LINGUISTIC GENERALIZATIONS OVER DESCRIPTIONS

Mary Dalrymple[*], Ronald M. Kaplan[†], and Tracy Holloway King[†]
[*]Oxford University [†]Palo Alto Research Center

Proceedings of the LFG04 Conference
University of Canterbury
Miriam Butt and Tracy Holloway King (Editors)

2004

CSLI Publications
`http://csli-publications.stanford.edu/`

Abstract[1]

LFG encodes linguistic generalizations not in terms of formal relations in a type hierarchy, 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 linguistic descriptions. In computational treatments, these named descriptions are generally referred to as *templates*. The use of templates allows for linguistic 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 shows only how pieces of descriptions are factored into patterns that recur across the lexicon and does not indicate the formal mode of combination of those pieces.

1 Introduction

A primary goal of syntactic theory is to identify generalizations about classes and subclasses of linguistic items, and in doing so to explore and characterize linguistic structure. A word like *yawns* belongs to several classes: it is a third-person, singular, finite, present-tense, intransitive verb. It shares some of these properties with a verb like *coughed*, and others with a verb like *devours*.

Linguistic theories have adopted different views as to how such 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). Such views were (and are) common among proponents of transformational approaches to syntax, since important linguistic generalizations were assumed to be best encoded transformationally, with the lexicon as a catchall for linguistic facts that could not be represented in general terms.

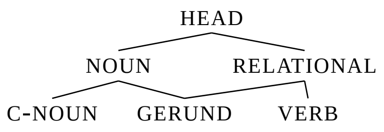
With the advent of constraint-based, nontransformational theories like LFG, this view of the lexicon changed. Bresnan (1978) observed that the effect of many transformations is better captured in terms of *lexical redundancy rules*: for example, the active and passive forms of a transitive verb, or the base and dative-shifted variants of a ditransitive verb, are related by lexical rules rather than by syntactic transformations. On this view, lexical information is no longer merely exceptional, idiosyncratic and therefore theoretically uninteresting. Instead, the lexicon and the rules relating lexical items become a prime locus of syntactic generalizations.

One of the first proposals for explicitly representing lexical generalizations 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 *yawns* belongs to the third-person singular present-tense class (like *devours*, *cooks*, and so on), the intransitive class (like *coughed*, *hiccup*, and so on), and to other classes as well. Classes may be subclasses of other classes, or may partition other classes along several dimensions: Flickinger analyzes VERB-TYPE and VERB-FORM as partitioning the class VERB, and FINITE as a subclass of VERB-FORM.

[1]This work was supported in part by the Advanced Research and Development Activity (ARDA)’s Advanced Question Answering for Intelligence (AQUAINT) Program.

Subsequent work within HPSG has built on this view. Linguistic generalizations in HPSG are captured by a *type hierarchy*, with more specific types inheriting information from less specific but related types. Construction Grammar (Kay, 1998) assumes a similar hierarchy, the constructional hierarchy. On the HPSG view, lexical generalizations are statable as relations between elements in the type lattice, where different subtypes represent alternatives, and a type can belong to multiple supertypes. For example, Malouf (1998) provides the following depiction of a partial type hierarchy of HEAD values:

(1)



This diagram represents an AND/OR lattice: the alternative types NOUN and RELATIONAL are disjunctively specified as different subtypes of the type HEAD. The type GERUND inherits from two supertypes, NOUN and RELATIONAL, and the information inherited from all supertypes is conjoined.

Work within LFG, on the other hand, has not appealed to typed feature structures to encode linguistic generalizations. Instead, LFG encodes lexical generalizations not in terms of formal inheritance relations between types, but in terms of inclusion 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 other linguistic descriptions. In computational treatments, these named descriptions are referred to as *templates*. A description containing a reference to a template is equivalent to that same description with the named equations, the template's definition, substituted for the template reference.

Template definitions can refer to other templates; thus, a template hierarchy similar to the type hierarchy of HPSG or Construction Grammar can be drawn to represent the inclusion relations between these named LFG descriptions. Importantly, however, the relation depicted in such a diagram shows only how pieces of descriptions are factored into patterns that recur across the lexicon and does not indicate the formal mode of combination of those pieces. The context of the template reference is what determines how the template definition combines with other parts of a larger description.

In the following, we will present several small template hierarchies and show how they can be used in the definition of linguistic constraints. For more discussion of computational issues related to the use of templates in grammatical description, see King et al. (2004).

