Chapter 3
A Type Composition Logic for Generative Lexicon

Nicholas Asher and James Pustejovsky

3.1 Introduction[1]

Recent work in discourse semantics has focused on modeling the determinants of meaning for linguistic utterances beyond the level of a single clause. As more parameters of interpretation have been incorporated into our model of meaning, the assumption sregarding compositionality have become much more complex (cf. Groenendijk and Stokhof 1990; Kamp and Reyle 1993; Asher 1993; Asher and Lascarides 2003). Similarly, at the level of the clause, richer notions of composition and lexical structure have surfaced to explain the systematic variation in meaning involved in polysemies and polymorphisms (cf. Nunberg 1995; Moravcsik 1975; Pustejovsky 1995; Copestake and Briscoe 1995; Jackendoff 1997). This tradition in lexical semantics argues that we need a notion of composition for which, combining the meanings of two words may result in a change to those meanings themselves. We concentrate here on one problem in particular, that of *copredication*, where apparently incompatible types of predicates are applied to a single type of object. As argued in Nunberg (1995), Pustejovsky and Boguraev (1993), and Copestake and Briscoe (1995), to handle such cases some context-sensitive notion of composition is needed, which is not what one finds in the standard theory of

[1]This work forms the basis for more extensive discussion of the Type Composition Logic, presented in Asher (2011), and its application in Pustejovsky (2006, 2011).

N. Asher (✉)
CNRS, Toulouse, France

Department of Philosophy, University of Texas, Austin, TX, USA
e-mail: nasher@bertie.la.utexas.edu

J. Pustejovsky
Department of Computer Science, Brandeis University, Waltham, MA, USA
e-mail: jamesp@cs.brandeis.edu

compositionality exemplified in classical Montague Grammar (Montague 1973). We believe these shifts in meaning during composition to be a matter of lexically-governed shifts in *semantic type*—in a manner similar to earlier work on "type shifting" (Partee and Rooth 1983; Klein and Sag 1985; Hendriks 1993). In this paper, we develop a method of composition that adds to the contents contributed by lexical elements when certain word meanings combine. At the end of this paper, we extend our method to treat other phenomena like the *qualia* that Moravcsik (1975) and Pustejovsky (1991, 1995) introduced to explain phenomena that are difficult to account for on a simple context-insensitive method of building sentence meanings.

In a similar vein, recent advances in discourse interpretation have furnished a way of integrating pragmatics and semantics together into a context sensitive theory of discourse interpretation. SDRT is one such approach (Asher 1993; Lascarides and Asher 1993; Asher and Lascarides 2003); exploiting the rhetorical function of information, it introduces a context-sensitive method of calculating the *discourse update* of a discourse with new information—viz., new information may be added to the context in a number of different ways reflecting distinct rhetorical functions. In the pair of examples in (1), for example, two very different rhetorical functions create coherent interpretations, but with different temporal and causal structures:

(1) a. John entered. Max greeted him.
    b. John fell. Max pushed him.

The interpretation of *Narration* in (1a) is consistent with the updates and lexical information associated with *enter* and *greet*. This relation is not consistent with (1b), however, while the relation *Elaboration* is.

Both GL and SDRT are reactions to theories of the lexicon and discourse update (i.e., an atomistic Fodorian lexicon (Fodor and Lepore 1998), and standard dynamic semantics, respectively), that fail to account adequately for a wide variety of phenomena having to do with the pragmatics/semantics interface. What earlier theories lack is an account of how the "composition" of new information in context could in fact alter the information as well as the elements in the context, in ways not predictable within a framework countenancing only operations like *lambda conversion* or *merge*. GL and SDRT make this the core of their approach to meaning.

Broadly speaking, context-sensitive approaches to both lexical composition and discourse interpretation have a common view about meaning, some of the same formal tools, and some of the same problems. GL and SDRT both use nonstandard formalisms to compute logical forms, for which model theoretic interpretations can be supplied. SDRT makes use of a special purpose "glue" logic with limited access to various information sources for building up logical forms of discourses from underspecified logical forms for clauses. This glue logic has a limited and partial access to the information content of discourse constituents, Asher and Fernando (1997) and Asher and Lascarides (2003) argue, because full access to information content would render the task of computing logical forms hopelessly complex. Though we do not believe all of linguistic understanding should necessarily be computationally simple, computing logical forms, which is the prerequisite to any deeper understanding, should be a simple matter. This in turn leads to a strong distinction in SDRT between information available to the linguistic system, of which the glue logic is part, and nonlinguistic information or world knowledge. This separates SDRT and us from competing approaches (e.g., Hobbs et al. 1993), which assume that a general purpose reasoning system makes no distinctions between linguistic and nonlinguistic knowledge.

The distinction between lexical and world knowledge as made in GL has strong linguistic motivation, as argued in Pustejovsky and Boguraev (1993), Jackendoff (2002), and Moravcsik (1998). The task of a lexicon is, at the very least, to supply the semantic material introduced by each word into the logical form of a clause. We suppose further that this information must be capable of model theoretic interpretation though we will not examine any of those details here. Secondly, this information must be able to combine compositionally, insofar as this is possible, to yield the logical form for a clause. There are constraints, or selectional restrictions, involved in this information that the lexical entry for each word must carry. For instance, the verb *weigh* takes a degree phrase or something denoting a quantity of weight as its second argument while its first argument must be a physical object of some sort. The verbs *recount* or *describe*, on the other hand, cannot take *merely* physical objects as first arguments. The fact that these verbs cannot take arguments of a certain kind leads to semantic anomaly when we attempt to violate these constraints, as can be seen from the examples below '!' indicates for us semantic anomaly.

(2) !Bob's idea weighs five pounds.

(3) !Bob's sack of fertilizer recounts the events leading up to the Civil War.

Hence, one can view the semantic component of a lexical entry as consisting of one part determining the model-theoretic content and another part carrying information that enables it to combine with other bits of lexical information to give a meaning for a whole clause. This latter sort of information should state constraints about the types of arguments the lexical entry either requires or introduces in the logical form.

Typed unification grammars and type calculi are the two main frameworks in which to carry out such a project in a way consonant with current approaches to syntax. GL's rich approach to lexical meaning was originally couched within a unification like framework (Pustejovsky 1995), but many of the formal mechanisms were not spelled out in complete detail. One of our tasks here is to provide some of those details. We believe that a natural deduction style type calculus with complex types of the sort we present here is quite suitable to this task. This approach is inspired by Howard's (1980) seminal paper and the topic of current work in polymorphic typed calculi (Amadio and Curien 1998; Crole 1993) (see also Crouch and van Genabith 2000). Thus, as in Montague Grammar and other standard frameworks, we will take a lexical entry to consist in a lambda term and a type assignment to the variables in the term. This will then determine via the standard interpretation for the lambda term a functional type for the whole expression. Unlike Montague Grammar, however, our *type composition logic* (TCL) will have a much richer system of types reflecting the information conventionally associated with a word in the GL approach, and correspondingly more complex rules for manipulating these types. Like SDRT's glue logic, the type composition logic builds up logical forms; but the composition logic builds up clausal logical forms (CLFs), whereas SDRT's glue logic builds discourse logical forms (DLFs) from CLFs. Like the construction of DLFs, the process for constructing CLFs is also quite simple. But again like SDRT's glue logic, the type composition logic has partial access to common sense information or world knowledge, which ultimately determines the compatibilities and incompatibilities between semantic types. With partial access to common sense knowledge, the type composition logic can exploit this information in guiding shifts in type during semantic composition more efficiently. Nevertheless, word meaning is distinct from non-linguistic or world knowledge at least in form and scope. Metaphysical information is drastically simplified into conventionalized type information; as a result, the type composition logic will be a drastically simplified reflection of certain ontological principles that underlie general reasoning. Hence, SDRT's approach to computing logical form will thus be reflected in the type composition logic for GL developed in this paper.

While SDRT's approach to discourse meaning and GL's approach to lexical meaning share many features, we believe it is important, to keep discourse interpretation and lexical semantic modules distinct. Many people have advocated dispensing with this distinction, where one general purpose pragmatic engine handles all reasoning operations homogeneously (e.g., Hobbs et al. 1993). We believe this approach is misguided for two reasons. First, the glue logic and the type composition logic have very different tasks. The type composition logic primarily checks the lexical type assignments in applying one lambda term to another and resolves type conflicts when they arise, as in cases of type coercion and co-composition. SDRT's glue logic, on the other hand, resolves elements left underspecified by lexical elements in the compositional process; it computes the optimal attachment points for new information in a discourse structure as well as the rhetorical roles for this information. The second reason for keeping lexical and discourse processes distinct is that the two systems interact in subtle and interesting ways, and merging these two modules would make it more difficult to formulate these distinctions systematically. Discourse structure and context, for example, can obviously affect lexical interpretation in context. Here we see an example of how it affects "logical metonymy".

(4) The goat hated the film but enjoyed the book.

Depending on whether the context is a fairy tale or not, (4) will convey the same or different sense of *enjoy* as that assumed in (5).

(5) The boy hated the film but enjoyed the book.

