Concepts, Definitions, and Inheritance

Interpreting the atoms of lexical decomposition

by

DAVID ZORNEK

Inheritance Networks and the GDIT

To begin, we will look at some examples of lexical decomposition, highlighting those aspects of each theory that are relevant to the present discussion. Each theory models something different in semantics: Jackendoff models internal representations of meaning, as a total theory of semantics, while Levin and Rappapor Havov model the semantic features of verbs that have consequences for syntax. Computational lexical resources provide lexicons of the internal semantic structure of commonly used word senses; different computational resources do it in different ways, but the basic goal of all is the same. James Pustejovsky's *Generative Lexicon* (GL) seeks to improve on traditional lexicons by decomposing words in a way that allows new word senses to be generated according to systematic polysemy, rather than being stored in the lexicon as independent entries. Regardless of which approach to lexical decomposition we look at, the general *method* of analysis—the method of composing meaning out of atoms, by means of some structural rules of composition—is the same. Our primary concern here is to understand this general method, rather than the nitty-gritty details of each individual theory.

1 Formal Thoeries of Lexical Decomposition

1.1 Jackendoff's cognitive semantics

At first glance, Ray Jackendoff seems to advocate a version of my own view. In [4], he argues that word meaning is decomposed into concepts, but his usage of the word "concept" is broader than mine. For Jackendoff, a concept is any mental representation, and it includes subjective representations of categories (which is, more or less, what cognitive scientists mean when they talk about concepts), subjective representations of propositions, mental representations of individual objects, and perhaps even the cognitive structural rules which Jackendoff believes to be the source of basic grammatical structure for natural languages.

Jackendoff's notion of concept is too broad to do the work that will be required here. The connection I draw between semantic and conceptual content will rely on empirical results pertaining to the way in which concepts (in the cognitive scientist's sense) are organized. These results have not been seen for other mental representations.

Moreover, his notion of concept is entirely subjective; that is, Jackendoff's concepts explicitly have no important connection to the external world. This is largely do to his view on the nature of mind, which is little more than a more readable rendition of Kant's *Transcendental Aesthetic* [4, Ch. 2]. Or faculty of sensation is somehow roused into action by external stimuli. Sensations are used as a kind of paint by the mind, which creates a subjective picture or "projected world" (cf. Kant's phenomenal world) by applying the paint in accordance with concepts, which act as a kind of blueprint. It is concepts, not the external world, that determines what we see in the projected world. And the projected world is *all* we see, all our words can possibly refer to. Since one of our goals here is to explain how words hook up with the world, Jackendoff's version of internalism will not do.

Nevertheless, Jackendoff's theory is decompositional and therefore exhibits the same features of other decompositional theories that will be important here. Where decomposition is concerned, it is type "concepts" (in the Jackendoff-ian sense) that are taken as atomic. The meaning of a lexical item (or a portion of its meaning, at least) is given in a *feature structure* in which types are assigned to different "features" or aspects of the word's meaning. The basic form of one of Jackendoff's feature structures is:

$$\begin{bmatrix} \textbf{event, thing, place,}\ldots \\ \text{token/type} \\ F(\langle \text{Entity}_1, \langle \text{Entity}_2, \langle \text{Entity}_3 \rangle\rangle\rangle) \end{bmatrix}$$

The top listing of the feature structure specifies an *ontological category* under which the lexical item falls. Ontological categories provide a partition of the ways in which our cognitive architecture allows us to represent word meaning. The middle listing gives an immediate *inheritance relation* (explained below) in which the word participates (normally by specifying a type from which the given word inherits), and the third and final listing gives the argument structure of the word, i.e. the types of arguments that the word can take, which will change depending on the sense in which the word is being used. For example, take the following typed feature strucutures for "novel" and "begin":

$$\begin{bmatrix} \textbf{thing} \\ \textbf{book} \\ F(\langle \textbf{property} \rangle) \end{bmatrix}$$

$$\begin{bmatrix} \textbf{event} \\ \textbf{transition} \\ F(\langle \textbf{animate\_object}, \langle \textbf{event} \rangle \rangle) \end{bmatrix}$$

"Begin" takes a primary argument of type **animate_object** (since only animate objects can begin actions) and a secondary argument of type **event** (since only events can be begun).[1] "Novel" might take arguments of type **property**, as it is used in the sentence "This novel is long"; but, in the sentence "The novel is on the table," the argument type is **place**. These are differences between senses of "novel," which are largely determined by the syntactic structure of the sentence and a grammatical conceptual structure inherent in human cognition. The details of how this works are interesting, but not relevant to the present project.

The ontological category and immediate inheritance relation, however, *are* relevant to the present project, so it is useful at this point to explain them.

Jackendoff's types are organized according to the *inheritance relation* $\sqsubseteq$.[2] Inheritance is reflexive and transitive, but not symmetric. Given two

[1]We might quibble over whether these type assignments are the *right* ones for "begin." It's not important here whether I accurately convey the specific types involved in the semantics of "begin," and these will suffice at least to get a handle on how Jackendoff's feature structures are supposed to work, which is the real goal here.

[2]Jackendoff does not call the relation inheritance; I have modified his terminology in order to make my own terminology consistent throughout this paper. He calls the relation an "IS-A" relation or a "type-token" relation, both of which are fairly common

atoms α and β such that $\alpha \sqsubseteq \beta$, we say that α inherits from β. Or, α is a *subtype* of β. If there is a third atom γ such that $\beta \sqsubseteq \gamma$, then what the inheritance relation tells us is that $\alpha \sqsubseteq \gamma$ as well. If we let $\mathbf{x}$ be a member of any domain to which the types can apply, we write $\mathbf{x}{:}\alpha$ to indicate that $\mathbf{x}$ is of type α. By inheritance, $\mathbf{x}{:}\alpha \vdash \mathbf{x}{:}\beta \vdash \mathbf{x}{:}\gamma$.

The broadest type, of which all other types are subtypes is the type **entity**. Beneath **entity**, there is a set of *ontological categories*, none of which inherit from each other, but all of which inherit from **entity**. These are **thing**, **event**, **state**, **action**, **place**, **path**, **property**, and **amount**. This sort of system might remind the reader of Aristotle's *Categories*, which might be regarded as the earliest exemplar of a type-inheritance theory of lexical semantics. Every lexical item will fall under one of the ontological categories.

As will be seen below, there will be a fundamental connection between these parts of the feature structure and semantic content. Before spelling out this connection explicitly, however, we will look at a few more examples of lexical decomposition.