2 Template definitions

We begin with a simple lexical entry for the verb *yawns*:

(2) *yawns* (↑ PRED)='yawn⟨SUBJ⟩'
(↑ VFORM)=FINITE
(↑ TENSE)=PRES
(↑ SUBJ PERS)=3
(↑ SUBJ NUM)=SG

This lexical entry contains information that is shared by other verbs. We can define the templates PRESENT and 3SG to encode this common information:

(3) PRESENT = (↑ VFORM)=FINITE
(↑ TENSE)=PRES

3SG = (↑ SUBJ PERS)=3
(↑ SUBJ NUM)=SG

The template name PRESENT names the functional description consisting of the two equations (↑ VFORM)=FINITE and (↑ TENSE)=PRES, and similarly for 3SG. With these definitions the entry for *yawns* can be rewritten as

(4) *yawns* (↑ PRED)=‘yawn⟨SUBJ⟩’
@PRESENT
@3SG

A template reference (or invocation) in a lexical entry or in the definition of another template, as in ((5) below), is marked by a preceding at-sign “@”. The present-tense and third-singular templates will be invoked by all similarly inflected verbs, so that the details of these subdescriptions are specified in one place but effective in many.

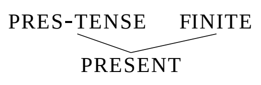
We can further subdivide the functional description named by PRESENT into two more primitive template definitions:

(5) FINITE = (↑ VFORM)=FINITE

PRES-TENSE = (↑ TENSE)=PRES

PRESENT = @FINITE
@PRES-TENSE

These template definitions can be arranged in a simple hierarchy that indicates their interdependencies:

(6)


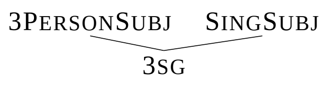
This diagram records the fact that the PRES-TENSE and FINITE templates are both referenced in (or inherited by) the definition of PRESENT. Similarly, we can also subdivide the 3SG template as follows:

(7) 3PersonSubj = (↑ SUBJ PERS)=3

SingSubj = (↑ SUBJ NUM)=SG

3Sg = @3PersonSubj
@SingSubj

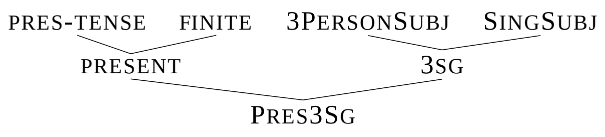
This information can also be represented as a template hierarchy:

(8) 

Finally, we can define a template Pres3Sg that combines both tense and agreement features:

(9) Pres3Sg = @present
@3sg

Putting all these definitions together, our template hierarchy becomes

(10) 

and the lexical entry for *yawns* further reduces to

(11) *yawns* $(\uparrow$ pred)='yawn⟨subj⟩'
@Pres3Sg

Thus we see that a number of hierarchically arranged generalizations can be expressed through a simple set of template definitions. The use of parameterized templates allows for further generalizations to be captured by factoring out information provided as an argument to the template. These are discussed next.

3 Parameterized templates

All intransitive verbs in LFG carry a semantic form that indicates the relation denoted by the verb and also the fact that the verb must appear in f-structures containing the single governable grammatical function subj. The predicate, of course, differs from verb to verb, but the subj subcategorization frame is common to all intransitives. We can define intransitive as a parameterized template that expresses the common subcategorization. The predicate itself can be provided as an argument that is specified differently in different lexical entries. This template can be used with all intransitive verbs:

(12) intransitive(p) = $(\uparrow$ pred)='p⟨subj⟩'

Whatever argument is provided in an invocation of this template will be substituted for the parameter to create the description that replaces the template reference. Thus the description in the original entry for the verb *yawns* can be equivalently specified as follows:

(13) *yawns* @intransitive(yawn)
@Pres3Sg

Arguments to parameterized templates can represent any part of an f-structure description: attributes as well as values and even whole subdescriptions can be parameterized. Templates can also take multiple arguments. For example, the template for a particle verb might take the verbal predicate as one argument and the form of the particle as another:

(14) VERB-PRT(P PRT) = (↑ PRED)=‘P⟨SUBJ, OBJ⟩’
(↑ PRT-FORM)=c PRT

The few templates we have defined serve to demonstrate the point that templates interpreted only by simple substitution allow commonalities between lexical entries to be represented succinctly and for linguistic generalizations to be expressed in a theoretically motivated manner. The parameterized template INTRANSITIVE(P) is shared by verbs like *sneeze*, *arrive*, and many others. The PRES3SG template is shared by verbs like *appears*, *goes*, *cooks*, and many others. The template PRESENT, used in defining the PRES3SG template, is also used by verbs like *bake*, *are*, and many others.