That is, discourse context can alter types: in a fictional interpretation, goats can become talking, thinking and reading agents, thereby assuming characteristics that they would not normally have, due to their sortal typing. Thus, conventional and lexical associations such as those encoded in the interpretation for (5) can be overturned by new or more specific information in a particular discourse context.[2] Furthermore, lexical ambiguities can be resolved by discourse in ways that override lexical preferences (Asher and Lascarides 1995).

In this paper, we begin to explore generally the integration of GL and SDRT processes, the problems that such an integration faces, and what advantages it might offer. Specifically, we concentrate on developing the type composition logic required to model one of the complex types of GL, for which we employ the various SDRT principles and strategies we've already outlined. As we are interested in the composition of information to construct logical forms, we will build on the standard way of getting logical forms, namely, the lambda calculus in which functional types are exploited. By relating types in the lexicon we can give partial, implicit definitions, which will help together with how the items compose, to determine inferences based on truth conditional contents. Secondly, by developing a strongly typed theory of lexical items and a theory of how such lexical items combine and interact in the process of semantic composition and of discourse interpretation, we can constrain the lexical semantics with predictions of semantically well-formed or ill-formed predications and word combinations. We outline a new type calculus that captures and extends one of the fundamental ideas of GL: providing a set of techniques governing type shifting possibilities for various lexical items so as to allow for the combination of lexical items in cases where there is an apparent type mismatch. These techniques themselves should follow from the way the lexicon is organized and its underlying logic.

3.2 Polysemy and Sense Extension

While the Generative Lexicon is perhaps best known for its development of the notion of *qualia* (based on Moravcsik's 1975 interpretation of *aitia*) another enrichment to the type system proposed in the Generative Lexicon is a complex type introduced to explain *copredications* in the context of polysemy. Copredications involve two or more predications on the same object. Many syntactic constructions give rise to copredications—relative clauses, and small clauses, for instance—but the classic cases of copredication are those that involve coordinated verbs or verb phrases as shown below.

[2]For a fuller discussion and a theory of this interaction using default unification and the glue logic DICE of SDRT see Asher and Lascarides (1995) or Lascarides and Copestake (1995).

(6) a. The book was a huge pain to lug home and turned out to be very uninteresting.
b. Mary picked up and mastered three books on mathematics.
c. The bottle has a nice label and is a merlot.
d. The temperature is ninety and rising.
e. Lunch was delicious but took forever.
f. The bay curves from the lighthouse to a sandy spit and is lovely to swim in.

The copredications that interest us involve predicates that select for two different, even incompatible types. In GL the underlined nouns receive a complex type; the so-called *dot objects* of GL first introduced by Pustejovsky (1994) are, in effect, best understood as objects of a particular complex type with two constituent types. The constituent types pick out aspects of the object, and the object's complex type reflects the fact that it may have several, distinct, even incompatible aspects. The term *dot object* thus refers to objects with a complex type (*not* to complex objects—whatever those might be—or to pairs of objects),[3] with several aspects, which have become part of the meanings of the words that denote such objects. Such dot objects allow for predications that are licensed over either of the two dot element types (see Pustejovsky 1995, 1998 for details).

Another mark of dot objects and the copredications that interest us is that neither typing required by each of the coordinated verbs or verb phrases of (6) fits fully comfortably as a dependent type of the other. For example, the verb *pick up* types its object as physical, whereas the verb *master* types its object as informational. Similarly, the figure and ground aspects inherent in the meaning of *bay*, like the physical aspect and informational aspect of a book are mutually interdependent; you cannot have one without the other. The intuition is that copredication requires these two types to be accessible *simultaneously* during composition; the function of dot objects is to make this possible.[4]

[3]Here the notation of earlier work on dot objects suggested these interpretations; but our approach here is resolutely different from those older attempts at description. We are very explicit that • is a type constructor and has nothing to do with the construction of a complex object.

[4]Not all copredications need involve dot objects. Some may exploit events that are conventionally associated with the types of the subjects, like those described in qualia structure. In (7), for example, it appears as though some predicates make reference to aspects having to do with the so called telic qualia role of the subject NP they are predicating i.e., the smoking and drinking events, respectively.

(7) a. Arnold's cigar is Cuban and lasted the whole afternoon.
b. Your last glass of wine was a Merlot and lasted half an hour.

Hence, copredication does not uniquely identify NPs typed as dot objects. Similarly, it is unclear whether *grinding* operations, which also license copredications, should be analyzed as

There are of course constraints on what dot objects can be formed. We see this when copredications become odd, zeugmatic or just unacceptable. Thus, as in (8) below, we see that contrastively ambiguous words (Pustejovsky 1995) do not introduce a dot object, where two distinct senses are simultaneously accessed. Hence, such words cannot support copredications.

(8) !The bank specializes in IPO's and is being quickly eroded by the river.

On the other hand, we see that many words appear to give rise to complex types, though not all copredications are equal (cf. (9b)). We believe that this has to do with the fact that a dot object's existence may depend not only on commonsense metaphysical intuitions, that are conventionalized as typing information in the lexicon, but also on discourse context and the rhetorical connections between the two predications.[5] For example, a noun such as *newspaper* denotes an object that has both physical and informational characteristics and so would have a complex type consisting of the type of physical objects and the type of informational objects. *newspaper* actually can denote a related entity, the organization that produces the objects of physical • informational type, but this type doesn't combine very well with the physical type, as copredications like (9b) are semantically anomalous, even though copredications involving the organization *as an agent* and the information in the newspaper are acceptable (9c).[6]

(9) a. The Sunday newspaper weighs 5 lbs and documents in depth the economic news of the week.
b. !The newspaper was founded in 1878 and weighs 5 lbs.
c. The newspaper contains some really useful information about restaurants and concerts but publishes a lot of useless junk as well.

What these examples demonstrate is the polysemous (and apparently polymorphic) nature of nouns such as *newspaper*. A dot object is a packaging of both types, reified through a coherence relation as one complex type, with the ability to exploit aspects of its type structure in diverse predicative contexts.[7]

involving dot objects, type-changing operations, or involve the exploitation of lexical information from the qualia structure. See Pustejovsky (1995) for discussion.

[5]The felicity of copredications often depends on the order of the predications as well. This again we feel is due to discourse factors. We don't go into this here, as it would involve bringing in too much of the SDRT framework, obscuring our restricted aim here to provide a type composition logic. In any case, we will keep such rhetorical constraints on felicitous copredications with dot objects separate from the composition logic.

[6]As a result, such concepts are actually double dot objects, but we ignore this point for now, cf. Pustejovsky 1995.

[7]We note as well that such types may be subject to discourse effects like parallelism; for instance, (9b) improves if we shift the second event to the past:

(9b') The newspaper was founded in 1878 and weighed 5 lbs in its first edition.

An alternative approach to these cases of copredication is not to postulate complex types for the argument of the predicates, but rather to change the types involved in the individual predications. Thus, we might try changing the type of the verb *document* so that it takes a physical object as a subject, but the verb phrase means roughly "instantiates an informational object that documents in depth the economic news of the week."[8] This approach, however, runs into immediate trouble, because we can't explain then why such a type shift works only with certain arguments, which on this view are all of some simple type. While newspapers, books, and theories can document something in the relevant sense, walls, windows, flowers, rocks and trees cannot. This selectivity is immediately explained, however, if we require all arguments of *document* to be of, or to have as a constituent type, the informational type. Under this analysis the verbs do not shift; and since *document* requires an informational object as its subject, sentences like *The wall documented the news of the week* will not yield a felicitous logical form because of the typing mismatch. On the other hand, certain nouns like *newspaper* introduce lambda terms whose main variable has a complex type containing both informational and physical types as constituents. In predication, *newspaper*'s type can be adjusted to one of its simpler constituent types so that the types match and predication succeeds. This is not to say of course that verbs *cannot* undergo type shifting; verbs of creation such as *bake* do appear to have distinct but related meanings, depending on the exact nature of their arguments. However, the copredications that we are interested in cannot be treated adequately by shifting the types of the predicates.

Copredications involving relative clauses and adjectival modification also sometimes require their arguments to be of complex type. For example, as pointed out in Pustejovsky (1998), some lexical items denote both an event and a participant in this event, as with the noun *dinner*. Both aspects of this complex type may be predicated, as witnessed in the sentence below.

(10) John stopped by during our delicious dinner.

The preposition *during* selects for a temporal object of type event or interval, while *delicious* selects a comestible substance. The noun *dinner* satisfies both these typing restrictions by virtue of its type, namely, its statusas a dot object denoting both event and substance.

Further, we have evidence that this information is so far conventionalized that it even affects the case system in some languages. There is considerable consensus

But we will not attempt to integrate such discourse effects with our story about complex types here.

[8]Klein and Sag (1985) take this approach to multiple subcategorization phenomena. However, as discussed in Pustejovsky (1995), type-shifting the predicate in such cases does not change the basic meaning of the predicate, but only the surface typing. Both Klein and Sag's analysis of *believe* and Godard and Jayez's (1993) treatment of coercion predicates involve meaning postulates to relate verb senses. The alternative analysis here is similar to the sense transfer operation proposed in Nunberg (1995). As we see, however, this is an inappropriate use of transfer.

that languages distinguish types for places (fixed elements in the terrestrial reference frame) and types for objects (elements that have a complex internal structure and can move with respect to the terrestrial reference frame). Evidence for these distinct types comes from Dutch, for example, where there are special pronouns for referring to locations.[9]

(11) a. Dat is een mooi weiland.
Daarin houd ik mijn koeien.
*In het houd ik mijn koeien.
b. That's a nice field.
Therein I keep my cows.
*In it I keep my cows.