1.2 Levin and Rappaport Havov's verb decomposition

In [5], Levin and Rappaport Havov offer a decompositional semantics for verbs specifically. In particular, they model those aspects of verb semantics that have consequences for which grammatical arguments must be, might be, or cannot be syntactically realized in a sentence. Like Jackendoff, they hold the view that there is some definite set of ontological categories that exist as part of the semantic theory, i.e. there is some universal set of atoms from which all verb meaning is constructed. In fact, they accept Jackendoff's ontological categories, *plus* an additional set of fixed ontological categories that inherit from Jackendoff's **event** type. Among these additional ontological categories are **cause**, **become**, and **act**; in another paper [6], they give a long list of others that are commonly seen in theories of verb decomposition. The subtypes of Jackendoff's other ontological categories, which

alternatives.

are not given any special ontological status, are left explicitly open-ended; that is, they can be whatever and however many in number are required to give a semantic analysis of all verbs in a natural language.

Some of the ontological categories in this additional set are called *predicates*, and they will determine the structure of the semantic entry, while the others are called *constants*, which provide semantic content and act as arguments for the predicates.

We cannot give a semantic entry for "novel" in their system, since it deals only with verbs, but we can give an entry for "begin":

$$[[x\ \mathbf{act}]\mathbf{cause}[x\langle \mathit{ACTION}\rangle]],$$

where ACTION is a variable standing for any subtype of the constant **action**. If some agent x *begins* to perform some *action*, then x *acts* to *cause* x to be in a state of *action.* For example, the event structure for the event described by (2) "Maria began reading" is

$$[[\text{Maria }\mathbf{act}]\mathbf{cause}[\text{Maria}\langle\mathbf{reading}\rangle]].$$

Again, we have a set of atoms that provide semantic content, i.e. the constants. And again the use of these atoms relies on an inheritance relation: inheritance from **action** is a constraint placed on the content of arguments for "begin."

2 Computational Lexical Resources

Lexical decomposition is not purely theoretical, but can also be seen in computational lexical resources used in technological implementations such as automated translation, corpus annotation, treebanking, automated sentence parsing, which are useful for application in artificial intelligence, speech recognition, language learning software, etc. Of course, the place of lexical decomposition in computational resources is an artifact of the fact that such resources are based on formal linguistic theories.

2.1 VerbNet

VerbNet, a lexical resource created and managed by Karin Kipper-Schuler, Martha Palmer, and others at University of Colorado, is currently the largest on-line verb lexicon for English. Lexical entries are organized into verb classes based on the predicates of Levin and Rappaport Havov[3], and a type inheritance system is provided for content-bearing constants (or atoms). Since VerbNet uses Levin and Rappaport Havov's formal system, an example of decomposition for "begin" has already been given.

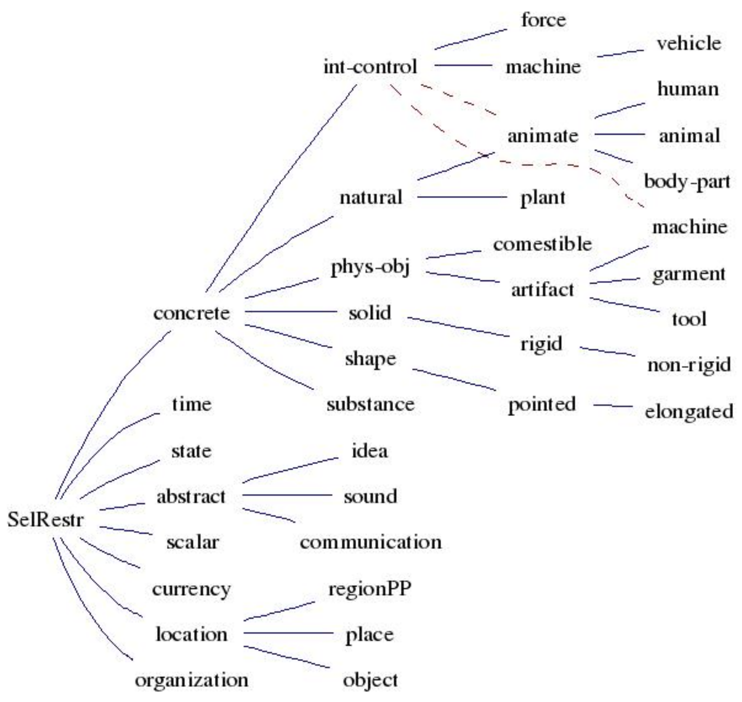


Figure 1: A sample type inheritance hierarchy from VerbNet

[3]In VerbNet, the predicates are called *thematic roles*, following [3], but they come down to the same thing as Levin and Rappaport Havov's predicates. I call them predicates because I have already introduced this term and VerbNet is explicitly based on their theory.

2.2 FrameNet

FrameNet is another very well-known computational lexicon currently managed by Collin Baker at the International Computer Science Institute, UC Berkeley. In FrameNet, lexical entries are organized according to *frames*, which are *composed* out of *type assignments* given to various features of a word (what features are is not relevant here). As in VerbNet, the types are organized in an inheritance hierarchy. FrameNet-style annotations are very widely used internationally, in many different languages, and provide the basis for much automated translation technology. Where VerbNet was explicitly based on the decompositional theory of Levin and Rappaport Havov, FrameNet is based on the linguistic theory of Charles Fillmore [2].[4] Since lexical entries in FrameNet are very large (and in fact, each sense of a word requires it's own very lengthy entry), I do not demonstrate the decomposition of "novel" and "begin" here, but the reader is referred to https://framenet.icsi.berkeley.edu/fndrupal/index.php?q=luIndex, where any lexical entry, including "novel" and "begin" can be viewed.

2.3 Linguistic Knowledge Builder

Ann Copestake's *Linguistic Knowledge Builder* (LKB) is not itself a lexicon, but is an engineering environment that can be used to create computational grammars and lexicons for natural languages. It is implemented in Common Lisp, an object-oriented programming language, and has a fundamental dependence on type inheritance for all of its operations. All of the salient features of LKB will come out in the discussion of James Pustejovsky's *Generative Lexicon* below, so I will say no more about it here. I bring it up at this time only because it is a computational resource and therefore is most appropriate to this section.

[4]Fillmore's work on frame semantics was a predecessor to much work on lexical decomposition, but it was not itself a decompositional theory, so it is not explained here.

2.4 Computational resources, in general

There are many other computational decompositional lexicons out there—far too many to list here. They universally involve some form of type inheritance. This is, in part, due to the fact that they are programmed using object-oriented languages, and in part due to the fact that they tend to be based on formal theories of lexical decomposition that involve type inheritance. It might be thought that object-oriented languages are useful for computational lexical resources because of the fact that they involve the same type inheritance relation that is already present in the formal theories. However, we might also think that there is some deeper cognitive structure that exhibits inheritance behavior in the way agents represent semantic content that gives rise to the presence of inheritance hierarchies in both formal theories and computational resources. If this turns out to be true, then the view I am presenting here should suffice as an explanation of why object-oriented languages are so useful in computational linguistics. Even if it turns out not to be true, i.e. that object-oriented languages are useful only because of the presence of inheritance in formal theories, then we can still inquire after why inheritance consistently shows up in the formal theories.

