike the cases of coercion, the change in the meaning of *flat* depending on what type of object it applies to doesn't arise from a type conflict: *flat* applies to physical objects; and tires, countries and beer are all physical objects all equally happily.

Let's now see how to apply our schema (17). As *flat* is subsective, we will assume that $\mathcal{O}_M$ is the identify functor and concentrate on $\mathcal{O}_N$. $\mathcal{O}_N$ is an operator that takes the head type of the modified noun as a parameter and provides on that basis an adjustment to logical form. Given the form of a TCL entry for a common noun, the type information from the noun is passed to the modifier, so implementing such an operator in TCL is straightforward. Here is the entry for a noun N whose type presuppositions on its head variable is A and where P ranges over adjectival denotations (simple predicates):

$$\lambda P\lambda x\lambda\pi\ (\mathcal{O}_N(P(x, \pi * P\colon \textsc{applies-to}(\textsc{hd}(N))) * x\colon \textsc{a}) \wedge N(x, \pi)) \quad (22)$$

In particular, let's look at the entry for *beer* to illustrate the approach:

$$\lambda P\lambda x\lambda\pi\ (\mathcal{O}_{\text{BEER}}(P(x, \pi * P\colon \textsc{applies-to}(\text{BEER}) * x\colon \textsc{liquid} \sqcap \textsc{mass})) \wedge N(x, \pi)) \quad (23)$$

[6] [23] also advocates something like this view, though the details are quite different from those of the approach developed here.

Let M and L stand for the types MASS and LIQUID. Now let's look at the entry for *flat*, whose type presupposition is just that the object it applies to must have at least three dimensions:

$$\lambda x \lambda \pi \ (\mathit{flat}(x, \pi * x \colon \text{3-D})) \tag{24}$$

Combining the λ terms in (24) and (23) yields:

$$\lambda x \lambda \pi \ (\mathcal{O}_{\text{beer}}(\mathit{flat}(x, \pi * \text{FLAT} \colon \text{APPLIES-TO}(\text{BEER}) * x \colon \text{L} \sqcap \text{M} * x \colon \text{3-D})) \wedge \text{beer}(x, \pi)) \tag{25}$$

The functor exploits the most specific type associated with the head noun to the adjective—in TCL type information is passed from the head to its arguments and so this fits with TCL's general architecture.

Now what is the content of $\mathcal{O}_{\text{BEER}}$ when applied to *flat*? Here's our attempt.

$$\begin{aligned} \lambda x \lambda \pi \ (\exists y(\textit{surface}(y, x, \pi * x \colon \text{3-D}) \wedge \\ \forall z, w((\textit{beer}(z, \pi * x \colon \text{3-D}) \wedge \text{surface}(w, z, \pi * x \colon \text{3-D}) \wedge \\ \textit{has-bubbles}(z, \pi * x \colon \text{3-D})) \rightarrow \text{flatter}(y, w)) \wedge \textit{beer}(x, \pi * x \colon \text{3-D})) \end{aligned} \tag{26}$$

To give the account some more meat, we must impose some more constraints on what functors can be. Formal semantics tells us that non-subsective modifiers will affect in some way the meaning of the noun and will often be cases of coercion, but that otherwise the functor $\mathcal{O}_M(N)(x)$ should should entail $N(x)$. But we don't have any constraints on the functor $\mathcal{O}_N$. How are the functors in the TCL model constrained by types? You can't just shift a fine-grained type anywhere in type space!

We also have to deal with what we call the *triggering puzzle* for co-composition: why do type adjustments take place in modifier-noun combinations when there is no incompatibility between the type presupposition of the predicate and the type of its argument as in the case *flat beer* or *explosive device* versus *electrical device*? What triggers these shifts? It appears that modifiers have a contextually sensitive meaning that depends on the fine-grained type information of what they modify. This shows that we cannot just understand meaning shifts as coercions forced by incompatibilities between the type presuppositions imposed on an argument by a predicate and the type presuppositions of the argument itself. Semantic composition allows for the introduction of new meaning even without shifts in type that are relevant for the selectional restrictions. Furthermore, this is not an aspect of modification that is restricted to a few adjectives, as our empirical studies show.[7]

[7] For more discussion see [7, 8].

5. Types revisited

To solve the triggering puzzle we posed at the end of the previous section, we need to revisit the notion of a type. We first quickly review how type theories conceive of types, and we then pass to a distributional point of view.