4 Templates and Boolean operators

In LFG, complex descriptions can be conjoined, disjoined, or negated. Since templates are just names for descriptions, we can also use these operators with templates. For instance, we could define a template PRESNOT3SG by negating the 3SG template, as follows:

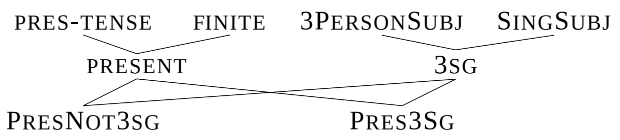
(15) PRESNOT3SG = @PRESENT
¬@3SG

The substitutions specified by these invocations produce the following description:

(16) (↑ VFORM)=FINITE
(↑ TENSE)=PRES
¬{(↑ SUBJ PERS)=3
(↑ SUBJ NUM)=SG}

The first two lines are the result of expanding the PRESENT template, and the third and fourth lines are the negation of the expansion of the 3SG template. This template can be used in the lexical entry of verbs which are present tense but whose subject is not third person singular (*yawn*, *bake*, *appear*, etc.).

With this addition we have the following template hierarchy:

(17)


This indicates that PRESNOT3SG includes (“inherits”) descriptions from both of its ancestors. However, unlike an HPSG type hierarchy, this does not entail that the inherited information is conjoined. For example, in (15) PRESNOT3SG invokes 3SG via negation. Template sharing is distinct from the mode of combination, which is determined by the context of the invocation.

For another illustration of this point, suppose that we have defined a parameterized TRANSITIVE template to be used for verbs like *devour*:

(18) TRANSITIVE(P) = (↑ PRED)=‘P⟨SUBJ, OBJ⟩’

This can be combined disjunctively with the INTRANSITIVE template to define a subcategorization template for verbs that can appear either with or without an object (*eat*, *cook*, *bake*, etc.):

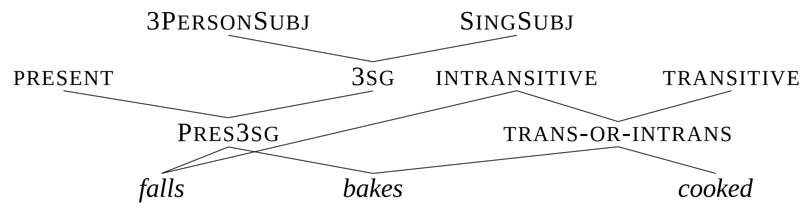
(19) TRANS-OR-INTRANS(P) = @TRANSITIVE(P) ∨ @INTRANSITIVE(P)

Notice that the parameter for TRANS-OR-INTRANS appears as an argument in the invocations of both INTRANSITIVE and TRANSITIVE. The reference @TRANS-OR-INTRANS(*eat*) thus expands ultimately to the disjunction

(20) (↑ PRED)=‘*eat*⟨SUBJ, OBJ⟩’ ∨ (↑ PRED)=‘*eat*⟨SUBJ⟩’

Finally, we can extend the hierarchical template inclusion diagram so that it bottoms out in particular lexical items, thus showing how generalizations are captured not only among the templates but also across the lexicon:

(21)



5 Expressing defaults

Default values can be expressed in LFG by means of existential constraints and disjunction. An *existential constraint* asserts that a feature must be present in an f-structure but it does not define the value that the feature must have. Thus the existential constraint in (22) is satisfied only if the f-structure denoted by ↑ has some value for the feature CASE:

(22) (↑ CASE)

This asserts that some (unspecified) value must be provided by a defining equation for the f-structure ↑. Otherwise, the existential constraint is not satisfied.

We can disjunctively combine this specification with a defining equation stating that the f-structure ↑ has the feature CASE with value NOM:

(23) (↑ CASE) ∨ (↑ CASE)=NOM

The first part of the disjunction is satisfied if $\uparrow$ has some value for CASE provided by a defining equation elsewhere in the functional description (the existential constraint in the left disjunct is satisfied). The second part of the disjunction is satisfied if $\uparrow$ has the value NOM for CASE (the defining equation in the right disjunct is satisfied). The effect is that NOM is the default value for CASE: if no other value is defined for that feature, the value NOM will be installed.

This technique for specifying a default value V for a designator D can be encapsulated in a parameterized template:

(24) DEFAULT(D V) = D $\vee$ D=V

and we can use this to make more obvious the fact that NOM is the default value of CASE:

(25) @DEFAULT(($\uparrow$ CASE) NOM)

An invocation of this default CASE assignment template could then be a part of the lexical description for a noun in a language with case clitics. If there is no case clitic to specify a particular case for the noun, the default NOM case will appear.

6 Templates and Phrase Structure Annotations

Since templates simply stand for pieces of functional descriptions, it is also possible to use templates in annotations on phrase structure rules, to capture recurring generalizations in the specification of the relation between c-structure configurations and f-structures. There is no difference in the way templates are defined or invoked when they are used in phrase structure rules; functional annotations in phrase structure rules can simply be replaced with a template reference.