Further evidence for this distinction comes from Basque, where the grammar encodes differences between location and objects via two genitive cases *-ko* and *-ren*; locations in general easily take the genitive *-ko* but not *-ren*, while objects in general do the reverse (Aurnague 2001). Aurnague (2001) distinguishes the following sortals: *places* (e.g., valley, field, river, mountain, hill), *objects* (e.g., apple, glass, chair, car), and *mixed objects* (e.g., house, church, town hall). Of particular interest are the "mixed objects" and the behavior of their expressions in Basque: they readily accept both forms of the Basque genitive. So if we accept the encoding hypothesis for Basque, mixed objects would appear to belong to two types, or two ontological categories, at the same time, PLACE and PHYSICAL-OBJ, neither of which is a subtype of the other (it is neither the case that the properties associated with physical objects are inherited as properties of places nor that the properties associated with places are inherited as properties of physical objects).

(12) Maite dut etxeko atea haren paretak harriz eginak direlariak.
(Michel Aurnague p.c.)
I like the door of the house the walls of which are made of stone.

More motivating data for the existence of dot objects comes from the following minimal pairs, involving quantification over different aspects of the meaning of the nouns *book* and *question*. Consider the sentences below.

(13) a. The student *read* every book in the library.
b. The student *carried* off every book in the library.

(14) a. The teacher *answered* every student's question.
b. The teacher *repeated* every student's question.

[9]This point is due to Melissa Bowerman, pc.

The quantification over books in (13) is sensitive in one case to its informational aspect, and in the other to its physical aspect. In (13a), we simply quantify over all informationally distinct individuals without reference to the instantiations of these informational units; it is not necessary, for example, for the student to have read every distinct copy of every book in the library. In (13b), however, every physical individual must have been taken in order to be true. Similar remarks hold for the distinction in (14b): an answer to the same question posed on multiple occasions will count as an answer to each question; this is not the case with the act of repeating the question, however, since this refers to copying the speech act rather than providing the informational content of the answer.

One might think that a simple account of these examples would just involve coercing *book* to be, in some cases, a physical object, and in other cases, an informational one. Such an analysis, however, makes it difficult to explain the copredication data. Furthermore, as with the construction in (10) above, we need access to both types simultaneously, in order to explain the predications for cases such as (15) and (16) below.

(15) John's mother burned the book on magic before he mastered it.

(16) Mary bought a book that contradicts everything Gödel ever said.

Since the verb *master* in (15) involves selecting for the informational sense of book, we cannot "use up" the dot object *book* when predicating burning of it in the first sentence. Otherwise we will be unable to bind the anaphor in the second clause; alternatively, if we try to coerce the object of *master* back to an informational object, we get a typing conflict with the typing requirements of *burn*).

In addition to copredication constructions, there are other grammatical and lexical devices that introduce or select dot objects. Pustejovsky (1998) argues that the verb *read* is a predicate that requires a dot object as its complement; it can even coerce its direct object into something of just this complex type, namely, an informational entity with physical manifestation.

(17) a. Mary $\underline{\text{read}}$ the book.
b. John $\underline{\text{read}}$ the rumor about his ex-wife.
c. Mary $\underline{\text{read}}$ the subway wall.

The coercion phenomenon in (17) involves a subtle shift in meaning. One can hear rumors and spread rumors, which one cannot do with books (even if you're listening to a book on tape); on the other hand, one can't see or look at rumors whereas one can see or look at a book. On the other hand, one can see a subway wall or look at it, without getting any informational content. However, in (17b, c) the arguments of *read* change their meaning. For instance, (17c) implies that the subway wall is a conveyor of information, and the only way to understand (17b) is to assume that the rumor has been printed or exists in some physical medium. One explanation of this phenomenon is that *read* coerces its arguments into objects of the same type as *book*. For both (17b) and (17c) the predicate coerces its complement to the appropriate type, that of an informational object with physical manifestation. In each of these cases, there is a "missing element" to the complex type: for (17b) the coercion effects the introduction of the physical manifestation to the otherwise informational type; for (17c) the coercion results in the introduction of an informational component to an otherwise merely physical type.

Barbara Partee has suggested (p.c.) that one might handle the quantificational ambiguity seen above with *read* and *carry off* by treating the entire phenomenon as an instance of the type/token distinction. According to this suggestion, (13a) makes reference to the type while (13b) refers to the token. While not discounting this approach completely, there appear to be two problems with this solution. First, simply reducing the above phenomenon to a type/token distinction does not solve the problem of how the copredication works; if the type/token suggestion were right, we could envision using that distinction along with our dot object apparatus in the analysis, but without the latter, it is not clear what the analysis would be. Furthermore, there are cases where reference seems to be made to more objects than are available under a simple type/token analysis. For example, in (18b), quantification is over informational tokens that are distinct from the actual physical object tokens that would be available.

(18) a. John hid every Beethoven 5th Concerto score in the library.
b. John mastered every Beethoven 5th Concerto score in the library.

Hence, for a dot object, if there are type and token interpretations available for each component type of the dot, then the underlying typing is more complex than originally countenanced.

One final argument against a type/token distinction for cases of dot object subselection can be seen in examples such as (19) below.

(19) a. John has stolen every book there is.
b. Frances has grown every wildflower in Texas.

While there are (improbable) interpretations exploiting the token reading of the quantified expression in each example above, the type interpretation is more felicitous. However, the interpretation of the generalized quantifier in (19a) makes clear that the type reading of *every book* is distinct from the informational content interpretation of the dot object in sentence (13). That is, the verb *steal* selects for physical instantiations of kinds of books. This is the true "kind interpretation", but it is distinct from that seen with the exploitation of part of a dot object from the verb *read* in (13).

We will re-examine much of these data from the perspective of the type composition logic we develop, later in the paper. First, however, we wish to turn to the metaphysical picture suggested by complex types and what in general should be the relation between commonsense metaphysics and the lexicon.

3.3 Constraints on the Mapping to Semantics

Thus far, we have seen evidence that common sense metaphysics and contextual factors constrain the construction of complex types—that is, which arguments we consider as having a complex •-type or, equivalently, of being dot objects. Common sense metaphysics informs lexical semantics by providing the basic types and basic relations between types. It also acts as one constraint on whether certain complex types are admissible (discourse context is another). The general metaphysical picture is that objects of complex type have non-necessarily spatio-temporal parts or aspects to them that fall under the simple types that are constitutive of the complex type. The information encoded in metaphysical categories is "lifted" conventionally into the type structure and then exploited in semantic composition. Predication, the application of a property to an object, may sometimes be restricted to a particular aspect of an object, something known in scholastic philosophy as *qua predication*, where philosophers speak of an X qua Y as having the property P. We think that such restricted predication need not be overtly marked in ordinary language, though it can be (see Asher 2004). When we need to look only at one aspect of a (dot) object of complex type, we assume that the predication involving the simple aspect is an "object-elaboration" of the dot object—it's elaborating on one aspect of the object.[10] For short, we will call this link *O-Elab*. O-Elab is a not necessarily physical, antisymmetric and transitive proper-part-of relation.[11]

The way predications behave actually tells us something about the metaphysical relation between aspects and things that have them. Aspects are mysterious, metaphysical beasts—they are some sort of individual trope perhaps. From the perspective of lexical semantics, however, aspects are atoms, and objects of •-type are just mereological sums of their aspects; •-types are hence idempotent, associative and commutative. We'll assume in addition that x is of type σ and y is of type τ and we have *O-elab(z,x)* and *O-elab(z,y),* then $x = y$; i.e. parts of an object singled out for predication that are of the same aspect are identical.

There is a further connection between commonsense metaphysics and the lexicon, but it is not a direct one. Metaphysics permits the construction of some complex types but not others. We represent this simply as a condition in the composition logic as, $\diamond x\colon \sigma \bullet \tau$, which states that it is consistent with information sources that are relevant to the lexicon, for the variable x to have the complex type $\sigma \bullet \tau$. This imposes, in effect, a *fence* or filter from commonsense metaphysics to lexical information.[12] One reason to distinguish commonsense metaphysics from the lexicon is that metaphysics is only one contributory factor to the logic of

[10]The name is intended to evoke an analogy to a similar relation in discourse. But the development of that analogy is for another time.

[11]In Asher (2004), the O-Elab relation is assumed to be asymmetric, but the work that is supposed to do there is perhaps better explained on pragmatic grounds than by stipulating a strange part of relation.

[12]For more on fences and their usefulness in discourse semantics, see Asher and Fernando (1997).

composition. Clearly both syntax and morphology contribute to the construction of semantic argument structures, while discourse context can also affect the semantic types in the lexicon. Hence, the fence $\diamond x$: $\sigma \bullet \tau$ may also function as a purveyor of information from context.