Regardless of why computational resources rely so heavily on inheritance, they all involve one fundamental question: Should we be *lumpers* or *splitters* in defining our atoms? Lumpers like to have fewer atoms that are less specific, but apply more broadly, while splitters like to have finer-grained, more specific atoms that apply in comparatively fewer lexical entries. Currently, the debate amounts mostly to matters of preference. However, in identifying what the atoms *are*, I will have pointed toward some empirical phenomenon that we can look toward when deciding how finely or coursely grained atoms should be.

3 Moral of the story: Lexical Semantic Content and Inheritance

3.1 Lexical Semantic Content

In any decompositional theory, there is a structural component and a content-providing component. Levin and Rappaport Havov [6] have made this exact point for a particular class of decompositional theories, called Lexical Conceptual Structures (LCS). Although LCSs are theories of verb decomposition, their comments apply to lexical decomposition in general. Many words might have the same general structure for their lexical entries, only differing in their semantic content, which is represented in the form of semantic atoms. For example, given the structural template $[\mathbf{become}[y\langle \textit{RES-STATE}\rangle]]$, where *RES-STATE* is a variable ranging over the domain of resulting states, we have the following three lexical entries:[5]

1. *dry*: $[\mathbf{become}[y\langle \mathbf{dry}\rangle]]$
2. *widen*: $[\mathbf{become}[y\langle \mathbf{wide}\rangle]]$
3. *dim*: $[\mathbf{become}[y\langle \mathbf{dim}\rangle]]$

It is obvious that "dry," "widen," and "dim" all have very different meanings, yet the structure of these lexical entries is the same, different only in the atom that occurs in the argument for **become**. The part of each lexical entry that is idiosyncratic to each word is the content provided by the atom. Note that the same point applies to both Jackendoff and Pustejovsky, since *all* of their lexical entries have the same structure.

In lexical decomposition, semantic content is always inherited from some basic atom. But this is only possible if atoms have content, so it makes sense to inquire after the content of atoms.

Either uninterpreted atoms have content or they do not. David Lewis [7] has given a compelling argument in the domain of sentential decomposition that uninterpreted atoms do not have content, which extends easily

[5] "Lexical entry" is a general term that applies to any structure or item that is used to convey the meaning of a word in the lexicon of a semantic theory.

to lexical decomposition. A decompositional theory provides a set of atoms or "markers," which are essentially a lexicon for the theory. The compositional rules provide a syntax according to which we combine the atoms into some sort of structure representing the semantics of whatever linguistic unit is in the domain of the theory (in Lewis's case, the linguistic units under consideration are sentences; in ours, they are words). "But," Lewis writes,

> we can know the Markerese translation of an English sentence without knowing the first thing about the meaning of the English sentence: namely, the conditions under which it would be true. Semantics with no treatment of truth conditions is not semantics. Translation into Markerese is at best a substitute for real semantics, relying either on our tacit competence (at some future date) as speakers of Markerese or on our ability to do real semantics at least for the one language Markerese.

Proponents of decompositional theories are able to get by without providing content to their atoms precisely because they and their audience are both native speakers of some natural language. The atoms of lexical decomposition are, *invariably*, represented by the same signs as words in natural language. Since we are all able to understand the words represented by these signs, in virtue of the fact that we are all fluent speakers of some natural language, we are able to make sense of decompositional lexical entries even without a "real semantics" for English or Markerese. Theorists of lexical decomposition rely on fluency to supply content to atoms where none is given within the theory. One of the main goals of this project is precisely to indicate one way in which we can do "real semantics" for Markerese, at the same time situating our decompositional theories within the broader context of "real semantics" for natural language.

If we follow Lewis in thinking that uninterpreted atoms do not have content, but want to retain the decompositional approach, then we will want to compose word meaning out of some atom whose content we understand independently of the decompositional theory. Absent some interpretation that provides content to atoms from without, it is difficult to see how we might obtain such understanding. Any theory of lexical decomposition is necessarily unable to non-circularly provide content to its atoms. Within the theory, the meanings of atoms can only be decomposed into other atoms; theories of lexical decomposition are *only* theories of lexical decomposition. But if the meanings of atoms are given decompositionally, then either: (a) they aren't *actually* atoms of the theory; (b) we will ultimately bottom out with the meanings of some atoms given in terms of themselves; or (c) we have a "decompositional circle," e.g. α is defined in terms of β and γ, β is defined in terms of δ and ϕ, and so on, until we reach some atom defined in terms of α. In any case, we cannot form a coherent view of how a decompositional theory can provide meaning to its own atoms. The current external source of content for atoms is their identification with English words, which retrieve content from fluency of native speakers. But if this is the actual source of content for the atoms, then we have an obvious circularity: linguistic items supply content to atoms, which supply content to linguistic items.

Nevertheless, linguists and philosophers alike (many of them at least) seem to agree that decompositional theories play a crucial role in semantics as a whole.[6] It is not the case that lexical decompositional theories are completely useless until and unless we can find ourselves in possession of a convincing account of the content of its atoms. Any lexical decompositional theory comprises only part of a total science of lexical semantics. Decompositional theories can tell us a great deal about the structural component of meaning. But, we take ourselves to be talking *about* something when we use words, and except when what we are actually talking about is atoms, to say that atoms are the providers of the content component of meaning doesn't tell us what our words are *about.* When I say, "The sky is clear today," I've said nothing at all about atoms. I've said something about the sky. Whatever sort of thing meaning is, *aboutness* is a major component of

[6]In the domain of sentential semantics, it is wholly uncontroversial that decomposition is the right approach. There is some controversy over lexical decomposition, and die-hard opponents of lexical decomposition may reject what is said here. Justifying lexical decomposition is outside the scope of this paper; I seek to fortify lexical decomposition, given that we already agree on the appropriateness of the decompositional approach to word meaning.

it. We might say that "The sky is clear today" is about the sky because the word "sky" refers to the sky. This is on the right track, but some words have meaning without referring to anything, e.g. unicorn, goblin, etc., so reference cannot be a necessary condition of meaning. As a guiding intuition, we might say: When a word has reference, its reference plays an important role in meaning. In cases where a word has no reference, some story will need to be told about the meanings of words, but I think that what I will argue is compatible with any plausible story that might be offered.

My proposal is that we can find some cognitive representation to interpret atoms in order to provide a non-circular external source of meaning. It seems to me to be intuitively plausible that cognition and meaning go hand-in-hand. Understanding a language involves having a certain cognitive relation to well-formed sentences of that language. But we can say a bit more. Our sense of *aboutness* is undoubtedly a mental phenomenon, and therefore we should look inward for an explanation of this sense. Moreover, by identifying atoms with some cognitive representation, we might be able to account for *aboutness* while simultaneously screening ourselves off from reference in a way that will leave room for meaning without reference where necessary. This picture should be able to satisfy referentialists such as Lewis, by allowing our cognitive structures to mediate between words and reference while also being palatable to subjectivists such as Jackendoff, who wish to give a theory of internal semantics in which linguistic units have no meaning beyond what is present in the mind.