5.1. Symbolic types

All type theories countenance a set of "simple" or "basic" types. What are simple types? Types are typically taken as primitives in type theories, and each lexical entry has a unique type. Furthermore, the functional relation between a lexical entry w and its type is "hard"; either w has type T_w or it doesn't. There are differences between theories, however. TCL for example countenances a structured set of basic types (a poset conferred by the subtyping relation $\sqsubseteq$), but assigns to common nouns a type that is aligned with their syntactic category as an incomplete noun phrase and with their denotational semantics as a set of objects in a model. That means that nouns must have a functional type, which determiners can then saturate; this is reflected in the logical translation of a common noun like *table*: λx table(x), with x bearing an atomic type. In fact, TCL assigns *two* types to every common noun, one its type presupposition, which is quite general and which is used to check well-formedness in composition, and then a fine-grained type, which is used in the co-composition story above. So in the entry for *table*, the lambda bound variable x would have the type presupposition of being a physical object, and a fine-grained type TABLE, where TABLE $\sqsubseteq$ P, where P is the type of physical objects. The type presupposition of *table* then would be P $\rightarrow$ PROP (where PROP is the type of propositions), and this would have to satisfy the selectional restrictions of what the noun is an argument to, say, a determiner.

On the other hand, it doesn't make sense to use the fine grained type to type the predicate in TCL. Were we to do this, then a predicate like *table*, which would have the functional type TABLE $\rightarrow$ PROP, couldn't even apply to something that wasn't of type TABLE; we couldn't form a sensible proposition that that thing over there is a table, if it were not a table; we could only form the absurd proposition $\bot$. There would be no false propositions predicted by the system or only one, the proposition $\bot$. Outside of the world of mathematics and necessary truths—and even there the idea that there is no meaningful distinction between false propositions is difficult to swallow—the conclusion is preposterous. Were TCL to make such a prediction, it would have to be rejected.

In MTT common nouns (CNs) are assigned just one, primitive atomic type rather than a functional type, which would seem a nice simplification. One is better than two on grounds of conceptual, Ockhamist economy, if you can get away with it.[8] For instance, common nouns like *table* and *man* are interpreted as types TABLE and MAN, rather than as functional types. They don't carry type

[8]For a fuller discussion of MTTs view of common nouns as types, see Stergios Chatzikyriakidis and Zhaohui Luo, On the interpretation of common nouns: types vs. predicates, chapter 2 of this volume.

presuppositions, as they would in TCL. In MTT, each common noun has the most specific atomic type associated with the noun in TCL as its principal semantic contribution; no distinction is made between type presuppositions and "proferred" content, which is also type-theoretic in MTT. In other words, common nouns function like proper names in having a simple entity as semantic value. This difference allows MTT approaches to avoid certain complexities of TCL like the use of continuations to pass type information in composition. On the other hand, this means MTT does not distinguish between semantic anomaly and falsity, whereas this is a core concern of TCL, and it is the main motivation for distinguishing between type presuppositions and fine-grained types.

The decision to make the type of nouns simple and specific has other ramifications as well. The decision to treat noun as simple types means that modifiers of nouns should now be functions from types into types, or a dependent type, if we want a uniform treatment of the first arguments of determiners. However, MTT takes adjectives to predicates from types to propositions to handle simple predications like *John is handsome*. 'Handsome' is interpreted as a predicate of type MAN $\rightarrow$ PROP. But if that's the right analysis for adjectives, then determiners wouldn't bind any argument in a modified noun phrase.

A further consequence is that determiners become in many ways ambiguous; roughly we need one determiner for each type of NP, reminding us of Russell's theory of typical ambiguity. Determiners don't have a determinate type as in Montague Grammar but rather a family of types. Not distinguishing between type presuppositions and most specific "proferred" types of common nous would have the undesirable consequence that predicates like verb phrases that take types as arguments also will have quite particular types. So the family of types of determiners would have to include one type of determiner for each distinct pair of atomic and predicate types.

The decision to use only fine-grained types has empirical consequences not just at the conceptual level but at levels of predications. A predicate of humans will not have the same type as a predicate of cats, for example, leading to puzzles as to why run of the mill co-predications like

(27) a. John and his cat are both well taken care of.
b. I like Mary and her dog.
c. The vehicle struck another car and then a pedestrian before coming to a stop.

should be well-formed and true, where *John* is typed as HUMAN. This would require a massive amount of type shifting or some other mechanism to get *John* and *his cat* to have the same type so as to be able to combine with the VP. Distinguishing between type presuppositions or presupposed types needed to check for well-formedness, and fine-grained "proferred" types eliminates the need for such a mechanism.