Our main reason for distinguishing between the lexicon and metaphysics is to distinguish the conventional aspects of word meaning from general world knowledge. If the lexicon is distinct from metaphysics, we open up the possibility that complex types only attach to some words and not others. Conventions will decide what words introduce complex types and what those complex types are. We will show how to account for such cases below in detail, but our point here is that by distinguishing metaphysics from the lexicon, we can both maintain that something like a person may have many aspects that are not part of the lexical entry. For instance, Nicholas Asher may have an aspect of which he is a philosopher, to which we can refer in language by means of the *qua* construction: *Nicholas as a philosopher* (Asher 2004). Nevertheless, there is no evidence that such aspects enter into the dot types for lexical entries.

But of course word meaning, at least the typing information that we are interested in, also in some sense reflects the way the world is and our commonsense metaphysics. The way we distinguish between lexical meaning and world knowledge is primarily a difference in the way this knowledge is presented. The type language of the lexicon is less expressive than that of commonsense metaphysics. The lexicon simplifies information that percolates up to it from commonsense metaphysics in many ways. First, type information is quantifier free, whereas it is hard to imagine any formalization of commonsense metaphysics doing without quantification—typically such formalizations exploit higher-order quantification. A second way our type logic will be simpler is that it will exploit type hierarchies, in which for instance incompatibilities are already precomputed (given by metaphysics). This makes the knowledge of the meaning of words much simpler and computationally much easier; and further, as our composition logic will use some default rules, the simplicity of the basic language is technically needed to make our logic tractable at all. Building logical forms, which is what we are trying to account for, should be relatively easy; it is, after all, a minimum standard of semantic competence for speakers of a language. Building logical forms is different and easier than grasping their full content.

3.4 A Type Composition Logic for GL

3.4.1 The Type Language

In the discussion above, we presented the notion of the complex •-type and related notions. We now need a logic for manipulating these types that will allow us to construct logical forms for interpretation that capture the motivating data from the first section. The data our logic addresses are those that arise from the process of combining meanings. In general this means building a logical form for an entire discourse, thus combining both the type composition logic and the glue logic; we will concentrate on the composition logic here, leaving the interactions with discourse contexts for another venue (see, however, Lascarides and Copestake 1995).

Our logic extends the lambda calculus for functional types with rules for manipulating •-types. These types resemble conjunctive types, and our natural deduction rules for exploiting and introducing them will resemble something like conjunction elimination and introduction. We need sometimes to exploit these complex types when a predicate applies to only one aspect of an object of complex type. But our rules are quite a bit more complicated than the introduction and elimination rules for simple conjunctive types, as they add material to logical form, as well as revise the types of variables. The reason for this is that when we predicate something of an aspect of a thing, we need to encode the information in logical form that the aspect is an aspect of some particular object—we don't want to lose that information since we may refer back to the object or the aspect of it in future discourse.

Besides these rules, we will assume the presence of a type hierarchy with a subtyping relation $\sqsubseteq$ that defines a partial order on the set of types and a greatest lower bound operation $\sqcap$ on the set of types. $\sqcap$ has the usual properties—e.g., idempotence, associativity, commutativity, and $\alpha \sqsubseteq \beta$ iff $\alpha \sqcap \beta = \alpha$. We will capture incompatibility between types in terms of their common meet, $\perp$.

Our type language takes as fundamental the notion of a term together with a typing context or type assignment that our rules can revise or extend. A typing context for a term t determines an assignment of types to all subterms of t. A term together with a typing context represents all the information contained in a typed feature structure. Our rules manipulate these type assignments. Our logic of the lexicon and of logical form construction at the clause level is like that of unification (Carpenter 1992) and other forms of logic manipulating types (Morrill 1992; Hendriks 1993); its complexity is no worse than simple unification, given that its operations are all driven by type adjustments and information about types in the lexicon.

This is not the only way one could go about implementing a composition logic to account for GL representations. Since most of the work on coercion and other generative operations has used the framework of typed feature structures together with the operation of unification (Pustejovsky 1995; Pustejovsky and Boguraev 1993; Copestake and Briscoe 1992; 1995), one might ask why we are proposing a new formalism. But as we have already argued, there are conceptual and computational advantages for our decision. As regards unification, the operation of unification is an efficient way of representing the replacement of one element with another that is determined to be more specific via some partial ordering. But with coercion, subselection, and co-composition, we must *transform* types during semantic composition.

Coercions and co-compositions can be captured via lexical rules (Godard and Jayez 1993; Copestake and Briscoe 1995). Such rules allow us to rewrite given feature structures as new ones. But this approach has several drawbacks. First, these rules allow us to change feature structures in an arbitrary way, whereas for us coercion is precisely the exploitation of something already in the given type structure. Such lexical rules don't discriminate between destructive type shifts like grinding (as in *Rabbit is good to eat and is all over my windshield right now*) and the ampliative inferences that are part of logical metonymy and copredication. In the latter, we *add* information about objects that are the typical denotata of the expressions involved. In the system of type rules to be introduced below, these two types of rules will be distinguished. Logical metonymy and copredication involve ampliative rules like dot and qualia exploitation. Type structures with these rules are not transformed; they are preserved but trigger the addition of new information to logical form. Finally, our framework allows a more flexible relation between world knowledge and the lexicon than that for unification. While head types are typically stable, we imagine that values for qualia structures may be highly contextually dependent, and as such we may be able to form such types dynamically in discourse. Given the standard treatment of qualia structure, these are taken to be universal features in typed feature structures and so are much more rigidly construed.

3.4.2 The Set of Types

We will first define the set of types for the logical system in general terms. We will assume there are *simple* types and complex types, *dot types*, for which we'll use the type forming operator • and functional types for which we'll use the type forming operator $\multimap$ to distinguish this from the material implication $\rightarrow$.

(20) a. PRIMITIVE TYPES: e the general type of entities and $\underline{t}$ the type of truth values. Below σ, τ range over all simple types, the subtypes of e as well as $\underline{t}$.[13]
b. FUNCTIONAL TYPES: If σ and τ are types, then so is $(\sigma \multimap \tau)$.
c. DOT TYPES: If σ and τ are types, then so is $(\sigma \bullet \tau)$.

We assume that the lexicon contains a library of types that determines the type for each lexical item. This library may also evolve as the discourse proceeds, in ways that we will not explore in detail for the present discussion.

The subtyping relation $\sqsubseteq$ affects functional types in the following way. The functional type from a more specific type of object α into β is itself a subtype of the functional type from α' into β' if α is a subtype of α'. Formally this means that: if $\alpha \sqsubseteq \alpha'$, then $(\alpha \multimap \beta) \sqsubseteq (\alpha' \multimap \beta)$. We also will assume that if $\beta \sqsubseteq \beta'$, then $(\alpha \multimap \beta) \sqsubseteq (\alpha \multimap \beta')$. Similar subsumption relations hold for the complex •-types.

Our Type Composition Logic (TCL) has the usual lambda terms familiar from compositional semantics, together with a set of type assignments, of the form $t\colon \sigma$

[13]The details of the relationship between e and its subtypes, as a join semi-lattice, in the simple type domain are spelled out in Pustejovsky (2001,2011).

where σ is some type and t is some term. Constraints on types will also be available; for instance, we may need to know that σ is a subtype of τ, something we express as $\sigma \sqsubseteq \tau$ or that two types are compatible, which we write as $\sigma \sqcap \tau \neq \bot$. We also have (minimal) information about syntactic structure that we will exploit in our rules; for instance, we will have a formula *head* ψ, where ψ is a term telling us that ψ is derived from some projection of the head or the head itself of the syntactic structure whose meaning we are currently trying to build up. We'll discuss this in more detail below. We also need our "fence" formula from discourse context and metaphysics $\Box$x: $\sigma \bullet \tau$. Most likely we will need formulas that allow us to put constraints on what types variables may be, like the reentrancy equations of unification, but we will not use such rules in the present paper.

In order to introduce the specific characteristics of the composition logic, let us examine what is involved in type coercion and subselection phenomena involving a dot object, i.e., a •-type. Consider, for example, the compositional interpretation of the noun phrase in (21).

(21) a heavy book

The interpretation of interest is predication of the book *qua* physical object as being heavy. Let us suppose that the adjective *heavy* is understood as an intersective adjective and so yields the lambda term in (22).

(22) $\lambda P \lambda x[\text{heavy}(x) \wedge P(x)]$

where x:PHYSICAL-OBJECT, or $x : p$ for short, is the type assignment to x. This of course implies that P is assigned type $p \multimap \underline{t}$. The adjective phrase itself has type $(p \multimap \underline{t}) \multimap (e \multimap \underline{t})$. This must combine with the semantic expression for *book* to create a full noun phrase in the DP analysis of syntax that will then combine with the determiner. Let us suppose that *book* introduces a predicate whose argument is conventionally determined to be an object with both a physical and an informational aspect. Thus, it yields the term $\lambda v \text{book}(v)$ together with the typing context $x : \text{physical} - \text{object} \bullet \text{information}$, or $x : p \bullet i$ for short. This implies that $\lambda v \text{book}(v)$ has type $(p \bullet i) p \multimap \underline{t}$. This, however, presents us with a type clash between the adjective's type and the noun's type; that is, we cannot combine these two lambda terms via lambda conversion because the type of the lambda abstracted variable P and the term that is to replace P don't match. Three questions arise in the context of this mismatch. First, should we make a type adjustment? If so, where should the type adjustment in this construction take place, on the type of the adjective itself, on the noun, or on some lower variable?[14] Finally, what sort of type adjustment should be made?

