1 Introduction

This is a review of the book *The Generative Lexicon* by James Pustejovsky [Pustejovsky, 1995]. Rather than mainly being a critical comment, it is geared towards introducing its main ideas to the reader. It is a book describing a concrete approach to lexical semantics. It differs from the traditional approach where each lexical entry represents one word sense. Instead, it constitutes a way of deriving a potentially infinite number of senses from finite resources. This is possible by means of generative devices which work on the lexical structures, using the context to generate a full sense representation. The book also presents the logical problem of polysemy, as well as limitations associated with the traditional approach. Given that background, the semantics of nominals and causation is studied from a generative lexicon point of view.

This report will mainly focus on the parts of the book that deal with the computational lexicon aspects, leaving out some of the theoretical background and support. The first part will present the formalism, constituted by the type system and the generative devices. The second part will show how the generative lexicon can be used in practice.

2 Overview

Lexical semantics is the study of how and what the words of a language denote. Traditionally, the lexicon has been treated as a static set of word senses, tagged with various linguistic information. Each entry represents a distinct sense corresponding to a distinct lexical item. This type of lexicon is referred to as a *Sense Enumeration Lexicon*. JP lists a number of theoretical and practical problems with this technique, as part of the motivation for the need of a generative lexicon.
1. The creative use of words in new contexts. It is possible to create new senses of a word when put in a new context. A new entry has to be added to a SEL every time this happens. Consider some distinct senses for the adjective *good*: a) *buy a good umbrella*, b) *J was looking for a good meal* and c) *J is a good teacher*. Each of these would require a separate entry in a SEL. When new contexts appear, such as *good children* or *good weather*, new entries are needed.

2. Word senses are not atomic definitions but overlap and make reference to other senses of the word. Consider a) *John baked the potatoes* and b) *Mary baked a cake*. In the former, the object undergoes a change-of-state whereas in the latter it is created. To have two separate lexical entries for *fry* seems unintuitive as there is an overlap of meaning.

3. A SEL makes no difference between contrastive and complementary polysemy. Contrastive polysemy means that multiple, unrelated meanings are associated with a single word. Complementary polysemy means that multiple, related meanings are associated with a word. Logical polysemy is a form of complementary polysemy describing systematically related senses within the same lexical category.

4. A single word sense can have multiple syntactic realisation. An example is *forget*: a) *let’s not forget where*, b) *he forgot to factor one thing into his plans*, c) *friends who forget the password*. In a SEL, these would correspond to separate senses.

The view of lexical ambiguity in natural language is characterised as *weakly polymorphic*. As opposed to the traditional organisation of active functors and passive complements, this view regards all lexical items as semantically active with a richer typed semantic representation. Semantic operations of various polymorphic subclasses, like lexically determined type changing, operate under well-defined constraints (see section 2.1.2). As all lexical items are considered semantically active, JP argues that it does not suffice to merely study verbs.

### 2.1 The Semantic Type System

This section will describe the generative lexicon. It consists of a representation of lexical information and operations which work on such

---

1. *The judge asked the defendant to approach the bar.*
2. *Example of a Product/Producer alternation: The newspaper fired its editor vs. John spilled coffee on the newspaper.*
structures. Combined, they are able to represent a potentially infinite number of senses for each lexical entry, as new senses may be generated in new contexts from the existing lexical entries.

The representation, with its four types of arguments, and the operations will be further explained in the examples (see section 3).

2.1.1 Representation

The structures are recursive and compositional, supporting an explanatory view of semantic modelling. They are strongly typed feature structures, allowing internal references, e.g. as the result of unification operations. The types are ordered in a directed acyclic graph instead of a traditional type hierarchy. Each type definition may contain not only an inheritance specification stating its super type, but also feature structures. Like in object oriented programming languages, the feature structures (attributes) of all super types are quietly present in a typed lexical structure (object).

The complete argument structure of a lexical type involves four levels of representation:

1. Argument structure: specifies the number, type and syntactic realisation of the logical arguments.

2. Event structure: defines any STATE, PROCESS and TRANSITION events involved. It may also have sub eventual structures.

3. Qualia structure: defines any FORMAL, CONSTITUTIVE, TELIC and AGENTIVE roles.

4. Lexical inheritance: as mentioned above, this describes how a type fits into the type lattice. The structures also allow for a kind of polymorphic inheritance, where a structure inherits from more than one super type. The resulting type is called a lexical conceptual paradigm, LCP. This is used to combine multiple senses of logical polysemy like the one mentioned in list item 3 above (see previous page).

2.1.2 Generative Devices

The operations that are used in composing lexical items are called generative devices. They connect the different levels of semantics described above. The three most important types of devices are:

1. Coercion, in the most general sense, is something that takes place where the expected complement does not have the right type. This may happen anywhere in the four levels of representation, where
some complementary information is searched for. When this is the case, the target will be coerced into the right type, by sequentially stepping up the type graph to more general types. As soon as the complement requirement is met, coercion stops. If the top is reached, a type error is happens.