Nevertheless, MTT is right to point out the need for fine-grained types. We can't make sense of co-composition without them. In TCL as in MTT, however, these fine-grained types don't have

much structure, and almost all theories are mute as to what content such atomic types convey, though [1] makes some informal remarks as to how such types should contain rules justifying the use of expressions with a particular fine-grained type. But without any content assigned to fine-grained types it's difficult to impose any substantive constraints on TCL functors, which we need to get an explanatory account. A similar difficulty would affect MTT's approach in that without assigning a content to atomic types, we don't know how to impose any substantive constraints on MTT's notion of coercive subtyping, which would have to be used for type shifting and adjusting meanings in a particular predicational context.

Can we take anything from the algebraic structure of DS meanings to make the type-theoretic account of TCL more concrete? We believe so. We will hypothesize each basic type to be a vector in some space V whose dimensions are syntactic/semantic constructs or something more semantic and abstract like the latent dimensions that come from factorization techniques like non-negative matrix factorization. More abstractly, a type is any distribution of real numbers over those dimensions. Types come with a subtype relation organizing the structure into a poset. We can now explore relations between types (or distributions over context dimensions) in a variety of ways;[15], for instance, has made some interesting proposals for encoding the subtyping relation.

5.2. Type distinctions from distributional semantics

With this new conception of types, how do we think of more general types, above the word level? The preliminary studies presented in this section look at general types from a distributional point of view, which type theorists have so far not explored. On the one hand, we explore a clustering algorithm like *k-means clustering*, which partitions finer grained types into sets of types. On the other hand, we make use of factorization techniques in order to extract latent characteristics from raw frequency data.

k-means clustering k-means clustering is a well-known clustering algorithm, which aims to group n instances (represented by d-dimensional vectors) into k groups, such that each instance belongs to the cluster with the nearest mean (in terms of a pre-defined similarity metric, e.g. cosine similarity). The algorithm consists of two steps. The first step assigns instances to the nearest cluster mean (centroid), while the second step calculates the new centroids based on the new cluster members assigned in the first step. The algorithm iterates between the two steps until it converges to a local optimum. There are no guarantees to find the global optimum, but in practice the algorithm converges to a stable result when using large-scale language data. Note that k-means is what is called a *hard* clustering algorithm: each instance is exclusively assigned to a single cluster.

We applied k-means clustering to a set of 5000 nouns, where for each of the 5000 nouns we note how frequently it appears with a particular dependency feature (80 000 in total). The bare

frequencies were weighted using a technique called *pointwise mutual information* (PMI), in order to give more weight to co-occurrences that are more surprising. The 5000 nouns (each a 80 000-valued vector) were then grouped into k groups. We tried out clustering sizes of $k = 2, 3, 5, 10, 20$, in order to investigate what kind of generalizations the algorithm would come up with when using limited cluster sizes.

When inspecting the results, we found the algorithm was able to come up with some interesting generalizations. For $k = 2$, the algorithm made a very clear division between animate and inanimate nouns. For $k = 3$, the algorithm found groups for animate, concrete, and abstract nouns. When we move to $k = 10$, we notice the algorithm tends to come up with finer characterizations, yielding clusters for locations and substances, among others. The same tendencies show up for the other clustering results of $k = 5$ and $k = 10$. To give an idea of the results, we show the first 10 words (alphabetically ordered) for $k = 2$ and $k = 3$, and a selection of clusters for $k = 10$.

k=2

- abbey, abbreviation, ability, abnormality, abolition, abortion, absence, absorption, abstraction, abstract, ...
- academic, accountant, activist, actor, actress, addict, adjudicator, administrator, adult, advertiser, ...

k=3

- ability, abnormality, abolition, abortion, absence, absorption, abstraction, abstract, abundance, abuse, ...
- academic, accountant, acquaintance, activist, actor, actress, addict, adjudicator, administrator, adult, ...
- abbey, abbreviation, academia, academy, accent, accessory, accommodation, accompaniment, accordance, ace, ...

k=10

- accessory, adapter, aircraft, air, alarm, alloy, aluminium, ambulance, ammunition, amplifier, ...
- abbey, accommodation, acre, airfield, airport, aisle, allotment, altar, amenity, apartment, ...
- acid, aggregate, alcohol, ale, antibiotic, antibody, antigen, apple, asbestos, ash, ...
- ...

These results seem to indicate that general type distinctions may very well be characterized by distributional data. By using the appropriate algorithms in order to make generalizations over a very large co-occurrence space, we are able to come up with type characterizations at different levels of granularity.