The first question has an obvious answer: since a phrase like *heavy book* is clearly felicitous, some sort of type adjustment should be made to allow lambda conversion

[14]Classic GL analyses (Bouillon 1997; Pustejovsky and Boguraev 1993; Pustejovsky 1995) have argued that adjectival subselection selects for a particular qualia role or the corresponding type for a quale within the feature structure of the nominal semantics. That is, they are typed to modify the particular qualia role of the noun in a specific construction. We compare this analysis to the present one below.

to take place so as to construct a logical form for the NP. For the second question there also seems to be a principled answer, which we state below as (23). The idea of (23) is that the syntactic head of any environment X should determine the typing of X. To be more precise, let's first define a *type clash* between two constituents A and B to occur whenever: if A is function that is supposed to apply to B, then the greatest lower found of the type τ of lambda abstracted variable in A and the type of B is $\perp$ or if B is function that is supposed to apply to A, then the greatest lower found of the type τ of lambda abstracted variable in B and the type of A is $\perp$. For example, A and B will have a type clash, when A is $\lambda x F x$ where $x : p$ and B is y where $y : i$ and $\mathrm{p} \sqcap \mathrm{I} = \perp$. Next, let's define the *tail* of any functional type $\alpha \multimap \beta$ to be β.

(23) **Head Typing Principle:** Given a compositional environment$X$ with constituents A and B, and type assignments A: α and B: β in the type contexts for A and B respectively that clash, if A is the syntactic head in the environment, then the typing of A must be preserved in any composition rule for A and B to produce a type for X.

This means that in the case of (21), we should adjust the adjective's type, given that the noun is the relative head in the construction. Similarly, when we combine a DP in object position with a governing verb to form a VP or a VP or NP with an adjoined modifying phrase, we want the verb's categorization to affect the way the NP is interpreted, given our principle that the head of the category should win out. For subjects of a sentence, given the Head Typing Principle, we need to establish what the head of the IP is. If we take standard $\bar{X}$-syntax as our guide, it is the inflection node which introduces an event to saturate the VP, which is its complement. By Type Accommodation, the result will then have the type phys $\multimap t$. So the Head Typing Principle tells us that we must change the type of the subject DP in order for it to conform to the typing of the I'.[15] It appears as though the VP's type will win out, forcing us to change the type of the subject if there is a type clash. Finally, for coordinate constructions, the Head Typing Principle doesn't determine how types should adjust, but a slight extension of it would dictate that in coordinate constructions *both* coordinated constituents will undergo a typing change. Thus, coordinate constructions may give rise to the introduction of complex •-types, each coordinated constituent supplying one of the constituent types to the complex type.

The Head Typing Principle dictates where we should make typing adjustments, should there be conflicts involving the type of a complement and the selectional context within which it appears. But what should those adjustments be? If we go back to our metaphysical underpinnings, then what complex types allow us to do is to predicate properties of aspects of individuals. But if an aspect of a thing exists, then the thing itself must exist as well; in this respect, aspects differ from parts.

[15]Results are largely equivalent if we choose HPSG as our syntactic guide; there the verb will be the lexical head and will once again force us to change the NP's type.

So in retyping a variable to represent an aspect of a thing, we should also have a variable representing the thing itself, and we need to make sure that we link the variable representing the aspect to the variable representing the thing via our parthood relation, O-Elab. Thus, type adjustments with complex •-types typically add more information into the logical form; that is, our type inferences actually change the formula.

The last issue concerns which of the two variables we need to be the argument to the predicate. For instance, for (21), we want to say that it is the physical aspect of the book that is heavy, but we don't want this type to percolate up into the main predication. So we introduce a new variable of complex type that is the argument of the property variable and that will end up being the argument to *book*, and we close the variable typed PHYS off existentially in the lambda term for *heavy* before the adjective and the noun combine. Formally for (21), this amounts to rewriting the lambda term for the adjective as:

(24) $\lambda P \lambda y \exists z[\text{heavy}(z) \wedge \text{O} - \text{elab}(z, y) \wedge P(y)]$

where the typing context for the formula is, z : phys, $y : p \bullet i$. By adjusting the type of the argument of P, the adjectival phrase can now combine with the translation of the noun phrase, carrying the appropriate typing on the head variable. It turns out that conjoining this information with the predicate variable P in either the DP or adjectival phrase gives the quantificational closure just the right scope. If we follow our principle that the head type should be preserved on the main argument, then the proper treatment for (21) must introduce a dot typed variable within the adjectival phrase. This leads us then to posit two sorts of rules, a rule of •-Exploitation, and a rule of •-Introduction. In each case we will want to rewrite the term whose type needs to be changed in the way we've just discussed.

Type conflicts involving a complex dot type and a constituent type may occur not only when we attempt to apply a quantifier or property of properties to a property as in (21), but also when we apply a higher order property to a quantifier. We will state the rules for each of these cases in the discussion that follows.

3.4.3 The Basic Rules for Type Composition Logic

In our discussion above, we've seen how the type composition logic may change the types of terms during the construction of a logical form. Thus, our rules may call for the revision of a type context; when a type context is revised with the assignment $t : \alpha$, which we write as $c * (t : \alpha)$, then the revised context contains $t : \alpha$ and all the types of terms that involve t have their functional types changed accordingly. If t does not occur in c then $c * (t : a)$ just extends c with the assignment of a to t; i.e., $c * (t : a) = c + (t : a)$. $c(t : a)$ simply means that the type assignment c includes the assignment of a to t. We'll write $c + c'$ to denote the merging of two typing contexts or the extension of c by c'.

With this notation out of the way, we now introduce the rules for our Type Composition Logic (TCL). As usual a lambda expression denotes a functional type, i.e., a $\alpha \multimap \beta$ type. Such rules should be understood as reduction rules, thereby giving rise to equivalent term expressions. Application is defined in terms of a context, c, which provides typing assignments to both the variable in the applicand and the argument.

(25) **Application:**

$$\frac{\lambda x \phi[t], \;\; c(x : \alpha, t : \alpha)}{\phi[t/x], c}$$

In terms of the type calculus itself, application corresponds to a rule of modus ponens for $\multimap$. The type calculus of course also has lambda abstraction which corresponds to a rule of conditional proof for $\multimap$.

The contexts that accompany the rule of Application and other operations may be updated or combined, as the result of a rule being applied. We will refer to this rule as *Merging Contexts*. We will write {.} braces around the function and [.] brackets around the argument for readability.

(26) **Merging Contexts:**

$$\frac{\{\lambda x \phi, c\}[t, c']}{\lambda x \phi[t], (c + c')}$$

This is a bookkeeping rule and does not really correspond to any properties of any of the type constructors.

As with the types available in the type semi-lattice from the lexicon (cf. Pustejovsky 1995; Copestake and Briscoe 1992), we have the rule of *Type Accommodation*. Type accommodation covers what results in type unification, in which a supertype can unify with a subtype yielding the subtype as the result. This rule allows us to shift the types in the case of compatible types, which we write as $\boldsymbol{\alpha} \sqcap \boldsymbol{\beta} \neq \bot$, to the meet of the two.

(27) **Type Accommodation:**

$$\frac{\lambda x \phi[t], c(x : \alpha, t : \beta), \;\; \alpha \sqcap \beta \neq \bot}{\lambda x \phi[t], c * (x, t : \alpha \sqcap \beta)}$$

Type Accommodation corresponds to a limited strengthening of the antecedent for $\multimap$ (limited, because Type Accommodation works only when we are to trying to apply one term to another). By the axiom on the subsumption relation, Type Accommodation applies to higher functional types defined from simple types that stand in the proper subtyping relations. For instance, if a determiner of type $(e \multimap t) \multimap ((e \multimap t) \multimap t)$ takes a physical property as an argument—i.e. something of type $(plimpt)$, then the axiom on subtypes will tell us that being a physical property is a subtype of being a property, and Type Accommodation will adjust the type of the determiner to $(p \multimap t) \multimap ((e \multimap t) \multimap t)$. This will allow us then to use Application to combine the meaning of the determiner and the physical property.

3.4.4 • Types and Dot Objects

As discussed above, there are strong motivations for enriching the domain of entity types. In addition to simple types of *e* and its associated semi-lattice of subtypes, we introduced the domain of dot objects (•-types). In this section we develop the rules allowing us to exploit a •-type during composition.

Let us look again at a representative example of a type mismatch involving a dot object, where the subject is a complex type and the predicate selects for one of the constituent types. Consider the predication in (28) below.

(28) The book is heavy.