To take an example, suppose that every adjunct in the grammar must be annotated with both its grammatical function and an ADJUNCT-TYPE feature, e.g., (26).

(26)
$$\begin{array}{lll} \text{VP} \longrightarrow & \text{V} & \text{ADVP*} \\ & \uparrow=\downarrow & \downarrow \in (\uparrow \text{ ADJUNCT}) \\ & & (\downarrow \text{ ADJUNCT-TYPE})=\text{VP-ADJ} \end{array}$$

This can be rewritten using a parameterized template:

(27)
$$\begin{array}{lll} \text{VP} \longrightarrow & \text{V} & \text{ADVP*} \\ & \uparrow=\downarrow & @\text{ADJUNCT(VP-ADJ)} \end{array}$$

where the ADJUNCT template expands to:

(28) a. ADJUNCT(P) = $\downarrow \in (\uparrow$ ADJUNCT)
@ADJUNCT-TYPE(P)

b. ADJUNCT-TYPE(P) = ($\downarrow$ ADJUNCT-TYPE)=P

Coordination is another instance where templates are useful for capturing generalizations over phrase-structure annotations. In NP coordination, the conjuncts must be labelled not only for their relation to the f-structure of the coordination (↓∈↑) but also for information as to how the person and gender features of the conjuncts are resolved. Instead of repeating this set of equations for each conjunct, a template can be invoked which contains all of the relevant annotations.

(29) a.

$$\begin{array}{llll} \text{NP} \longrightarrow & \text{NP+} & \text{CONJ} & \text{NP} \\ & @\text{NP-CONJUNCT} & \uparrow=\downarrow & @\text{NP-CONJUNCT} \end{array}$$

b.

$$\begin{array}{lll} \text{NP-CONJUNCT} & = & \downarrow\in\uparrow \\ & & @\text{CONJ-PERS} \\ & & @\text{CONJ-GEND} \end{array}$$

The templates CONJ-PERS and CONJ-GEND are then defined so as to impose the proper constraints on feature resolution in coordinate structure. Adopting the theory of feature indeterminacy and feature resolution proposed by (Dalrymple and Kaplan, 2000), the person and gender features are represented as sets of markers, and resolution is accomplished by the set-union implied by specifying subset relations on those marker sets:

(30)

$$\begin{array}{lll} \text{CONJ-PERS} & = & (\downarrow \text{PERS}) \subseteq (\uparrow \text{PERS}) \\ \text{CONJ-GEND} & = & (\downarrow \text{GEND}) \subseteq (\uparrow \text{GEND}) \end{array}$$

Thus, templates can be used to capture generalizations across functional descriptions not only within the lexicon but also within phrase structure annotations.

7 Conclusion

We have observed that templates can play the same role in capturing linguistic generalizations as hierarchical type systems in theories like HPSG. An important difference is that templates are simply a notational device for factoring descriptions, not part of a formal ontology. Template invocations are interpreted solely as instructions to substitute the named descriptions into other descriptions and do not require a more elaborate mathematical characterization. They are simply a means of collecting together and reusing linguistically significant collections of descriptions.

Templates differ from hierarchical type systems in two ways. First, parameterized templates allow for templates to be systematically specialized in particular ways. Second, since templates are just ways of naming descriptions, they can participate in Boolean combinations of descriptions involving disjunction, conjunction, and negation. These two differences may make it easier and more intuitive to express linguistic generalizations in comparison to hierarchical type systems.

References

Bresnan, Joan. 1978. A realistic transformational grammar. In Morris Halle, Joan Bresnan, and George A. Miller (editors), *Linguistic Theory and Psychological Reality*. Cambridge, MA: The MIT Press.

Dalrymple, Mary and Ronald M. Kaplan. 2000. Feature indeterminacy and feature resolution. *Language* 76(4), pp. 759–798.

Flickinger, Daniel P. 1987. *Lexical rules in the hierarchical lexicon*. Ph.D. thesis, Stanford University.

Kay, Paul. 1998. An informal sketch of a formal architecture for Construction Grammar. In *Proceedings of the Conference on Formal Grammar, HPSG and Categorial Grammar*. Saarbrücken. URL `www.icsi.berkeley.edu/ kay/cg.arch.ps`.

King, Tracy Holloway, Martin Forst, Jonas Kuhn, and Miriam Butt. 2004. The feature space in parallel grammar writing. *Research on Language and Computation*. In press.

Kiparsky, Paul. 1982. Word-formation and the lexicon. In F. Ingemann (editor), *Proceedings of the 1982 Mid-America Lingistics Conference*.

Malouf, Robert. 1998. Categories, prototypes, and default inheritance. In *Proceedings of the Joint Conference on Formal Grammar, Head-Driven Phrase Structure Grammar, and Categorial Grammar*, pp. 207–216. Saarbrücken.