Two problems with the current situation have already been mentioned: there is no coherent, non-circular way for decompositional theories to provide content to their own atoms without relying on prior understanding of the meanings of words; without content for the atoms, we are unable to account for *aboutness*. Also, we should keep in mind that atoms are supposed to be carriers of semantic content; therefore, any viable interpretation must exhibit some behavior that reflects fundamental observations about semantic content. At this point, we are able to identify three criteria that a successful interpretation of the atoms must meet. I will take the current project to have been successful when I have established an interpretation that

(i) accounts for *aboutness*,

(ii) is not itself dependent on linguistic meaning, and

(iii) exhibits some fundamental property that parallels semantic content.

3.2 Inheritance

Inheritance has been shown up in every instance of lexical decomposition covered here. In fact, I am aware of no instance of lexical decomposition—even outside of what has been included above—that does not involve an inheritance order over the atoms. The existence of atoms is essential to lexical decomposition; decomposition, after all, is decomposition into atoms. But inheritance seems to show up mostly by accident. There is nothing about decomposition *per se* that necessitates inheritance. We can at least conceive of decomposing word meanings into atoms that are not related by inheritance. This seems to point toward the hypothesis that there is something fundamental about semantic content that exhibits inheritance behavior, apart from the role that semantic content plays in lexical decomposition. In fact, as will be seen in Chapter 3, there is a kind of cognitive representation that exhibits inheritance behavior and which is intuitively plausible as a provider of lexical semantic content—namely, concepts.

4 James Pustejovsky's Generative Lexicon

James Pustejovsky's theory of lexical decomposition, the Generative Lexicon (GL) [8] will provide a case study for developing the conceptual interpretation of semantic atoms, so it will be useful to look at GL in some level of detail.

5 Sense Enumeration Lexicons and Their Problems

GL is a theory of lexical semantics based on LKB [9], which is in turn based on Bob Carpenter's logic of typed feature structures [1]. GL is a model of word meaning in which word senses are generated by the mechanisms underlying systematic polysemy. Historically, the simplest and most direct means of handling polysemy has been to allow the same word to be listed multiple times in the lexicon, with each listing storing a different semantics for the word. A precise characterization of this sort of *Sense Enumeration Lexicon* SEL is

> A lexicon L is a *Sense Enumeration Lexicon* if and only if for every word w in L, having multiplie senses $s_1, \ldots, s_n$ associated with that word, then the lexical entries expressing these senses are stored as $\{w_{s_1}, \ldots, w_{s_n}\}$. [8, p. 34]

There are three main classes of arguments against the use of SELs:

1. *The Creative Use of Words.* It is simply an observed fact about language that new senses of words can be created in very systematic ways whenever we need them. Even when we have not encountered a sense s_i of some word w (or even when w_{s_i} did not exist prior to the time at which we encounter it), we understand it properly as w_{s_i} provided certain conditions are met. SELs can only explain the understanding of word senses that already exist in the lexicon.

2. *The Permeability of Word Senses.* SELs require that word senses are atomic, distinct structures, but this does not seem to match up with observations about how words are used. Word senses are usually related to one another, and it is often unclear whether a sense s_i is actually distinct from s_j. If senses are generated by schematic rules, the question of whether two senses are actually distinct becomes irrelevant to our understanding of the polysemy. A listing of separate

word senses cannot do justice to the semantic richness of word meanings, which can capture complex relations between senses, as well as multiple distinct uses.

3. *Difference in Syntactic Forms.* SELs will tend to store separat semantic entries for each syntactic form of a word, but it is arbitrary to create separate word senses just because the word can participate in different kinds of sentences. For example, SELs would store separate lexical entires for the use of "forget" in factive (*forget that*), non-factive (*forget to*), embedded question (*forget where*), and other syntactic structures. It is simpler to simply have one semantic entry for "forget" and to define a comparitively small number of generative rules which can be applied to any word in different syntactic contexts.

These inadequacies are spelled out in greater detail by Pustejoveky [8, Ch. 4], and for a fuller discussion of them, the reader is referred to his book.

6 Typed Feature Structures

In the GL, the semantics of a lexical item α is represented as a quadruple $\langle \mathcal{A}, \mathcal{E}, \mathcal{Q}, \mathcal{I} \rangle$. Each component is discussed below.

6.1 The argument structure $\mathcal{A}$

Argument structure is the best-understood of the components and is regarded as a minimal specification of lexical semantics, although it is far from adequate as a complete characterization of the semantics of any lexical item. In much research, the argument structure is regarded as the most important determinant of verb meaning in particular. There are four main types of argument discussed by Pustejovsky:

1. *True Arguments.* True arguments are necessarily made explicit in syntax, e.g. in the sentence "*Josh* is driving the automobile." The sentence would not be well-formed if no driving agent were given explicitly.

2. *Default Arguments.* Default arguments are logically implied by the qualia structure, but are not necessary for syntactic well-formedness, e.g. "Josh is driving *the automobile*." Ignore, for the moment, the technical language about qualia structure; it will be made clearer later. The real point here is that the semantics of the verb "is driving" logically entail that it is an automobile being driven, regardless of whether this is explicitly stated in the sentence, and the sentence is well-formed regardless of whether the default argument is stated, i.e "Josh is driving" is a well-formed sentence, and it is understood in the use of the verb "is driving" that he is driving an automobile. Although the default argument is not necessary for the syntactic well-formedness of the sentence, it is necessary for its logical well-formedness and is therefore implied even when not stated. Explicit expression of default arguments is, for the most part, optional.

3. *Shadow Arguments.* Shadow arguments are similar to default arguments in that they are not necessarily stated in syntax. However, they differ from default arguments in that they are expressable only under certain circumstances. Without going into a level of detail unecessary for our present purposes, the fundamental difference is that, while the default argument is implied by the qualia structure of a lexical item, a shadow argument is actually incorporated into the semantics of a lexical item. As a result, a shadow argument is only expressible when it is further narrowed by means of a subtyping operation. For example, we would not say, "Mary buttered her toast with butter." That would be redundant. We might, however, say "Mary buttered her toast *with Land o'Lakes butter*."

4. *True Adjuncts.* True adjuncts are syntactically optional, like default arguments, but are not tied to the semantics of any lexical item in the expression. Nevertheless, they do contribute to the semantics of the expression as a whole, by providing some situational context, e.g. "Josh is driving the automobile *South on I-71*."

The grammatical arguments of the lexical item "build" are encoded into a list structure ARGSTR in the following manner:

$$\begin{bmatrix} \textbf{build} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{animate\_individual} \\ \text{ARG}_2 = \textbf{artifact} \\ \text{D-ARG}_1 = \textbf{material} \end{bmatrix} \\ \ldots & \end{bmatrix}$$

Although for verbs, the features listed in ARGSTR will coincide with the grammatical arguments of the item being defined, this will not always be the case. It should become obvious why these entries are called arguments, even where they are not grammatical arguments, later.