The Head Typing Principle tells us that we have to change the type of the subject DP, while the type of the VP remains unchanged. Confirmation of the Head Typing Principle comes from this implication about changing the type of the DP. To see why, suppose that the quantification in the DP in (28), and more importantly in (29) is over dot objects. Suppose that we assert (29) in a context in which some books from the library have been stolen and others borrowed.

(29) Every book is now back in the library.

Suppose in addition that there are five copies of *Anna Karenina*, six copies of *The Possessed* and four copies of *Madame Bovary* but only one copy of each has been returned. Assuming that the head of the construction is the subject DP and universally quantifying over dot objects implies that (29) is true in that case. Indeed dot objects are difficult to "count", but it seems that we can individuate at least some of them, viz. books, in terms of the individuation conditions of either constituent type. Our intuitions, however, dictate that (29) is neither ambiguous nor indeterminate but false in this context. To avoid such "sloppy" individuation conditions, we need to resort to simple types. The Head Principle dictates that we need to type the DP in (29) so that it quantifies over physical objects. If we quantify over every physical book in the library, then this will make (29) false in the context we have specified. This means that we need to shift the type of the DP so that it has a simple type by shifting the type of the head variable in the DP; i.e., if our DP looks like

(30) $\lambda P \forall x(\psi \rightarrow P(x))$

where x has a complex type, then we need to shift the type of x, and that will in turn shift the type of P to the appropriate type.

In fact if we attend to the Head Typing Principle and to the lexical categories elements of which may type their arguments as having a complex or simple subtype of *e* (the type of all entities), then we can get an idea of exactly what sort of exploitation rules we need for complex types. Lexical elements that fall under the determiner (D), inflection (I) or Adverb (Adv) categories may impose type requirements on their arguments but they do not involve complex types, as far as we have been able to determine. Hence, our rules will not apply to type conflicts between a determiner and its complement NP, or between an Inflection morpheme and its complement VP, for example. The categories whose elements do have selectional restrictions involving complex types are verbs, nouns, adjectives and prepositions. The Head Typing Principle dictates that our rules must change the types in the cases of type clash that interest us of: DPs when combining with a verb or VP, DPs when combining with a preposition, and adjectives when combining with an NP. It turns out that for all of these cases, we need only two sorts of rules: one where an argument to a DP meaning forces a shift in the DP's meaning, and one where a functor taking a DP meaning as an argument forces a shift in the meaning of the DP. The case we just considered is one where a DP argument forces a shift in the DP meaning. Below we consider cases of the second type.

Let us now formalize these observations. For an expression, ϕ, in which there is a λ-bound property variable whose type is a function from objects of a complex type to objects of some other type, we will suppose that the property variable in ϕ takes an argument x that is of complex type, something we will write as $\lambda P \phi(P(x))$. We introduce a new existentially bound variable v with type $\alpha \bullet \beta$ and replace x with v within the part of the predication, call it Δ, in ϕ that is responsible for the original typing of x. Intuitively, Δ is the main predication in ϕ. We also shift the type of x to a constituent type of the $\bullet$-type, thus changing the property variable's type to be a function of type α or β. Finally, we add the relevant parthood connection between x and v to Δ by conjoining to Δ, *O-elab*(x, v). To this end, we designate $\Delta(\phi, x)$ to be the smallest subformula of the term ϕ containing predications responsible for assigning x the type it has in ϕ and such that no predications in ϕ outside of $\Delta(\phi, x)$ impose that complex typing on x. Given this definition $\Delta(\phi, x)$ must be a formula with x free, since neither quantification nor lambda abstraction imposes any typing requirements on the variables they bind. Furthermore, $\Delta(\phi, x)$ will not include the property variable itself, since it inherits its type from its argument x, not the other way around. This allows us to make our variable substitution, to retype x, and to add the *O-elab* condition without any problem. To illustrate Δ, we look to some particular constructions. For instance, in the case of the logical form for a simple, adjectival phrase, Δ would constitute the material contributed to the lambda term from the adjective, as we saw in the previous section. In the case of a DP, Δ would be the formula in the restrictor of the generalized quantifier logical form.

We can now state this version of $\bullet$-Exploitation with a pair of substitutions, which look like the expression, $\chi\{\frac{\phi}{\psi}\}$. One other bit of notation has to do with the square brackets; they represent an application that hasn't yet taken place. That is with $P[x]$ we haven't yet applied the property that P stands for to x; similarly the lambda expression with its typing context, $[\psi, c']$ hasn't yet been integrated with the lambda expression with its context on its left. We enclose the complex expression that is to apply to $[\psi, c']$ in curly brackets to help for readability. Below $\phi(P[x])$ represents the fact that the property variable P is to apply to x in the expression ϕ, and ψ : $\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} \multimap \gamma$ represents the fact that ψ is typed either as $\alpha \multimap \gamma$ or as $\beta \multimap \gamma$. In this rule and the following rules concerning $\bullet$-types, we will assume that $\alpha \sqcap \alpha' \neq \bot, \beta \sqcap \beta' \neq \bot$.

(31) **•-Exploitation (•E):**

$$\frac{\{\lambda P\phi(P(x)),\ c(P:(\alpha\bullet\beta)\multimap\gamma)\}\left[\psi,\ c'(\psi:\begin{bmatrix}\alpha'\\ \beta'\end{bmatrix}\multimap\gamma)\right],\text{head}(\psi)}{\left\{\lambda P\phi\left[\frac{\exists v(\Delta(\phi,x)[\frac{v}{x}]\wedge\text{O}-\text{Elab}(x,v))}{\Delta(\phi,x)}\right],\ c*\left(x:\begin{bmatrix}\alpha\sqcap\alpha'\\ \beta\sqcap\beta'\end{bmatrix},v:\alpha\bullet\beta\right)\right\}[\psi,c']}$$

•-Exploitation does two things; it adds material to the logical form of the lambda term to which it applies and it also revises the type contexts to reflect a shift in the typing of some of the variables in the altered lambda term. If we look just at what happens to the type for x, •-Exploitation corresponds to something like a conjunction elimination rule for •-types, but it is more complicated than that since it forces us in reintroduce a variable of •-type. It is in fact an *ampliative* rule.[16]

Let us look at an example of •E at work. Consider the sentence in (28), where the VP in effect predicates only of the physical aspect of a $p\bullet i$.[17]

1. $[[\text{the}]]=\lambda Q\lambda P\exists x(Q[x]\wedge P[x]),\langle P,Q:e\multimap \underline{t},x:e\rangle$[18]
2. $[[\text{book}]]=\lambda v\text{book}(v),\langle v:p\bullet i\rangle$
3. $[[\text{the book}]]=\lambda Q\lambda P\exists x(Q[x]\wedge P[x]),\langle P,Q:e\multimap \underline{t},x:e\rangle\,[\lambda v\text{book}(v),\langle v:p\bullet i\rangle]$
4. As $e\sqcap(p\bullet i)=p\bullet i$, by Accommodation, which revises the typing context, we get:

   $\lambda Q\lambda P\exists x(Q[x]\wedge P[x]),\langle P,Q:(p\bullet i)\multimap \underline{t},x:p\bullet i\rangle\,[\lambda v\text{book}(v),\langle v:p\bullet i\rangle]$
5. Now we use Application and Merging Contexts to get a term for *the book*:

   $\lambda P\exists x(\text{book}(x)\wedge P[x]),\langle x:p\bullet i,\ P:(p\bullet i)\multimap \underline{t}\rangle$
6. The logical form for *is heavy*, and the interpretation in this sentence is the following: $\lambda u\text{heavy}(u),\quad\langle u:p\rangle$
7. The syntax dictates:

   $\lambda P\exists x(\text{book}(x)\wedge P[x]),\langle P:e\multimap \underline{t},x:p\bullet i\rangle\,[\lambda u\text{heavy}(u),\quad\langle u:p\rangle]$
8. By•-Exploitation:

   $\{\lambda P\exists x(\exists v(\text{book}(v)\wedge\text{O}-\text{Elab}(x,v))\wedge P[x])\,\langle v:p\bullet i,x:p\rangle\}[\lambda u\text{heavy}(u),\langle u:p\rangle]$

[16]For some cases we may have to treat the existential quantifier on v as having its force determined by the original over x. As in DRT, we would have to treat such quantifiers over x as unselective. The cases we have in mind would be those where ϕ is of the form $Qx(\psi(x),\chi(x))$, and both restrictor and nuclear scope have material that is responsible for typing x originally as being of complex type. We will not deal with this complexity here.

[17]This effectively replaces the *Dot Object Subtyping* rule, $\dot{\Theta}^{llet}$, as developed in Pustejovsky (1995) pp. 150–151.

[18]We assume for illustration purposes that the computation will accommodate the presuppositions of definiteness locally in the composition.