Factorization techniques Factorization techniques give us another means for exploring more general types in the types as vectors view. First of all, we explore factorization techniques in order to automatically induce a mass-count continuum from distributional data. The mass-count property of nouns is well known to correlate with distributional data in the linguistic literature. In particular, certain determiners select for count nouns (for example *each*) while others select for mass nouns (e.g. *much*). Two-way co-occurrence frequencies of 18 determiners and 2000 nouns are automatically extracted from a larger corpus, and the resulting matrix was factorized using singular value decomposition.[9] The two most important dimensions that come out of the decomposition are presented in figures 2 and 3, for determiners and nouns respectively. What emerges from the figures, is that the two most important dimensions seem to correlate with a degree of massness, and a degree of plurality.

In figure 2, a count-mass continuum seems to emerge, with countable determiners (*two*, *three*) appearing on the left-hand side, and mass determiners (*much*) on the right-hand side. To a lesser degree, singular determiners seem to appear towards to bottom of the graph, while plural determiners appear towards the top. A similar tendency is present for the noun space in figure 3. The results seem to show a promising start for automatically finding general types (or type presuppositions) on the basis of distributional generalizations involving simple types.

Secondly, in order to explore a more fine-grained type continuum, we again make use of non-negative matrix factorization, as outlined in section 3.1. As input to the NMF algorithm, we used exactly the same data as for the k-means clustering: a matrix of 5000 nouns cross-classified by 80 000 dependency relations and weighted using PMI. We defined our limited number of dimensions $k = 2, 3, 5, 10, 20$ similar to the number of clusters we used in our k-means clustering algorithm.

When inspecting the results, we notice a similar tendency emerges as we saw with k-means: when using a limited number of dimensions, the algorithm comes up with broad generalizations (animacy, concreteness, ...), whereas a larger number of dimensions yields more refined distinctions. We are showing the results for $k = 3$, and part of the results for $k = 10$. Note that this time, we are showing the top words for each dimension, i.e. the words with the highest value for the dimension in question.

k=3
- idea, use, issue, study, result, form, change, value, matter, policy, ...
- man, player, people, friend, member, woman, club, team, company, boy, ...
- water, plant, food, surface, land, wood, oil, tree, cell, air, ...

k=10
- cost, benefit, use, number, amount, rate, requirement, value, type, provision, ...
- worker, member, people, officer, individual, teacher, student, staff, manager, parent, ...

[9] We used singular value decomposition because of its guarantee to provide the best possible fit for the original data given a limited number of dimensions. The top dimensions are thus guaranteed to explain the most variance present in the original data.