6.2 The extended event structure $\mathcal{E}$

Event structure can get rather complicated, due to the fact that events can often be broken down into sub-events or phases; for instance, in the sentence "I drove to Kentucky," the event "drove" can be broken down into the process of driving to Kentucky and the state of being at Kentucky, which exist in a relation of strict ordering. The event structure EVENTSTR consists in a typing of all subevents $e_1, \ldots, e_n$ as well as a typing of the relation structure RESTR. There are three types of events: **process**, **state**, and **transition**. In order to avoid unnecessary complexity, I do not give a full listing of all acceptable relation types that might be used. For the time being, at least, I limit myself to the strict ordering relation $<_\alpha$[7]. Also, since event structures are not of central importance at this stage of the project, I do not offer definitions for **process**, **state**, and **transition**, instead leaving it to common intuition to provide understanding. In some cases, a certain subevent will be regarded as more significant than the others; this semantic information can be captures by also specifying a HEAD, which is the subevent that is regarded as more important than other subevent; it is possible to have mul-

[7]Later, I will also make use of the relation structure $< \circ$, which indicates that events may be strictly ordered or overlapping; $\circ$ is the symbol for overlap.

tiple heads. Headedness plays a major role in the polysemy of verbs, which will only be given a cursory treatment in this project. The following is an example of the event structure for the word "build":

$$\begin{bmatrix} \textbf{build} & \\ \text{EVENTSTR} = & \begin{bmatrix} \mathrm{E}_1 = \textbf{process} \\ \mathrm{E}_2 = \textbf{state} \\ \text{RESTR} = <_\alpha \end{bmatrix} \\ \ldots & \end{bmatrix}$$

6.3 The qualia structure $\mathcal{Q}$

Pustejovsky's qualia structure is borrowed from Aristotle's four modes of explanation.[8] The qualia structure QUALIA is constituted by typing the four following features:

1. *Constitutive.* The material properties of an object; the relation between an object and its proper parts.
    (a) Material
    (b) Weight
    (c) Parts and component elements
2. *Formal.* The property that distinguishes an object from others within a larger genus to which the object belongs.
    (a) Orientation
    (b) Magnitude
    (c) Shape

[8]We should not read too much into the word "qualia;" as near as can be ascertained, it does not take the usual meaning we see in Philosophy of Mind. I am uncertain why Pustejovsky chose this word, although he gives a citation to Moravcsik (1973,1975) as an influence on his understanding of Aristotle's four modes of explanation, which might shed some light on this choice if it turns out to be important to the discussion. In any event, Moravcsik will be examined in due time, but at this point, I have not yet looked this far into the genealogy of qualia structure.

   (d) Dimensionality
   (e) Color
   (f) Position

3. *Telic.* The purpose or function of an object.
   (a) Purpose for which an object was created by some agent.
   (b) A built-in function or aim toward which the natural activities of an object point.

4. *Agentive.* The origin of an object or factors involved in its "bringing about."
   (a) Creator
   (b) Artifact
   (c) Natural kind[9]
   (d) Causal Chain

As an example, consider the following QUALIA structure for "novel":

$$\begin{bmatrix} \textbf{novel} & \\ \ldots & \\ \text{QUALIA} = & \begin{bmatrix} \text{CONST} = \textbf{narrative} \\ \text{FORMAL} = \textbf{book} \\ \text{TELIC} = \textbf{reading} \\ \text{AGENT} = \textbf{writing} \end{bmatrix} \end{bmatrix}$$

This simple listing of qualia values "tells us nothing about how a particular lexical item denotes, however. For example, although a novel's purpose is the activity of reading and it comes about by someone writing it, we do not want to claim that the comon noun *novel* actually denotes such activities." Pustejovsky's solution to this problem is to bind qualia values by

[9]Pustejovsky lists this, but I am hesitant to do the same, due to some confusions and/or skepticism I have pertaining to natural kinds, which are off-topic from the present discussion.

regarding them as argument-taking expressions.[10] The improved QAULIA structure is:

$$\left[\begin{array}{ll}\textbf{novel} & \\ \mathrm{ARG}_1 = & \textbf{x:book} \\ \mathrm{QUALIA} = & \left[\begin{array}{l}\mathrm{CONST} = \textbf{narrative(x)} \\ \mathrm{FORMAL} = \textbf{x} \\ \mathrm{TELIC} = \textbf{read(y,x)} \\ \mathrm{AGENT} = \textbf{write(z,x)}\end{array}\right]\end{array}\right]$$

The above structure is equivalent to the λ-expression

$$\lambda x\big[\textbf{novel}(x) \wedge \mathrm{CONST} = \textbf{narrative}(x) \wedge \ldots \wedge \mathrm{AGENT} = \lambda z[\textbf{write}(z,x)]\big].$$

6.3.1 The CONSTITUTIVE quale.

The CONSTITUTVE quale can specify not only the relation between a referent object and its parts, as in the CONSTITUTIVE quale **narrative(x)** given in the typed feature structure for the word "novel" above, but also the relation between a referent object and some other object, of which it is a part. For instance, consider the partial typed feature structure for "hand," which involves the relation **part_of**:

$$\left[\begin{array}{ll}\textbf{hand} & \\ \mathrm{ARG}_1 = & \textbf{x:limb} \\ \mathrm{QUALIA} = & \left[\begin{array}{l}\mathrm{FORMAL} = \textbf{x} \\ \mathrm{CONST} = \textbf{part\_of(x,y:body)}\end{array}\right]\end{array}\right]$$

The other main function that can be performed by the CONSTITUTIVE quale is that it can specify the material out of which an object is made, where this is especially relevant to the semantics of a word. For example, in a typed feature structure for "doubloon," we might give CONST=**gold**(x).

[10]It is not obvious at this point why this move should be valuable. I believe that this point is clearest when considering the polysemous behavior of novel's so I defer demonstration and explanation until later.

6.3.2 The FORMAL quale

Let us consider each of the qualia in turn. The FORMAL quale can take structures of either of the following forms:

1. *Simple Typing.* The FORMAL quale defines the sortal typing of the argument.

2. *Complex (Dotted) Typing.* The FORMAL quale defines a relation between arguments of different types.

The above is an example of simple typing. An argument is defined in the parameter ARG_1=**x:book**; this says that not only is **x** an argument to be used within the scope of the lexical entry **novel**, but that it is an argument of type **book**. (Expressions of the form $\alpha : \tau$ are defined to mean that α has type τ.) Notice that, since **x** has already been typed in ARG_1, there is no need to give a sortal typing for the argument in FORMAL directly by writing FORMAL=**book(x)**. The advantage to typing at the ARG_1-level is efficiency in the LKB system, which is not our main purpose here. If one has no aims of efficiency in LKB, we might also just have ARG_1=**x** and FORMAL=**book(x)** with the same semantic effect. (Note that this is a somewhat simple example; in other cases, not considered here, we may find that refusal to apply sortal typing to arguments in the ARGSTR greatly increases the complexity of our representations. I will follow the convention given here wherever possible.)