9. By Merging Contexts and Application,

$$\exists x(\exists v(\mathrm{book}(v) \wedge \mathrm{O}-\mathrm{Elab}(x,v)) \wedge \lambda u \mathrm{heavy}(u)[x]), \langle x : p,\ u : p,\ v : p \bullet i\rangle$$

10. By Application:

$$\exists x(\exists v(\mathrm{book}(v) \wedge \mathrm{O}-\mathrm{Elab}(x,v)) \wedge \mathrm{heavy}(x)), \quad \langle v : p \bullet i, x : p\rangle$$

The rule of •-Exploitation lets us take any modifier of a noun that would force a dot type (the adjective *readable* would be one such example) and apply it to a noun with a simple type that is the constituent of the modifier's type. We could then combine the two together to get a noun phrase of the simple type as required. Thus if we have a sentence such as (32) below:

(32) John turned off every readable screen.

our rule will produce a noun phrase that looks like the following, before the determiner meaning is applied:

(33) $\lambda x(\exists v(\mathrm{readable}(v) \wedge \mathrm{O}-\mathrm{Elab}(x,v) \wedge \mathrm{screen}(x)), \langle x : p, v : p \bullet i\rangle$

When the determiner meaning is applied, we will get a quantification over all physical screens, which is what is intuitively required.

Our rule of •-Exploitation makes the quantification over objects of the constituent types always have scope over the quantification over objects of • type. But is this right? Consider for instance, the following.

(34) Three books by Tolstoy are heavy.

Following the derivation above, we would get a logical form for this sentence according to which on a distributive reading there are three physical aspects p_1, p_2, p_3 each of which have to satisfy the formula $\exists v(\text{book by Tolstoy}(v) \wedge \mathrm{O}-\mathrm{Elab}(x,v))$, where $x : p$ and $v : p \bullet i$, and each of which are heavy. Nothing in our semantics forces the three aspects to be part of the *same* book. In fact, quite the opposite. Our semantics for O-elab makes such an interpretation incoherent, for if we have O-elab(p_1, b) and O-elab(p_2, b), $p_1 = p_2$, which contradicts the meaning of the quantifier. In our semantics of O-elab, this subformula of the logical form of (34) can only be satisfied if there is a distinct book for each distinct physical aspect. Though there is a collective reading of the DP(the three books together are heavy), our semantics precludes having a collective reading of the formula in the restrictor of the quantifier.[19] Thus, we end up predicting that (34) is true only if there are three distinct books each with its own physical aspect that is heavy. Because of the particular dependency of aspects on the substances to which they belong, there is a quantificational dependency between variables for aspects and variables ranging over the substances of which they are parts.

[19]For details on how such distributive and cumulative readings together are possible, see Asher and Wang 2003.

There is one other case of •-exploitation to consider, namely, the one where the complex type/simple type conflict occurs between an expression that has a generalized quantifier as an argument and a generalized quantifier. This is the second type of rule we alluded to above. This could occur for instance when a verb types its argument as a physical object but the noun in the complement types its argument as a complex type, say $p \bullet i$. This situation can be illustrated by the following example.

(35) John's mother burned the book on magic before he mastered it.

The verb *burn*'s object argument must be a physical object, and as the Head Typing Principle dictates, although the object DP enters the composition with type $p \bullet i$, there must be some way to coerce it into having the right type, to satisfy the typing context and thereby allow the λ-conversion from the verb to go through. The way we do this is to apply a kind of •-Exploitation on the generalized quantifier to coerce it into the right type.

Let us look at the details. In (35), we see a problem with the typing of the expressions we are trying to compose (recall that p (physical-object) in this context is a subtype of e in the semi-lattice structured domain of entities, i.e., $p \sqsubseteq \mathrm{e}$):

(36) $$\lambda \mathcal{P} \lambda w \mathcal{P}[\lambda u(\mathrm{burn}(w,u))], \langle \mathcal{P} : (p \multimap \underline{t}) \multimap \underline{t}, u : p, w : p\rangle$$
$$[\lambda P \exists x(\mathrm{book}(x) \wedge P(x)),\ \langle P : (p \bullet i) \multimap \underline{t}, x : p \bullet i\rangle]$$

Because we are not changing the sense of the predicate in any way (that is, *burn* should still mean *burn*) it is undesirable to change the type of the variable $\mathcal{P}$ over DP denotations; rather, we want to change the type of the object itself. In that case, •E won't apply directly, but we can invoke a type shifted version of it, which we call •-Exploitation^{TS} (•E^{ts}). As before, we assume $\alpha \sqcap \alpha' \neq \bot$, $\beta \sqcap \beta' \neq \bot$.

(37) **•-Exploitation:**^{TS}

$$\frac{\left\{\lambda \mathcal{P}\phi,\ c\left(\mathcal{P}\colon \left(\begin{bmatrix}\alpha' \\ \beta'\end{bmatrix} \multimap \gamma\right) \multimap \delta\right)\right\} [\lambda P \psi(P[x]),\ c'(P : (\alpha \bullet \beta) \multimap \gamma)],\ _{\mathrm{head}}(\phi)}{\{\lambda \mathcal{P}\phi,\ c\}\left[\lambda P \psi \left\{\frac{\exists v(\Delta(\psi,x)\{\frac{v}{x}\} \wedge \mathrm{O{-}Elab}(x,v))}{\Delta(\psi,x)}\right\},\ c' * \left(v : \alpha \bullet \beta, x\begin{bmatrix}\alpha \sqcap \alpha' \\ \beta \sqcap \beta'\end{bmatrix}\right)\right]}$$

The type shifted version of •-Exploitation applies to (35'), and we can now rewrite the object DP so that λ-reduction can take place, as illustrated below.

(38) $$\lambda \mathcal{P} \lambda w \mathcal{P}[\lambda u(\ \mathrm{burn}(w,u))], \langle \mathcal{P} : (p \multimap \underline{t}) \multimap \underline{t}, u : p, w : p\rangle\ [\lambda P \exists x$$
$$(\exists v(\mathrm{book}(v) \wedge \mathrm{O} - \mathrm{Elab}(x,v)) \wedge P[x]),\ \langle P : p \multimap \underline{t}, x : p, v : p \bullet i\rangle]$$

Applying *Merging* and *Application*, we get the following expression:

(39) $$\lambda w\ \lambda P\ \exists x(\exists v(\mathrm{book}(v) \wedge \mathrm{O} - \mathrm{Elab}(x,v) \wedge P[v]))[\lambda u(\ \mathrm{burn}(w,u))],$$
$$\langle P : p \multimap \underline{t}, x : p, v : p \bullet i, u : p, w : p\rangle$$

We can now continue the λ-reductions with Application to get:

(40) $\lambda w \exists x \exists v(\text{book}(v) \wedge \text{O} - \text{Elab}(x, v)) \wedge \ \text{burn}(w, v), \ \langle w : p, x : p, v : p \bullet i \rangle]$

When we apply this to the subject DP, we get the desired reading: namely, that the physical manifestation of the book has been burned, though the dot object book remains for discourse binding. Given the Head Typing Principle, we do not need any other •-Exploitation rules.

3.4.4.1 •-Introduction

Whereas •-Exploitation merely selects as an argument a constituent type of a •-type to facilitate application, sometimes predicates will force the introduction of a variable of • type. This happened in our discussion of the phrase *a heavy book*. As illustrated earlier, a verb such as *read* can also select a dot object (17a) or coerce a lower type to dot object status (17b–c).

(17) a. Mary <u>read</u> the book.
b. John <u>read</u> the rumor about his ex-wife.
c. Mary <u>read</u> the subway wall.

The mechanism for performing this shift is already implicit in our •E rule. To turn that rule into a • introduction rule, we need merely to readjust which variable is introduced and ends up being lambda bound, and which variable is existentially quantified over. Instead of existentially binding the dot-typed variable as in •E, we will existentially quantify over the constituent-typed variable, allowing the dot-typed variable to combine with its dot-typed property. This rule, •-introduction or •I, applies when the head of the construction is the argument (in the rule below the argument is ψ). Once again, we assume $\alpha \sqcap \alpha' \neq \bot$, $\beta \sqcap \beta' \neq \bot$.

(41) **•-Introduction (•I):**

$$\frac{\left\{\lambda P \phi(P[x]), c\left(P{:}\begin{bmatrix}\alpha' \\ \beta'\end{bmatrix} \multimap \gamma\right)\right\}[\psi,\ c'(\psi{:}(\alpha \bullet \beta) \multimap \gamma)], \text{head}(\psi)}{\left\{\lambda P \phi\left[\frac{\exists v(\Delta(\phi, x)\left[\frac{v}{x}\right] \wedge \text{O}-\text{Elab}(v, x))}{\Delta(\phi, x)}\right], c*\left(v{:}\begin{bmatrix}\alpha \sqcap \alpha' \\ \beta \sqcap \beta'\end{bmatrix}, x{:}\alpha \bullet \beta\right)\right\}\ [\psi,\ c']}$$

•-Introduction is needed to construct the properly typed lambda term for (2), *a heavy book*. Recall that *heavy* has a logical form $\lambda P \lambda x(\text{Heavy}(x) \wedge P(x))$ where

$x : p$ and $P : p \multimap \underline{t}$.*book* yields the term $\lambda v \text{book}(v)$ together with the typing context $v : p \bullet i$. Using $\bullet$-Introduction on the lambda term for *heavy* we get

(42) $\lambda P \lambda x \exists z((\text{Heavy}(z) \wedge \text{O} - \text{elab}(z, x) \wedge P[x]),$

where $x : p \bullet i \;\; z : p$ and $P : p \bullet i \multimap \underline{t}$.