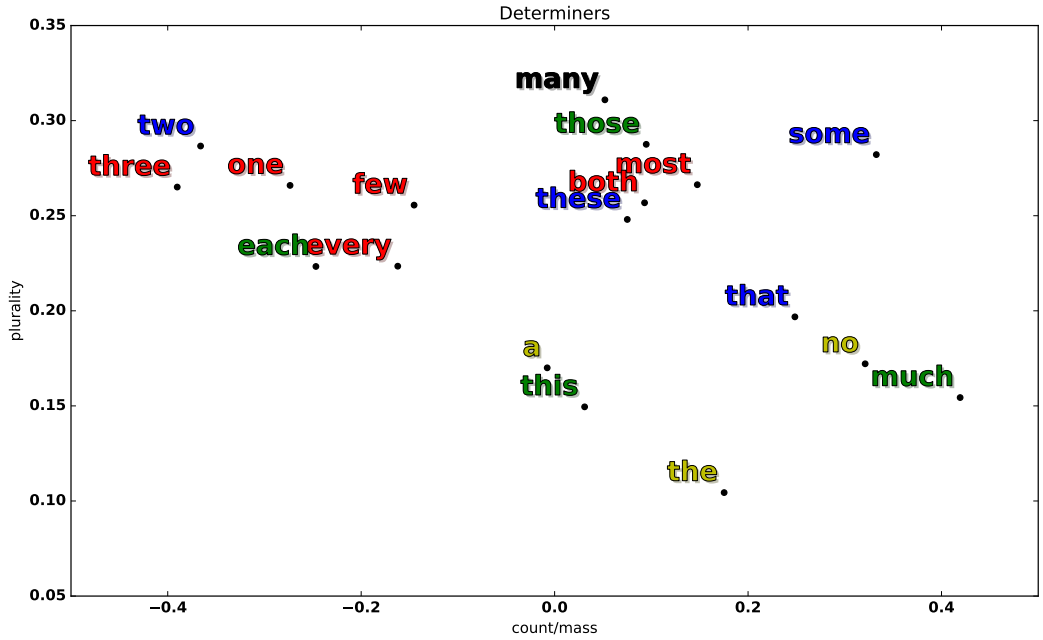

Figure 2: Determiners in reduced SVD space of two dimensions

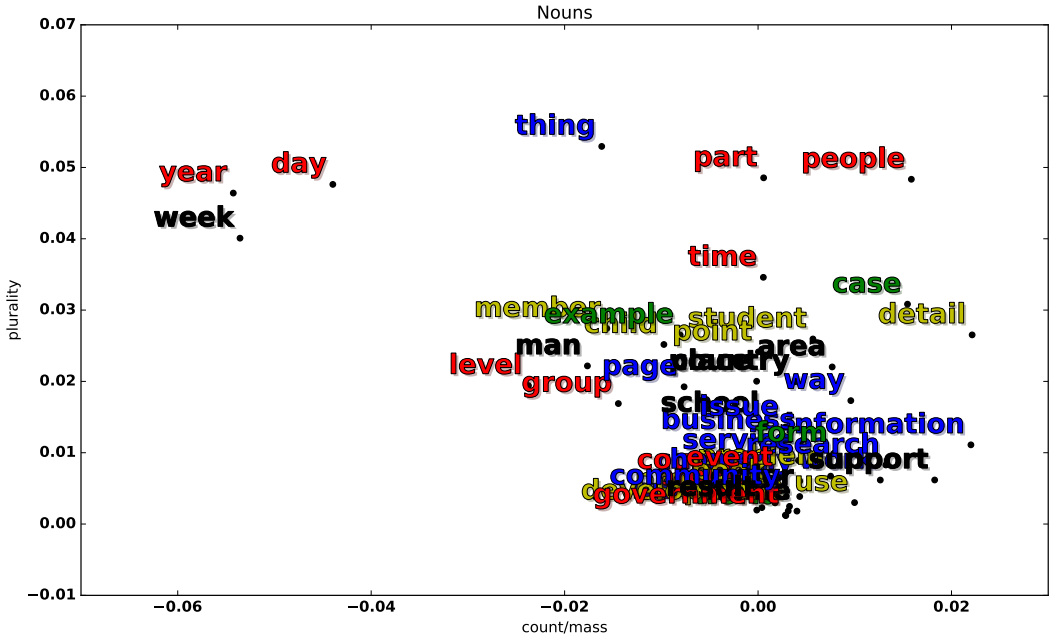

Figure 3: Nouns in reduced SVD space of two dimensions

- theory, principle, concept, understanding, analysis, interpretation, study, knowledge, philosophy, aspect, ...
- day, week, time, night, year, month, meeting, minute, event, ...
- ...

Note that the results presented in this section constitute an intuitive exploration, and the interpretation of the results relies entirely on the authors' judgement. Still, the examples, which have not been cherry-picked in any way, indicate that the general type tendencies described are genuinely present in the data. Notice for instance that on the $k = 10$ reduction, eventualities, which are a general type and a type presupposition of the object argument for aspectual verbs and verbs like *enjoy* are grouped in with temporal entities, while we have as yet ill understood divisions between types of informational or abstract objects (dimensions 1 and 3). We plan to study these divisions further.

More importantly, however, what emerges from our hypothesis that types are vectors in a space of contexts is that types are very much context dependent. Each word has a type, but each word in a more specified context has a slightly different type. If you fill in the predicational context, you will get a (slightly) different type from the type for the word in isolation. There's another dimension of variation as well. Change the corpus and you get a different set of types, or a different type for each word. In fact the space of basic types is continuous in principle with every vector in that vector space constituting a type; but only a very few are lexically realized as simple words in a given corpus or overall by any language. A few more but (still constituting a set of measure 0 since there will be an most countably many and the total number of vectors is uncountable) will correspond to words modified by surrounding contexts. According to the NMF view of types, while basic types may cluster together to suggest higher types upon dimension reduction, the size of the dimension reduction will determine the set of more general types one gets. What higher types one wants will depend on one's aims and purposes, though we take comfort from the fact that the general divisions and the types present at higher dimension reductions (where the NMF reduced space has in fact fewer dimensions) are by and large subdivided in spaces with fewer dimension reductions.

A moral we draw from this is that primitive types are never plainly of one higher type or not. They are more or less close to any given higher type, which on the NMF approach is an NMF dimension. Given that the space of types is continuous, there is no clean cutoff, for example, between those types that are subtypes of the "higher" type and those that are not. Consider for instance the graph in figure 4 for the NMF factorization with $k = 10$. For each of the 10 dimensions, the value for each of the nouns on that dimension is plotted in sorted descending order. All of the dimensions exhibit a hyperbolic curve with a relatively long tail, exemplifying the continuous nature of the latent dimensions.

