Symmetric Slot Grammar (SSG): A Bi-directional Design for MT

P. Newman, IBM Palo Alto Scientific Center
1530 Page Mill Road, Palo Alto, CA 94304, USA

Abstract

Grammar formalisms combining a lexicalist orientation with an immediate-dominance/linear-precedence (ID/LP) partitioning *of* information have considerable potential for bi-directional use, fundamentally because the information is expressed in a modular way and thus can be applied differently in parsing and generation. This paper describes an in-process design for a bi-directional formalism of this type. The formalism is intended for use in a multi-lingual machine translation system, and is founded on the ideas of the currently unidirectional "Slot Grammar" of McCord (1989a).

1. Introduction

In machine translation it is highly desirable to separate transfer from generation, with generation processors source-language independent, for economic reasons. This allows the use of only one generation grammar per target language L instead of one per translation pair x $\rightarrow$ L. Also, it lowers the cost of modifying transfer interfaces, e.g., to incorporate more semantic and textual knowledge. Bi-directional grammars ensure source-independent generation, and provide further economic advantage. Also, as observed by Dymetman and Isabelle (1988), they facilitate grammar validation.

There have been major strides taken in recent years in bi-directional formalisms, based on many different paradigms. Among these efforts are: (a) the CRITTER system described by Dymetman and Isabelle (1988), in which an annotated definite clause grammar is compiled differently, based on the annotations, for the two purposes, (b) the inversion of a systemic generator by Kasper (1988) (in which phrase structure is said to be added manually for parsing), (c) the DEW generator of Caraceni and Stock (1988), which combines a Iexicalist framework with augmented transition network (ATN) rules, and which seems to employ a "generate and test" approach to generation, and (d) the Berlin GPSG effort of Buseman and Hauenschild (1988) in which GPSG (Gazdar et al. 1985) is adapted for implementation purposes to allow feasible rule specification and sequencing.

Head-driven grammars provide a very promising basis for bi-directional use. By the term "head-driven grammars" is intended grammars like HPSG (Sag and Pollard 1987) and Slot Grammar (McCord I989a) which: (a) follow the imrnediate-dominance/linear-precedence (ID/LP) partitioning of information associated with GPSG, and (b) represent substantial amounts of ID information in lexical terms. Some important advantages of this combination, as discussed by Newman (to appear) are that: (a) the lexical focus facilitates mappings to and from abstract representations of utterance content suitable at transfer interfaces, and (b) the modularity afforded by the ID/LP partitioning allows rules to be combined in different ways in parsing and generation. So, for example, during the parse ordering constraints can be checked as they become applicable, while in generation one can first produce all components of a constituent and then order them. This modularity also offers the potential for subsuming within the grammar the analysis and use of textual features relating to ordering in a natural way.

In this paper we describe an in-process design for a bi-directional grammar of this type, intended for use within a machine translation system. The description is given in terms of set of suggested modifications and additions to Slot Grammar. The latter, which derives from an earlier hybrid effort by McCord (1980). is currently in use for parsing, but not generation, within another experimental MT system, LMT-2 (McCord I989b). It is used here as a foundation because it has some interesting features which seem naturally extensible to a bi-directional version - most notably the use of named *slots* to represent dependents of various types. The slots are used as a basis for expressing many other facts about the so-labelled dependents in a convenient, modular way.

We first outline the major structures of Slot Grammar and how they are currently used. We then sketch the intended machine translation context for the adaptation, to provide a general motivation for the proposed modifications. The body of the paper is devoted to the proposal of a set of modifications and extensions to the current Slot Grammar structures to support bi-directional use. It should be emphasized that these proposals should be interpreted as directional; they have not been tested sufficiently to represent a final form and a number of important areas require further work.

2. Overview of Slot Grammar

Table 1 on page 3 illustrates some of the major structures as given by McCord (1989a).

Table 1. Some Major Constructs of Slot Grammar.

1. lexicon entries
give < v(obj.iobj)
< v(obj.advpart(up))

2. adjunct rules
adjuncts(noun, det.quant...)

3. ***slot filler rules***
iobj = = > f(noun(*,dat.,*,*))
iobj = = > f(prep(to,*,*))

4. ***head/slot ordering rules***
lslot(SLOT) <- CONDITION
rslot(SLOT) < - CONDITION

5. ***slot/slot ordering rules***
iobj < < obj < - lf(noun(*,*,*)
obj < < iobj < - rf(prep(*,*,*)

The *lexicon entry (1)* for a word consists of subentries each representing a different syntactic configuration in terms of part-of-speech + major differences in expected complements. Expected complements are specified in terms of *slots,* which may be parameterized. For example, a slot "advpart" (adverbial particle) may take the specific particle as a parameter.

