Lexical Structure as Generalizations over Descriptions
Mary Dalrymple (King’s College London), Ron Kaplan (PARC), Tracy Holloway King (PARC)

A primary goal of generative syntactic theory is to identify generalizations about classes and subclasses of lexical items, and in doing so to explore and characterize the structure of the lexicon. A word like *bakes* belongs to several lexical classes: it is a third-person, singular, present-tense verb which can be either transitive or intransitive. It shares some of these properties with a verb like *cooked*, and others with a verb like *appears*.

Linguistic theories have adopted different views as to how lexical generalizations should be captured. Early theories viewed the lexicon as “a kind of appendix to the grammar, whose function is to list what is unpredictable and irregular about the words of a language” (Kiparsky, 1982). With the advent of constraint-based, nontransformational theories like LFG, this view of the lexicon changed. Bresnan (1978) observed that the effect of many transformations is better captured in terms of lexical redundancy rules: for example, the active and passive forms of a transitive verb are related by lexical rules rather than by syntactic transformations. On this view, the lexicon and the rules relating lexical items become a prime locus of syntactic generalizations.

One of the first proposals for explicitly representing lexical structure was made by Flickinger (1987), who represents the lexicon as a hierarchy of word classes. Each class represents some piece of syntactic information: the word *bakes* belongs to the third-person singular present-tense class (like *appears*, *cooks*, and so on), the transitive/intransitive class (like *baked*, *eat*, and so on), and to other classes as well. Classes may be subclasses of other classes, or may partition other classes along several dimensions. Subsequent work within HPSG has built on this view. The lexicon in HPSG is represented as a type hierarchy, with more specific types inheriting information from less specific but related types. These views state the relation between types and their subtypes either as a relation between structures or in terms of operations on such structures.

Work within LFG, on the other hand, has not appealed to typed feature structures to encode lexical generalizations. In contrast to HPSG, LFG encodes lexical generalizations not in terms of relations between structures, but in terms of relations between descriptions of structures. An LFG functional description – a collection of equations – can be given a name, and this name can be used to stand for those equations in lexical descriptions. In computational treatments, these named descriptions are generally referred to as templates. The use of templates allows for lexical generalizations to be captured. Template definitions can refer to other templates; thus, a template hierarchy can be drawn to represent inclusion relations between these named LFG descriptions. Importantly, however, the relation depicted in such a diagram is not between structures, but between pieces of descriptions that recur across the lexicon.

In this paper, we will present a fragment of a template hierarchy and show how it can be used in the definition of lexical constraints. We begin with a simple lexical entry for the verb *bakes*:

(1)  
\[
\text{bakes V \{ (↑ PRED)=′bake\{}(SUBJ,OBJ)′ | (↑ PRED)=′bake\{}(SUBJ)\}}
\] 
\[
↑ \text{TENSE}=\text{PRES}
\] 
\[
↑ \text{SUBJ PERS}=3
\] 
\[
↑ \text{SUBJ NUM}=\text{SG}
\]

This lexical entry contains information that is shared by other verbs. We would like to use the templates PRESENT and 3SG to encode this information, associated with the appropriate functional information:

(2)  
\[
\text{PRESENT} = (↑ \text{TENSE})=\text{PRES}
\]
\[
\text{3SG} = (↑ \text{SUBJ PERS})=3
\]
\[
(↑ \text{SUBJ NUM})=\text{SG}
\]

The template name PRESENT names the functional description (↑ TENSE)=PRES, and similarly for 3SG. In fact, we can further subdivide the functional description 3SG in terms of the following template definitions:
(3)  a.  3PersonSubj = (↑ subj pers)=3  b.  SingSubj = (↑ subj num)=sg

We would also like to provide a general template that can be used with all verbs that can be either transitive or intransitive. We can do this by using a parametrized template that takes a single argument providing the Pred specification:

\[(4) \text{Transitive-or-Intransitive}(P) = \{(↑ \text{Pred})='P'\langle \text{subj},\text{obj}\rangle \mid (↑ \text{Pred})='P'\langle \text{subj}\rangle\}\]

When this template is used, whatever argument is given to the template will be used as the Pred specification for the lexical entry. For the verb \textit{bakes}, the template will be called as in (5).

\[(5) @(\text{Transitive-or-Intransitive bake})\]

Arguments to parametrized templates can represent any part of an f-structure description: attributes as well as values can be parametrized.

Here, too, we can define \text{Transitive-or-Intransitive}(P) in terms of more basic templates:

\[(6) \text{Transitive-or-Intransitive}(P) = \{ @(\text{Transitive P}) \mid @(\text{Intransitive P}) \}\]

\[
\begin{align*}
\text{Transitive}(P) &= (↑ \text{Pred})='P'\langle \text{subj},\text{obj}\rangle' \\
\text{Intransitive}(P) &= (↑ \text{Pred})='P'\langle \text{subj}\rangle'
\end{align*}
\]

We could also have chosen to define a template \text{Pres3SG} for verb forms like \textit{bakes}, as in (7).

\[(7)  a.  3sg = @(3\text{PersonSubj})  b.  \text{Pres3SG} = @(\text{Present}) @(\text{SingSubj}) @ (3sg)\]

The template definitions can be arranged in a template hierarchy to indicate interdependencies. The template hierarchy fragment in (8a) represents verb agreement information. The template hierarchy fragment in (8b) represents verb valency requirements.

\[(8)  a.  \text{Pres3SG} \quad b.  \text{Transitive-or-Intransitive} \quad \text{Transitive} \quad \text{Intransitive} \]

\[
\begin{align*}
\text{Present} & \quad 3sg \quad \text{Transitive} \quad \text{Intransitive} \\
\text{3PersonSubj} & \quad \text{SingSubj}
\end{align*}
\]

Given these template definitions, the lexical entry for \textit{bakes} is:

\[(9) \textit{bakes} V @(\text{Transitive-or-Intransitive bake}) @(\text{Pres3SG})\]

The parametrized template \text{Transitive-or-Intransitive}(P) is shared by verbs like \textit{eat}, \textit{cook}, and many others. The \text{Pres3SG} template is shared by verbs like \textit{appears}, \textit{goes}, \textit{cooks}, and many others. The template \text{Present}, used in defining the \text{Pres3SG} template, is also used by verbs like \textit{bake}, \textit{laugh}, and many others. The use of templates allows commonalities between lexical entries to be represented succinctly and for lexical generalizations to be encoded in a theoretically motivated manner.

Thus we show how templates allow lexical (illustrated above) and syntactic (to be illustrated in the paper) generalizations to be expressed in LFG, playing much the same role as the hierarchical type systems at the core of theories like HPSG. The template approach differs from the typed-feature-structure approach in that it is purely a notational device for factoring descriptions; templates are not part of a formal ontology and do not require an elaborate mathematical characterization. We have demonstrated that the ability to assign names to parameterized descriptions and then to use the assigned names to reference those descriptions in other descriptions is all that is needed to capture the kinds of linguistic insights that complex inheritance mechanisms would otherwise require.