These continuous curves reflect usage. Consider simple types that might be subtypes of HUMAN and those that are not. Animals like cats are often anthropomorphized, and the word *cat* will

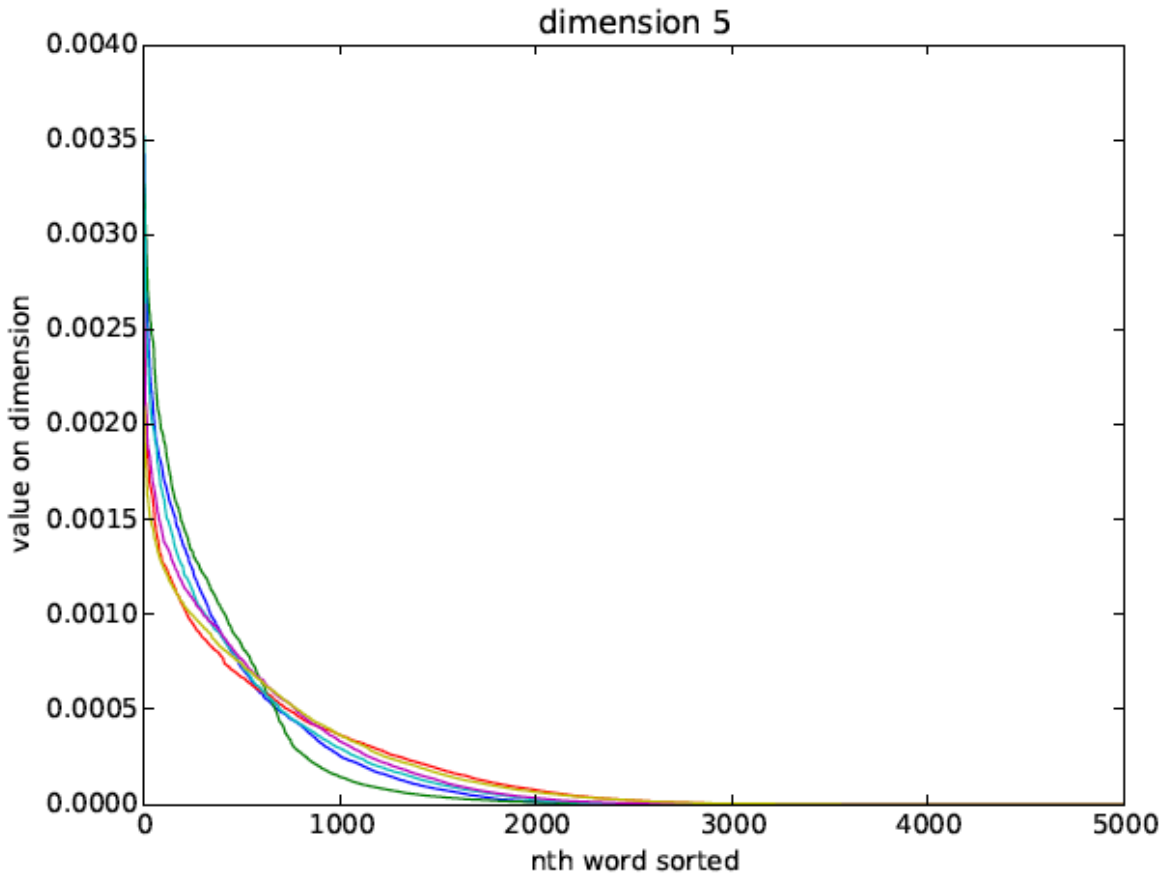

Figure 4: top 5000 nouns relative to $k = 10$ dimensions

participate in many of the same predications that the word *person* does. So the vector or type for *cat* will be closer to that for *human* than say *chair*. But ontologists would plainly balk at putting cats as a subtype of humans. Since general words also have types, we can compare the distributions over the more general types and the more specific ones. A subtyping relation would have to come in degrees as well. For we could compute subtyping in terms of a distance relation from the core type, but establishing a cutoff value to give a binary decision would be arbitrary.

We nevertheless inject a notion of caution on reading off any type structure from these results. Because distributional methods confound a number of semantic relations, it would be unwise to draw the conclusion that clustering or dimension reduction methods will tell us a clean story about the subtyping relation without more testing. Clustering gives us some clues as to subtyping, but it confounds many other semantic relations with subtyping. And not every cluster can be thought to be a type that is a type presupposition. For instance, *cat* and *dog* cluster with *animate being* and *physical object* but also with *pet*. Yet the type PET is not a type presupposition of any predicate that we can think of.