2. Co-composition is used when multiple elements within a phrase behave as functors, forming new, non-lexical senses for the words. Constraints in all elements are satisfied, rather than having one active functor matching elements that are passive.

3. Selective binding is where a lexical item operates on a subpart of a phrase, without changing types.

3 Examples

Here are some examples to illustrate how the generative lexicon works in practice:\footnote{The book does not describe a working system, but rather a theory. However, in many cases JP illustrates the lexical entries by borrowing notation from working systems. In this sense, the examples are practical.}

As explained above, coercion is an operation that will force an argument to the correct type, by searching its super nodes. Consider the following example of type coercion:

(1) Mary drives a Honda to work.

Drive expects an argument that is of the vehicle type:

\[
\begin{align*}
\text{drive} \\
\text{EVENTSTR} & = \begin{bmatrix} E_1 = & e_1: \text{process} \\
& E_2 = & e_2: \text{process} \\
& \text{RESTR} = & < o_\infty \end{bmatrix} \\
\text{ARGSTR} & = \begin{bmatrix} \text{ARG1} = & x: \text{human} \\
& \text{ARG2} = & y: \text{vehicle} \end{bmatrix} \\
\text{QUALIA} & = \begin{bmatrix} \text{FORMAL} = & \text{move}(e_2,y) \\
& \text{AGENTIVE} = & \text{drive}_{act}(e_1,x,y) \end{bmatrix}
\end{align*}
\]

A short explanation. \(< o_\infty \) represents exhaustive ordered overlap which lexicalises two basically simultaneous sub events, where \( e_1 \) starts before \( e_2 \). The process event of being in motion, \( e_1 \), is defined by \( \text{FORMAL} = \text{move}(e_2,y) \). The deep syntactic object in motion is \( \text{ARG2} = y: \text{vehicle} \). Process event \( e_2 \) is defined as an act of driving the vehicle \( y \), performed by human \( x \). Honda is defined as follows:
There is a type conflict. ARG2 of \textit{drive} is of type \texttt{vehicle}, and ARG1 of \textit{Honda} is \texttt{car}. Type coercion will however climb the type ladder to look for the vehicle type. The type \texttt{car} is defined as follows, and at this type level the check succeeds, as ARG1 is \texttt{vehicle}.

\[
\begin{align*}
\textit{car} & \quad \text{ARGSTR} = [\text{ARG1} = \texttt{x:vehicle}] \\
& \quad \text{QUALIA} = [\text{QUALIA} = 2, \text{FORMAL} = \texttt{x}, \text{TELIC} = \textit{drive}(e,y,x), \text{AGENTIVE} = \textit{create}(e,\textit{Honda-co},x)]
\end{align*}
\]

Another example of coercion is \textit{true complement coercion}. Consider the propositional attitude verb \textit{believe}, which has two arguments, \texttt{[human]} and \texttt{[prop]}. It should be able to handle constructions like:

\begin{enumerate}
\item Mary believes that he left.
\item Mary believes him to have left.
\item Mary believes the book.
\item Mary believes John.
\end{enumerate}

\textit{Book} is defined as follows:

\[
\begin{align*}
\textit{book} & \quad \text{ARGSTR} = [\text{ARG1} = \texttt{x:info}, \text{ARG2} = \texttt{y:physinfo}] \\
& \quad \text{QUALIA} = [\text{QUALIA} = 2, \text{FORMAL} = \texttt{hold}(x,y), \text{TELIC} = \textit{read}(e,w,x,y), \text{AGENTIVE} = \textit{write}(e',w,x,y)]
\end{align*}
\]

Note that the definition covers both the way it is typically used (read), and the way it is produced (write). Using a lexical conceptual paradigm, it inherits both from \texttt{info} and \texttt{physobj}. The reason for this is that \texttt{book} has the sense of a container of information, and the sense of a physical object. A special coercion operation called \textit{type pumping} performs the selection of the right inheritance branch when this entry is composed with another entry. In example (2c) above, the second \texttt{[prop]} argument accepts \texttt{book} as follows:
4 Concluding Comments

This report has given a brief overview of the generative lexicon, along with some examples of how lexical items are encoded and combined to generate sense.

The book describes a large number of cases of polysemy, and how those fit into the generative lexicon. Most of these have been left out from this report in order to not get into too much detail. They are quite important to anyone who wants to get a good understanding of the full coverage of the work. What has also been left out is the syntactic realisation information that exists in the representation structure.

With well-founded arguments, Pustejovsky points out many limitations in the traditional computational lexicon. The most important are the creative use of words in new contexts, that word senses are not atomic definitions and that SELs are not useful for complementary polysemy. However, as with most manual or possibly semi-automatic processes, the compilation of a generative lexicon may be problematic. The entries will reflect the language understanding of the person who inserts them into the lexicon. Knowledge derived from corpora is not prone to suffer from this potential flaw. On the other hand, corpora derived knowledge might be difficult to refine to the extent necessary to make it useful in a generative lexicon.

References