PUSTEJOVSKY'S GENERATIVE LEXICON

DAVID ZORNEK

1. Sense Enumeration Lexicons and Their Problems

Pustejovsky's Generative Lexicon (GL) is a theory of lexical semantics based on Ann Copestake's Linguistic Knowledge Builder (LKB), which is in turn based on Bob Carpenter's logic of typed feature structures. The main phenomenon the GL is intended to explain is 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 $\{w_{s_1}, \ldots, w_{s_n}\}$. [**?**, 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 [**?**, Ch. 4], and for a fuller discussion of them, the reader is referred to his book.

2. 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.

2.1. 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.

2.2. **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 $<_\alpha$[1]. 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 multiple 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":

[1]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.

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

2.3. **The qualia structure $\mathcal{Q}$.** Pustejovsky's qualia structure is borrowed from Aristotle's four modes of explanation.[2] 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
   (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[3]
   (d) Causal Chain

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

[2]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.

[3]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.

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

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 regarding them as argument-taking expressions.[4] The improved QAULIA structure is:

$$\begin{bmatrix} \textbf{novel} & \\ \text{ARG}_1 = & \textbf{x:book} \\ \text{QUALIA} = & \begin{bmatrix} \text{CONST} = \textbf{narrative(x)} \\ \text{FORMAL} = \textbf{x} \\ \text{TELIC} = \textbf{read(y,x)} \\ \text{AGENT} = \textbf{write(z,x)} \end{bmatrix} \end{bmatrix}$$

The above structure is equivalent to the λ-expression

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

2.3.1. *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**:

$$\begin{bmatrix} \textbf{hand} & \\ \text{ARG}_1 = & \textbf{x:limb} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \textbf{x} \\ \text{CONST} = \textbf{part\_of(x,y:body)} \end{bmatrix} \end{bmatrix}$$

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

[4]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.

especially relevant to the semantics of a word. For example, in a typed feature structure for “doubloon,” we might give CONST=**gold**(x).

2.3.2. *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.

2.3.3. *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:[5]

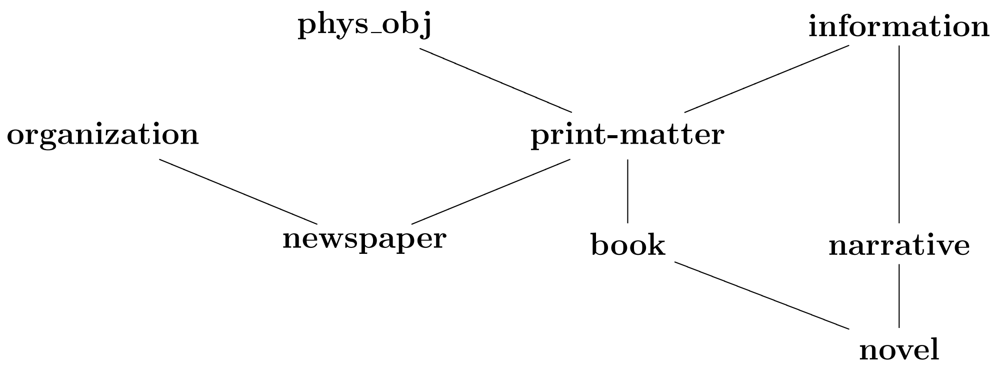
(1) The knife cut the bread.
(2) Josh cut the bread with the knife.

2.3.4. *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

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

type may place restrictions on typing. Mostly, the technical issues surrounding three-place AGENTIVE qualia will be outside the scope of this project.

2.4. **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 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,”

$$\begin{bmatrix} \textbf{bake} & \\ \text{EVENTSTR} = & \begin{bmatrix} \text{E}_1 = \textbf{e}_1\textbf{:process} \\ \text{HEAD} = \textbf{e}_1 \end{bmatrix} \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = & \boxed{1} \begin{bmatrix} \textbf{animate\_ind} \\ \text{FORMAL} = \textbf{phys\_obj} \end{bmatrix} \\ \text{ARG}_2 = & \boxed{2} \begin{bmatrix} \textbf{mass} \\ \text{FORMAL} = \textbf{phys\_obj} \end{bmatrix} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \textbf{state\_change} \\ \text{AGENT} = \textbf{bake\_act}(\textbf{e}_1, \boxed{1}, \boxed{2}) \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{cake} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:food\_ind} \\ \text{D-ARG}_1 = \textbf{y:mass} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{CONST} = \textbf{y} \\ \text{FORMAL} = \textbf{x} \\ \text{TELIC} = \textbf{eat}(\textbf{e}_2\textbf{,z,x}) \\ \text{AGENT} = \textbf{bake\_act}(\textbf{e}_1\textbf{,w,x}) \end{bmatrix} \end{bmatrix}$$

we can construct a typed feature structure for “bake a cake”:

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

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.

3. 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.[6]

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

[6]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.

these first, which is subtype coercion.[7] Take the sentence "Josh drives a Honda to work." The semantics for "drive" yield the following typed feature structure:

$$\begin{bmatrix} \textbf{drive} & \\ \text{EVENTSTR} = & \begin{bmatrix} \text{E}_1 = \textbf{e}_1\textbf{process} \\ \text{E}_2 = \textbf{e}_2\textbf{process} \\ \text{RESTR} =< \circ_\alpha \end{bmatrix} \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:human} \\ \text{ARG}_2 = \textbf{y:vehicle} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \textbf{move(e}_2\textbf{,y)} \\ \text{AGENT} = \textbf{drive\_act(}\text{e}_1\textbf{,x,y)} \end{bmatrix} \end{bmatrix}$$

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

$$\begin{bmatrix} \textbf{Honda} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:car} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \textbf{x} \\ \text{TELIC} = \textbf{drive(e,y,x)} \\ \text{AGENT} = \textbf{create(e,Honda-Co,x)} \end{bmatrix} \end{bmatrix}$$

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

$$\begin{bmatrix} \textbf{car} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG}_1 = \textbf{x:vehicle} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORMAL} = \textbf{x} \\ \text{TELIC} = \textbf{drive(e,y,x)} \\ \text{AGENT} = \textbf{create(e,z,x)} \end{bmatrix} \end{bmatrix}$$

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:

[7]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 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.

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

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." [**?**, 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.*