If this is right, how would type presuppositions now work? How could we check types for semantic well-formedness? The binary "yes/no" decisions of semantic well-formedness in TCL would function as a limiting case. We could provide a score to each predication depending on how close the fine-grained types are to matching type presuppositions, provided we could isolate type presuppositions in a reliable and robust way, or to each other. The closer the distances, the better or more commonplace the predication. We say *commonplace* because, recall, vectors encode usage and so closer distances just say of a predicate's argument place α and the actual argument that is supposed to fill in α in the space of the most common syntactic/semantic constructs share many of those dimensions. On the other hand, a limiting case like *soft number* in example (10) would have a large value reflecting the fact that the type presuppositions of *soft* and *number* are incompatible. Thus TCL's binary view of semantic well-formedness would morph into a more graduated scale, which might more accurately reflect the intuitions of ordinary speakers. For instance, while both (28) and (29) involve a type clash between the demands of the predicate and the type of the argument, there is a difference in interpretability, which a "soft " theory of types can now capture.

(28) The square root of two is blue.

(29) The square root of two is a pencil.

Nevertheless, the functor approach of TCL would still differentiate between co-composition and coercion. Only the latter seems to introduce new discourse entities [1, 3].

5.3. Distributional semantics and TCL functors

Our view of types as vectors also answers the question that has hovered over us during the whole paper: what triggers the meaning shifts in co-composition? Given our new view of types as highly context sensitive objects, there is no need for triggering: there is always some sort of shifting due to composition, because composition fills in the context for a given word's type and thus shifts it slightly. The DS counterpart of a TCL functor is in fact a transformation of v in V into a vector v' where the values on certain dimensions have been modified due to the presence of the context being filled in slightly. In the latent vector weighting model, what we do is use two vector spaces, the original vector space V where each word is represented in a space of syntax/semantics contexts and a vector space V' with reduced dimensions, where lexical meanings have a more topical representation. Computing the conditional probability of each dimension z of V' relative to the vector for the adjective, then provides a way of calculating the probability of each element of V given the presence of the adjective. This "slightly more determined context" vector $v*$ now furnishes our functor: $\lambda v v * .v$, where $v * .v$ signifies the point-wise product of the two vectors, though other combinations are possible. For subsective nouns the distance in terms of similarity between v and v' is close, but for adjectives this spread could be considerable.

Since our vectors are just points in $\mathbb{R}^n$ and all of our transformations are linear maps, we can observe a constraint on functors that says that functors should not move types too much. As types are now points in a metric space, we can apply the Lipschitz condition.

Definition 1 *A function f obeys the* Lipschitz condition *iff for any points $x, y \in \mathbb{R}^n$ $\|f(x) - f(y)\| \leq C\|x - y\|$, where C is some constant.*

The Lipschitz condition entails that that functors should treat similar types similarly. For different sorts of modifiers, we could set the Lipschitz constant to different values. Lipschitz is satisfied by our vectorial interpretation of our functors.

What do these interpretations of functors do for truth conditions in TCL? The spelling out of TCL vectors should encode the information about shifted meanings, which is something we could do by translating our top ten closest vectors for co-composed adjectives and nouns into a conjunction of predications involving the head argument of the noun.

However, there are other constraints that we might want to put on our meaning shifting functors. We might add a criterion of *minimal change* inspired by theories of belief revision [14]: the functor should change no more of the basic meaning of its argument than is required to accommodate the type demands of the predicate. It is not obvious at all that our distributional method of calculating the functor or any other distributional method of composition that could be adapted to calculate such a functor obeys such a minimal change constraint at present. And there is a certain amount of work to be done just to make this constraint precise. Following the work on belief revision,

this would mean that we would have to assign a partial ordering over contextual dimensions thus imposing a structure on the vectorial representation that it does not have in and of itself. It is unclear to us at present what such an ordering should be or how it could be implemented.

6. Conclusions and prospects

In this paper we have looked at a formal model of co-composition. We have demonstrated that this phenomenon is widespread, far more so than coercion, but it is only the latter to which type-theoretic approaches to lexical semantics have paid considerable attention. DS approaches predict that co-composition is just a fact of composition. Furthermore, there is a continuum of cases between true coercion in which repair is needed and co-composition. This is what our DS realization of the functor approach tells us. In this respect standard conceptions of types and type-theoretic semantics are at a disadvantage; the division between co-composition and coercion is too large and predicts a difference of principle where there may be none. They may even be just two perspectives on one and the same process. We have also shown that some formal models, in particular TCL's functor approach is quite compatible with a DS approach and we think that both are mutually complementary. DS vectorial approaches provide us with a way to fill in a type hierarchy that theories like TCL presuppose and provide some sort of content for fine-grained and more general types.