Now, we shall consider an entry for **book**, which involves complex typing:

$$\begin{bmatrix} \textbf{book} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:information} \\ \text{ARG}_2 = \textbf{y:phys\_obj} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \textbf{information} \cdot \textbf{phys\_obj} \\ \text{FORMAL} = \textbf{hold(y,x)} \\ \text{TELIC} = \textbf{read(e,w,x}\cdot\textbf{y)} \\ \text{AGENT} = \textbf{write(e}',\textbf{v,x}\cdot\textbf{y)} \end{bmatrix} \end{bmatrix}$$

The entry **information·phys_obj** in the above is included because the dotted type mud be accessible with the QUALIA structure for well-formedness; otherwise the argument **x·y** is untyped. We will make use of dotted types later when discussing the polysemy of nominals more closely later.

6.3.3 The TELIC quale

Generally, the TELIC quale corresponds to Aristotle's final cause, but in terms of specifics, the TELIC quale may take a number of different formal representations, each corresponding to a different mode of final causation. Pustejovsky only considers two modes in detail, but I do not think he considers this to be an exhaustive list of possible modes for the TELIC quale.

1. *Direct Telic.* The direct object of some action. Consider the following partial typed feature structure for the word "beer":

$$\begin{bmatrix} \textbf{beer} & \\ \text{ARG}_1 = & \textbf{x:liquid} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \textbf{x} \\ \text{TELIC} = \textbf{drink(e,y,x)} \end{bmatrix} \end{bmatrix}$$

   The TELIC quale is given as the three-place relation **drink(e,y,x)**, which we interpret as "y drinks x" (where the action of drinking has an event type structure laid out in **e**). The referent of the word being defined is intended as the direct object of the action represented by the two-place relation giving in the TELIC quale. This is contrasted with:

2. *Purpose Telic.* An object of facilitation for some action. Consider the following partial typed feature structure for the word "knife":

$$\begin{bmatrix} \textbf{knife} & \\ \text{ARG}_1 = & \textbf{x:tool} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \textbf{x} \\ \text{TELIC} = \textbf{cut(e,x,y)} \end{bmatrix} \end{bmatrix}$$

> Here again, the TELIC quale is given as a three-place relation, with the direct object as the third argument of the relation, but the referent of the word being defined is not given as the third argument of the relation. Although the purpose of the knife is cutting, it is not the knife itself which we cut. Rather, the knife is the thing we use to cut some other object, or in some grammatical structures, the knife itself is the thing which does the cutting.

These differences in structure of the TELIC quale are what allow for the following linguistic alternations between sentences which describe the same causative structure between agents, objects and instruments:[11]

1. The knife cut the bread.
2. Josh cut the bread with the knife.

6.3.4 The AGENTIVE quale

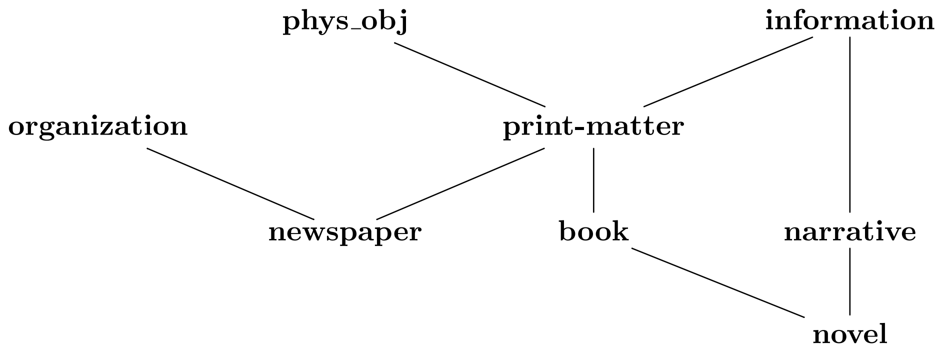
The AGENTIVE quale is given as a two- or three-place event relation, with the referent of the word being defined as the final argument of the relation. For nominals with a simply typed FORMAL quale, we will typically use a two-place relation; in nominals with a complex typed FORMAL quale and for verbs, we will typically use a three-place relation, since the event structure then becomes relevant to how we understand the word and the an event type may place restrictions on typing. Mostly, the technical issues surrounding three-place AGENTIVE qualia will be outside the scope of this project.

6.4 The lexical inheritance structure $\mathcal{I}$

The lexical inheritance structure $\mathcal{I}$ is a lattice which defines what is to be considered as a type and the relations between types. Following Carpenter's Logic of Type Feature Structures, the fundamental relation between types is inheritance. The inheritance lattice is a partial ordering $\sqsubseteq$ over types and

[11]The formal mechanism used to explain the alternation will be *type coercion*, which is discussed later.

we say that α *inherits from* β just in case $\alpha \sqsubseteq \beta$. If α inherits from β, then α is a *subtype* of β, i.e. for any object $\mathbf{x}{:}\alpha$, it is also the case that $\mathbf{x}{:}\beta$. The entire logic of typed feature structures is a spelling-out of the consequences of the inheritance relation. We do not need to be too concerned with the details of the logic here and can simply refer to Carpenter's work where necessary. A sample inheritance lattice is given below:



Neither Pustejovsky nor Carpenter commits us to any specific inheritance structure, nor even a specific set of types. Carpenter, in particular, is explicit that, so far as his logic is concerned, any set of types and any inheritance structure will do, provided they possess a certain set of very general characteristics. We need not be too concerned with Carpenter's conditions; it will be difficult to cook up any set of types or inheritance relations which do not meet them, but which are also plausible for use in the GL. A more important issue arises from the fact that Pustejovsky does not consider the question of whether there are some inheritance structures that are better than others for modeling language. Indeed, Pustejovsky doesn't say much about $\mathcal{I}$ at all, other than what it is. One of my hypotheses is that there is a deep relation between the hierarchical organization of concepts and inheritance structures and that, by giving an appropriate model of concepts which is sufficient to explain hierarchy, we can get greater insight into $\mathcal{I}$. Importantly, it is my hope that we will get some answer to the question of whether any $\mathcal{I}$ at all will do, or whether only some $\mathcal{I}$s are viable candidates as part of a linguistic structure in the GL.