This can now combine with the head noun *book* to give us:

(43) $\lambda x \exists z((\text{Heavy}(z) \wedge \text{O} - \text{elab}(z, x) \wedge \text{Book}(x))$

and this will combine with the determiner to give the right meaning for the whole DP.

The only other case we need to consider is where a higher order λ-abstracted variable carries the complex type and it is a head with respect to its argument. Such is the case in a sentence such as (44).

(44) John read every wall.

In this example, the verb *read* takes a DP as its object that it must coerce into a dot type. This is done through a variant of the type shifted version of $\bullet$-Introduction. It looks very similar to $\bullet$-Exploitation^{TS}, and we'll assume once again that $\alpha \sqcap \alpha' \neq \bot$, $\beta \sqcap \beta' \neq \bot$.

(45) **$\bullet$-Introduction with Type Shifting ($\bullet$I^{TS}):**

$$\frac{\{\lambda \mathcal{P}\phi,\, c(\mathcal{P}{:}((\alpha\bullet\beta)\multimap\gamma)\multimap\delta)\left[\lambda P\psi(P[x]),\, c'\left(P{:}\begin{bmatrix}\alpha\\ \beta\end{bmatrix}\multimap\gamma\right)\right],\, \text{head}(\phi)}{\{\lambda \mathcal{P}\phi,\, c\}\left[\lambda P\psi\left\{\frac{\exists v(\Delta(\psi,x)\{\frac{v}{x}\}\wedge \text{O}-\text{Elab}(v,x))}{\Delta(\psi,x)}\right\},c'*\left(x{:}\alpha\bullet\beta,v{:}\begin{bmatrix}\alpha\sqcap\alpha'\\ \beta\sqcap\beta'\end{bmatrix}\right)\right]}$$

The rule of $\bullet$-I^{TS} transforms the logical form for the DP *every wall* into:

(44') $\lambda P \; \forall x[\exists v[\text{wall}(v) \wedge \text{O} - \text{elab}(v, x)] \rightarrow P(x)], \quad \langle x : p \bullet i, \quad v : p\rangle$

The DP in (1) may now combine with the verb, while allowing the verb's argument type to win out and get the appropriate quantificational force from the DP, which is what is desired.

3.5 Conclusion

In this paper, we have outlined a type theoretic interpretation of Generative Lexicon Theory. This involved developing an extension to the lambda calculus, Type Composition Logic, with rules for exploiting and introducing complex types.

These rules suffice to handle much of the data about these types that has come to light in work on the Generative Lexicon. But we think there are many extensions to this work. Rules for complex types have already been shown to be useful in the analysis of indirect speech acts (Asher and Lascarides 2001). On the other hand, complex types and their exploitation have proved useful in reasoning about discourse structure (Asher and Lascarides 2003); verbs with a causative structure often yield complex types that support certain discourse connections. We think this work can also further be extended by extending the notion of complex types, beyond those we have considered here. For instance, a verb like *buy* may introduce in fact a complex type, in which one type of eventuality serves as a Background to the other. And the same anaphoric mechanisms for further specifying these types that we referred to earlier and discussed by Danlos (1999) might apply here:

(46) Kim sold her truck. Sandy bought it.

By merging concerns of the lexicon with those of discourse interpretation together, we can explore these hypotheses further.

Acknowledgment We would like to thank Sheila Asher, Pascal Denis, Tim Fernando, Ivan Sag, Stan Peters, David Israel, Alex Lascarides, Ann Copestake, José Castaño, Roser, Saurí, and Johanna Seibt for helpful comments on earlier drafts. This work was partially supported by NIH Grant DC03660, to James Pustejovsky.

References

Amadio, R., & Curien, P. L. (1998). *Domains and lambda calculi*. Cambridge: Cambridge University Press.

Asher, N. (1993). *Reference to abstract objects in discourse*. Dordrecht: Kluwer.

Asher N. (2004). Things and their aspects, manuscript.

Asher, N. (2011). *The web of words*. Cambridge: Cambridge University Press.

Asher, N., & Fernando, T. (1997). Effective labeling for disambiguation. In *Proceedings of the second international workshop in computational linguistics*, Tilburg, The Netherlands.

Asher, N., & Lascarides, A. (1995). Lexical disambiguation in a discourse context. *Journal of Semantics, 1*, 69–108, Oxford University Press.

Asher, N., & Lascarides, A. (2001). Indirect speech acts. *Synthese, 128*, 183–228.

Asher, N., & Lascarides, A. (2003). *Logics of conversation*. Cambridge: Cambridge University Press.

Bouillon, P. (1997). *Polymorphie et sémantique lexicale: le cas des adjectifs*. Lille: Presses Universitaires du Spetentrion.

Carpenter, B. (1992). Typed feature structures. *Computational Linguistics, 18*, 2.

Copestake, A., & Briscoe, T. (1992). Lexical operations in a unification-based framework. In J. Pustejovsky & S. Bergler (Eds.), *Lexical semantics and knowledge reperesentation*. Berlin: Springer.

Copestake, A., & Briscoe, T. (1995). Semi-productive polysemy and sense extension. *Journal of Semantics, 15*.

Crole, R. (1993). *Categories for types*. Cambridge: Cambridge University Press.

Crouch, D., & van Genabith, J. (2000). *Linear logic for linguists*, ESSLLI-00 Course material manuscript.

Danlos, L. (1999). Event coherence in causal discourses. In P. Bouillon & F. Busa (Eds.), *The syntax of word meaning*. Cambridge: Cambridge University Press.

Fodor, J., & Lepore, E. (1998). The emptiness of the lexicon: Critical reflections on J. Pustejovsky's "*The Generative Lexicon*". *Linguistic Inquiry, 29*, 269–288.

Godard, D., & Jayez, J. (1993). Towards a proper treatment of Coercion Phenomena. In *Proceeding of the 1993 European ACL.*

Groenendijk, J., & Stokhof, M. (1990). Dynamic predicate logic. *Linguistics and Philosophy, 14*.

Hendriks, H. (1993). *Flexible Montague Grammar, LP-1990-09*. Logic, Philosophy and Linguistics (LP) Series: ILLC Publications.

Hobbs, J., Stickel, M. E., Appelt, D. E., & Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence, 63*, 69–142.

Howard, W. A. (1980). The formulas-as-types notion of construction. In J. P. Seldin & J. R. Hindley (Eds.), *To H. B. Curry: Essays on combinatory logic, lambda calculus and formalism*. New York: Academic.

Jackendoff, R. (1997). *The architecture of the language faculty*. Cambridge: MIT Press.

Jackendoff, R. (2002). *Foundations of language*. Oxford: Oxford University Press.

Kamp, H., & Reyle, U. (1993). *From discourse to logic*. Dordrecht: Kluwer Academic.

Klein, E., & Sag, I. (1985). Type-driven translation. *Linguistics and Philosophy, 8*, 163–202.

Lascarides, A., & Asher, N. (1993). Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and Philosophy, 16*, 437–493.

Lascarides, A., & Copestake, A. (1995). The pragmatics of word meaning. In *Proceedings SALT V*.

Levin, B., & Hovav, M. R. (1995). *Unaccusatives: At the syntax-lexical semantics interface*. Cambridge, MA: MIT Press.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In K. Hintikka, J. Moravcsik, & P. Suppes (Eds.), *Approaches to natural language* (pp. 221–242). Dordrecht: Kluwer. (Reprinted in Thomason 1974, pp. 247–270).

Moravcsik, J. (1975). Aitia as generative factor in Aristotle's philosophy. *Dialogue, 14*, 622–36.

Moravcsik, J. (1998). *Meaning, creativity, and the partial inscrutability of the human mind*. Stanford: CSLI Publications.

Morrill, G. (1992). *Type-logical grammar*. Utrecht: Onderzoeksinstituut voor Taal en Spraak.

Nunberg, G. (1995). Transfers of meaning. *Journal of Semantics, 12*.

Partee, B., & Rooth, M. (1983). Generalized conjunction and type ambiguity. In S. Bäuerle & A. von Stechow (Eds.), *Meaning, use, and interpretation of language*. Berlin: Walter de Gruyter.

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics, 17*, 409–441.

Pustejovsky, J. (1994). Semantic typing and degrees of polymorphism. In C. Martin-Vide (Ed.), *Current issues in mathematical linguistics*. Holland: Elsevier.

Pustejovsky, J. (1995). *The generative lexicon*. Cambridge, MA: MIT Press.

Pustejovsky, J. (1998). The semantics of lexical underspecification. *Folia Linguistica, XXXII*.

Pustejovsky, J. (2001). Type construction and the logic of concepts. In P. Bouillon & F. Busa (Eds.), *The Syntax of Word Meaning*.: Cambridge University Press.

Pustejovsky, J. (forthcoming). *Meaning in context: Mechanisms of selection in language*. Cambridge: MIT Press.

Pustejovsky, J. (2011). Coercion in a general theory of argument selection. *Journal of Linguistics, 49*(6).

Pustejovsky, J., & Boguraev, B. (1993). Lexical knowledge representation and natural language processing. *Artificial Intelligence, 63*, 193–223.