As a concluding and somewhat of a side note, what can formal semantic, type-driven theories of lexical semantics and composition give to distributional methods? We have said little about this in the body of the paper. Nevertheless, we think there are several advantages to using a formal theory and a distributional theory together. DS theories are strongest at the individual word level. Formal theories are stronger at higher levels; they provide a logical form that supports many inferences and also a clear proof-theoretic semantics for functional words and morphemes that formal semantics has already had success in analyzing. Finally a theory like TCL also provides a denotational semantics and truth conditions or dynamic semantic update conditions for sentences or whole texts, which DS in its present form is completely incapable of providing. And it's not clear that DS will ever be in a position to furnish such things. We hope to explore this marriage of convenience further in future work.

References

[1] Nicholas Asher. *Lexical Meaning in Context: A Web of Words*. Cambridge University Press, 2011.

[2] Nicholas Asher. Types, meanings and coercions in lexical semantics. *Lingua*, 157:66–82, 2015.

[3] Nicholas Asher and Zhaohui Luo. Formalization of coecions in lexical semantics. In Emmanuel Chemla, editor, *Proceedings of Sinn und Bedeutung*, Paris, 2012.

[4] Nicholas Asher, Tim van de Cruys, Antoine Bride, and Márta Abrusán. Integrating type

theory and distributional semantics: a case study on adjective-noun compositions. *Computational Linguistics*, in press.

[5] 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 Natural Language Processing*, pages 1183–1193, Cambridge, MA, October 2010. Association for Computational Linguistics.

[6] Robert Brandom. *Making it explicit: Reasoning, representing, and discursive commitment*. Harvard University Press, 1998.

[7] Antoine Bride, Tim van de Cruys, and Nicholas Asher. Une évaluation approfondie de différentes méthodes de compositionalité sémantique. In *Proceedings of TALN 2014*, pages 36–44, Marseille, 2014.

[8] Antoine Bride, Tim Van de Cruys, and Nicholas Asher. A generalisation of lexical functions for composition in distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 281–291, Beijing, China, July 2015. Association for Computational Linguistics.

[9] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis: A Festschrift for Joachim Lambek*, 36(1-4):345–384, 2011.

[10] Georgiana Dinu and Mirella Lapata. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172. Association for Computational Linguistics, 2010.

[11] Johannes Dölling. Flexibility in adverbal modification: Reinterpretation as contextual enrichment. In E. Lang, C. Maienborn, and C. Fabricius-Hansen, editors, *Modifying Adjuncts*, pages 511–552. de Gruyter, Berlin, 2003.

[12] Michael Dummett. *The Seas of Language*. Oxford University Press, 1993.

[13] Markus Egg. Beginning novels and finishing hamburgers: Remarks on the semantics of *to begin*. *Journal of Semantics*, 20:163–191, 2003.

[14] Peter Gärdenfors and David Makinson. Revisions of knowledge systems using epistemic entrenchment. In Moshe Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 83–95, San Francisco, 1988. Morgan Kaufmann.

[15] Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[16] Hans Kamp and Barbara Partee. Prototype theory and compositionality. *Cognition*, 57:129 –191, 1995.

[17] Zhaohui Luo. Type-theoretical semantics with coercive subtyping. *SALT20, Vancouver*, 2010.

[18] Zhaohui Luo. Contextual analysis of word meanings in type-theoretical semantics. *LACL'11, LNAI 6736*, 2011.

[19] Zhaohui Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 2012.

[20] Ofra Magidor. *Category Mistakes*. Oxford University Press, 2013.

[21] J. Mitchell and M. Lapata. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[22] John Mitchell. Coercion and type inference. In *Proc. of Tenth Annual Symposium on Principles of Programming Languages (POPL)*, 1983.

[23] Barbara Partee. Privative adjectives: subsective plus coercion. In Rainer BÄUERLE and T. Ede ZIMMERMANN, editors, *Presuppositions and Discourse: Essays Offered to Hans Kamp*, pages 273–285. De Gruyter, 2010.

[24] Barbara Partee and Vladimir Borschev. Privative adjectives: Subsective plus coercion. Forthcoming in a *Festschrift* for Hans Kamp, 2004.

[25] Barbara Partee and Mats Rooth. Generalised conjunction and type ambiguity. In Bauerle, Schwarze, and von Stechow, editors, *Meaning, Use, and Interpretation of Language*, 1983.

[26] Stephen Pulman. Aspectual shift as type coercion. *Transactions of the Philological Society*, 52(2), 1997.

[27] James Pustejovsky. *The Generative Lexicon*. MIT Press, 1995.

[28] Peter Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.

[29] Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. Latent vector weighting for word meaning in context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1012–1022, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[30] Ludwig Wittgenstein. *Philosophical investigations*. Blackwell's, Oxford, 1953.