The examples given above are only very simple examples, chosen (mostly from Pustejovsky) precisely because of the simple way in which they illustrate specific structures. In fact, typed feature structures can be used to represent quite complex lexical semantic information and can even be used to compose a semantics for an entire phrase out of its constituent words. For example, given the following typed feature structures for "bake" and "cake,"

$$\left[\begin{array}{ll} \textbf{bake} & \\ \text{EVENTSTR} = & \left[\begin{array}{l} \text{E}_1 = \mathbf{e_1{:}process} \\ \text{HEAD} = \mathbf{e_1} \end{array}\right] \\ \text{ARGSTR} = & \left[\begin{array}{ll} \text{ARG}_1 = \ \boxed{1} & \left[\begin{array}{l} \textbf{animate\_ind} \\ \text{FORMAL} = \textbf{phys\_obj} \end{array}\right] \\ \text{ARG}_2 = \ \boxed{2} & \left[\begin{array}{l} \textbf{mass} \\ \text{FORMAL} = \textbf{phys\_obj} \end{array}\right] \end{array}\right] \\ \text{QUALIA} = & \left[\begin{array}{l} \textbf{state\_change} \\ \text{AGENT} = \mathbf{bake\_act}(\mathbf{e}_1, \boxed{1}, \boxed{2}) \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{ll} \textbf{cake} & \\ \text{ARGSTR} = & \left[\begin{array}{l} \text{ARG}_1 = \mathbf{x{:}food\_ind} \\ \text{D-ARG}_1 = \mathbf{y{:}mass} \end{array}\right] \\ \text{QUALIA} = & \left[\begin{array}{l} \text{CONST} = \mathbf{y} \\ \text{FORMAL} = \mathbf{x} \\ \text{TELIC} = \mathbf{eat(e_2,z,x)} \\ \text{AGENT} = \mathbf{bake\_act(e_1,w,x)} \end{array}\right] \end{array}\right]$$

we can construct a typed feature structure for "bake a cake":

$$\begin{bmatrix} \textbf{bake a cake} & \\ \text{EVENTSTR} = & \begin{bmatrix} \text{E}_1 = \mathbf{e}_1\textbf{:process} \\ \text{E}_2 = \mathbf{e}_2\textbf{:state} \\ \text{RESTR} = <_\alpha \\ \text{HEAD} = \mathbf{e}_1 \end{bmatrix} \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = & \boxed{1}\begin{bmatrix} \textbf{animate\_ind} \\ \text{FORMAL} = \textbf{phys\_obj} \end{bmatrix} \\ \text{ARG}_2 = & \boxed{2}\begin{bmatrix} \textbf{artifact} \\ \text{CONST} = \boxed{3} \\ \text{FORMAL} = \textbf{phys\_obj} \end{bmatrix} \\ \text{D-ARG}_1 = & \boxed{3}\begin{bmatrix} \textbf{material} \\ \text{FORMAL} = \textbf{mass} \end{bmatrix} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \textbf{create} \\ \text{FORMAL} = \textbf{exist}(\mathbf{e}_2, \boxed{2}) \\ \text{AGENT} = \textbf{bake\_act}(\mathbf{e}_1, \boxed{1}, \boxed{3}) \end{bmatrix} \end{bmatrix}$$

The reader is advised not to do too much work in trying to make sense of the above structures. The point here was simply to demonstrate the level of complexity that can can be captured by typed feature structures, and there are some aspects of the above demonstration that are unexplained. In particular, the inclusion of **create** in the QUALIA structure for "bake a cake" and the sudden appearance of the **artifact** type in the ARGSTR structure; these are due to the fact that "bake a cake" can be used in different sense, e.g. the act of creating a cake, the process of baking, etc., and explanation of how they are yielded by the rules of co-composition is not important at present.

7 Type Coercion

The main mechanic of polysemy for nominals is *type coercion.* As we have seen, the typed feature structure for verbs will demand that the verb take arguments of a specific type, e.g. "build" takes a subject of type **animate_individual** and an object of type **artifact**. However, we can unproblematically supply verbs with arguments that differ from their designated types. For instance, consider the following sentences:

1. Josh began *a book.*
2. Josh began *reading a book.*
3. Josh began *to read a book.*

In each of the above sentences, the verb "began" takes an argument of a different type; the argument type in (1) is **artifact** in (1), while in (2) and (3), we see an argument with different sorts of event structures. One approach has been to regard "began" as the polysemous word; when used in different senses, "began" takes arguments of different types. By and large, these approaches have had to rely on some sort of sense-enumerative lexicon, which is exactly what Pustejovsky's GL is intended to avoid (and I agree with his reasons for being dissatisfied with sense-enumerative lexicons). In the GL, the sense of the verb is the same in all three sentences, and it is the sense of the argument that changes to accommodate the type expected by the verb. But this is only possible under certain conditions; in particular, the expected type needs to be available from the QUALIA structure of the argument, and there must be an available type-shifting operation available to act on the expression.[12]

Two type-shifting operations are considered by Pustejovsky (although these examples are not considered an exhaustion of all possible type-shifting operations): subtype coercion and true complement coercion. We will consider the simplest of these first, which is subtype coercion.[13] Take the sen-

[12]As near as I can tell, the available type-shifting operations are just the systematic polysemies. Pustejovsky does not consider non-systematic polysemy in his book, but it may be that non-systematic polysemy can occur when no type-shifting operation is available.

[13]Mandy Simons has pointed out that Pustejovsky's example of subtype coercion might actually be more complicated than the lets on. The nominal "Honda" is considered as a class noun and a subtype of the class noun "car." But there are also other senses in which "Honda" is a proper noun referring to the organization, make of car, etc. It is not clear from Pustejovsky's treatment of "Honda" how the alternation between class noun and proper noun might be involved in the polysemous behavior of the word, especially since the sense in which "Honda" is a class noun (and the reason for regarding is as a

tence “Josh drives a Honda to work.” The semantics for “drive” yield the following typed feature structure:

$$\left[\begin{array}{ll} \textbf{drive} & \\ \text{EVENTSTR} = & \left[\begin{array}{l} \text{E}_1 = \mathbf{e_1 process} \\ \text{E}_2 = \mathbf{e_2 process} \\ \text{RESTR} =< \circ_\alpha \end{array}\right] \\ \text{ARGSTR} = & \left[\begin{array}{l} \text{ARG}_1 = \mathbf{x{:}human} \\ \text{ARG}_2 = \mathbf{y{:}vehicle} \end{array}\right] \\ \text{QUALIA} = & \left[\begin{array}{l} \text{FORMAL} = \mathbf{move(e_2,y)} \\ \text{AGENT} = \mathbf{drive\_act(e_1,x,y)} \end{array}\right] \end{array}\right]$$

According to the above structure, “drive” expects an object argument of type **vehicle**. But, if we accept the following typed feature structure for “Honda”:

$$\left[\begin{array}{ll} \textbf{Honda} & \\ \text{ARGSTR} = & \left[\begin{array}{l} \text{ARG}_1 = \mathbf{x{:}car} \end{array}\right] \\ \text{QUALIA} = & \left[\begin{array}{l} \text{FORMAL} = \mathbf{x} \\ \text{TELIC} = \mathbf{drive(e,y,x)} \\ \text{AGENT} = \mathbf{create(e,Honda\text{-}Co,x)} \end{array}\right] \end{array}\right]$$

the argument provided by “Honda” is of type **car**, not of type **vehicle**. However, we also have available the structure for type **car**:

$$\left[\begin{array}{ll} \textbf{car} & \\ \text{ARGSTR} = & \left[\begin{array}{l} \text{ARG}_1 = \mathbf{x{:}vehicle} \end{array}\right] \\ \text{QUALIA} = & \left[\begin{array}{l} \text{FORMAL} = \mathbf{x} \\ \text{TELIC} = \mathbf{drive(e,y,x)} \\ \text{AGENT} = \mathbf{create(e,z,x)} \end{array}\right] \end{array}\right]$$

subtype of “car”) seems to be derivative of these other senses. These complications should not hinder the task at hand, however, which is simply to give an illustration of subtype coercion, so we ignore them for the time being.

Observe that the above structures imply that **Honda** $\sqsubseteq$ **car** $\sqsubseteq$ **vehicle**. By the inheritance relation, then, any object of type **car** is of type **vehicle** also. Next we equip ourselves with the subtype coercion operation:

$$\frac{\alpha : \sigma_1, \qquad \Theta[\sigma_1 \leq \sigma_2] : \sigma_1 \to \sigma_2}{\Theta[\sigma_1 \leq \sigma_2](\alpha) : \sigma_2}$$

I do not wish to dedicate too much time at present to explaining the formalism of the operation itself, largely since I take it to be not too difficult to interpret, at least on the level of generality required here, without explanation. In essence, what the rule says is that "given an expression of type α of type σ_1, which is a subtype of σ_2, there is a coercion possible between σ_1 and σ_2, which changes the type of α in this composition, from σ_1 to σ_2." [8, 114].

The second form of type coercion we will look at is *true complement coercion*. This is a more dramatic form of type coercion, which rather than simply making explicit a typing that the inheritance relation already supplies implicitly, causes a strict shifting from one type into another type in the absence of an inheritance relation between the two types. True complement coercion is the form of type coercion involved in the uses of "began" above. A typed feature structure for "begin" is:

$$\begin{bmatrix} \textbf{begin} & \\ \text{EVENTSTR} = & \begin{bmatrix} \text{E}_1 = \mathbf{e}_1\textbf{transition} \\ \text{E}_2 = \mathbf{e}_2\textbf{transition} \\ \text{RESTR} =< \circ_\alpha \end{bmatrix} \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:human} \\ \text{ARG}_2 = \mathbf{e}_2 \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \mathbf{P(e_2,x)} \\ \text{AGENT} = \textbf{begin\_act}(\mathbf{e}_1,\mathbf{x},\mathbf{e}_2) \end{bmatrix} \end{bmatrix}$$

Here, we do not pay much attention to the relation given under the FORMAL quale. It is not relevant to an understanding of true complement coercion. All we need be concerned with here is that the direct object of “begin,” is typed as an event, i.e. ARG$_2$=$\mathbf{e}_2$. But, if we look at the typed feature structure of “book,”

$$\begin{bmatrix} \textbf{book} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:information} \\ \text{ARG}_2 = \textbf{y:phys\_obj} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \textbf{information} \cdot \textbf{phys\_obj} \\ \text{FORMAL} = \textbf{hold(y,x)} \\ \text{TELIC} = \textbf{read(e,w,x}\cdot\textbf{y)} \\ \text{AGENT} = \textbf{write(e}'\textbf{,v,x}\cdot\textbf{y)} \end{bmatrix} \end{bmatrix}$$

we see that referents of “book” carry the dotted type **information·phys_obj**. Now, without going into too much detail about the behavior of dotted types, which would be too far afield from this general overview here (although there will be interesting things to say about this topic farther along in the project), I will simply state that, in the case of dotted types, a word may be regarded as either type, depending on context. However, neither **information** nor **phys_obj** is an event. There are two event types available in the QUALIA structure, however: the TELIC quale **read** and the AGENTIVE quale **write**. The availability of these event types enables “begin” to take “book” as an argument. In the current case, the usual way to read “Josh began a book” in most contexts will be to shift the type of “book” to **read**, although there are some contexts in which we might shift the type to **write**. In particular, if we are in the midst of a discussion about Josh’s work as an author, it might be natural to understand the sentence in this way. How we choose between two viable candidates for type coercion is a more complicated question to which I have no answer at present. However, the basic structure of true complement coercion should at least be clear from what has been said here.

I have slightly misrepresented the details of true complement coercion here, mostly for matters of simplicity. In fact, the FORMAL quale of “begin” is a relation unifying a human subject with an event into a proposition, and the TELIC (or AGENTIVE) quale of “book” may be read as a proposition involving some (untyped) subject; it is this propositional reading which enables us to use "book" as a stand-in for an entire proposition. The details get rather gory and technical, however, and I am still working toward full understanding myself. True complement coercion appears to be a rather complicated form of polysemy, and and rather than diving in to such turbulent waters headfirst, it may be prudent to first consider the inner workings of some simpler forms of polysemy, which are not discussed by Pustejovsky. In particular, we will begin by looking at:

1. Count/Mass alternations, e.g. *lamb.*
2. Container/Containee alternations, e.g. *bottle.*
3. Figure/Ground reversals, e.g. *door, window.*
4. Product/Producer diathesis, e.g. *newspaper, Honda.*
5. Plant/Food alternations, e.g. *fig, apple.*
6. Process/Result diathesis, e.g. *examination, merger.*
7. Place/People diathesis, e.g. *city, New York.*

References

[1] Bob Carpenter. *The Logic of Typed Feature Structures.* Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1992.

[2] C. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Converence on the Origin and Development of Language and Speech*, 280:20–32, 1976.

[3] R. Jackendoff. *Semantic interpretation in generative grammar.* MIT Press, Cambridge, 1972.

[4] Ray Jackendoff. *Semantics and Cognition.* The MIT Press, Cambridge, MA, 1983.

[5] B. Levin and M. Rappaport Havov. Building verb meanings. In M. Butt and W. Geuder, editors, *The Projection of Arguments: Lexical and Compositional Factors.* CSLI Publications, Stanford, CA, 1998.

[6] B. Levin and M. Rappaport Havov. Lexical conceptual structure. In K. von Heusinger, C. Maienborn, and P. Portner, editors, *Semantics: An International Handbook of Natural Language Meaning.* Mouton de Gruyter, Berlin, 2008.

[7] David Lewis. General semantics. *Synthese*, 22(1):18—67, 1970.

[8] James Pustejovsky. *The Generative Lexicon.* The MIT Press, 1998.

[9] Antonio Sanfillipo. LKB encoding of lexical knowledge. In Ted Briscoe, Valeria De Paiva, and Ann Copestake, editors, *Inheritance, Defaults, and the Lexicon*, Studies in Natural Language Processing, pages 190—222. Cambridge University Press, Cambridge, 1993.