Complements are listed in a canonical order subsequently reflected in a semantic predicate which forms part of the parser output. Lexical transformations are used to change the complement slots for passivization and nominalization. Subjects are omitted as verb complements, as they are assumed, and provisions are made for identifying complements as obligatory or optional. A lexicon subentry may also specify additional associated words and word-types expected in fixed positions relative to the head, for use in specifying lexicalized compounds and non-compositional forms, e.g., "at least N."

Adjunct rules (2) specify, for each syntactic category, the types of adjuncts which can modify heads in that category. Adjunct types are also given in terms of slots.

Slot filler rules (3) indicate, for each slot, constraints on constituents which can fill that kind of slot. The constraints are specified in terms of structures with which dependent must unify ("f(...)"). Slot filler rules are used for the specification of many types of "immediate dominance" rules in addition to dependent frame constraints. For such purposes, e.g., to specify agreement requirements, additional condition terms may be included in the rules, including unifiers for the head ("hf(....)"), or arbitrary conditions (which, in the current version of slot grammar, refer to separate Prolog goals).

Head/slot ordering rules (4) indicate on which side or sides of a head a particular slot filler can appear. The applicability of these rules may also be constrained based on unification with heads, the candidate slot filler, or other arbitrary constraints.

Slot/slot ordering rules (5) indicate required precedence relationships among slot fillers, which also may be conditional.

We will restrict our discussion of Slot Grammar to these types of rules. There are other rule types, most of which are also given in terms of slots, used in specifying such aspects of the grammar as long distance dependency and coordination.

A bottom up chart parser is used as the basic parsing mechanism. Each newly-formed adjacent pair is examined to determine whether one member of the pair represents an expected complement or adjunct of the other. If that is the case, the ordering constraints are examined to determine if any would be violated by the identification. If these tests are passed, a larger node is formed by adding the complement or adjunct to the developing constituent associated with the head.

Output from the parser consists of a relatively flat parse tree. Each node of the tree provides both (a) a "semantic" representation listing only the complements of the head, in canonical order (i.e., the same order independent of slot assignments), and (b) a list of all dependent constituents in source order, specifying the assigned slots. The semantic

representation isolates substantive parts of complements from obligatory ones, such as obligatory prepositions, to some extent.

This parser is currently operational within an experimental machine translation system LMT-2 (McCord 1989b). In this system, parsing is followed by a lexical transfer phase and then by generation. Lexical transfer is guided by the association of one or more target lexical specifications with each source lexicon subentry. The target specifications contain complement slot lists matching those of the source in length and order; differences in complement expression are indicated by the use of different slotnames and by features. The transfer process substitutes the target lexical items and slots for the source, identifying where structural differences exist. The generator, which is dependent on both the source and target languages, then produces a target tree by tree transformations keyed to target slotnames.

3. Context Of Bi-Directional Adaptation

To motivate the specifics of the bi-directional adaptation, we discuss the intended machine translation system context, MT/LASC, which is different from the current Slot Grammar usage context.

MT/LASC is a semi-interlingual, yet transfer-based, design. The unilingual lexicons contain entries for words decomposed to subentries based on category + expected complements, as described in the previous section. The subentries are linked to a shared "common lexicon" of concepts and associated frames. (A unilingual subentry may be linked to multiple concepts, and vice versa.)

Utterances are represented at the transfer interfaces (i.e., analysis output and generation input) by relatively abstract hierarchic feature structures. A constituent is represented as a collection of features associated with the head and constituent as a whole, plus a list of "dependent connectors" identifying the dependents. Each dependent connector is a feature structure; one feature is "value = the actual dependent," and other features describe the particular head/dependent relationship. For complements, this relational information includes its "complement index" in the canonical order for complements of the head.

In this structure many source-language specific elements have been removed or transformed. Thus pro-forma complements (e.g., adverbial particles), and pro-forma elements of complements (e.g., the obligatory "for" in "wait for Bill") are removed, and certain closed class items, such as determiners and auxiliaries, are represented only by the appropriate constituent features. Remaining lexical items are represented as "word-senses," matching concepts in the semi-interlingual lexicon.

Also, variations in complement structure beyond variations in obligatory lexical items are removed, so that, for example, clefting (e.g., "It was John who...") is eliminated. Finally, textual devices used in the source are captured by features indicating, to the extent possible, theme and focus elements, and special marking devices, such as topicalizalion, passivization, and clefting.

During parsing, a similar structure is used, but one which retains surface features. Lexical items are represented at this stage by indices to subentries of the unilingual lexicon. The disambiguation of unilingual subentries to common-lexicon concepts is achieved subsequent to parsing by selecting links to the concept lexicon using an algorithm described by Newman (1988).

Transfer is responsible for further decomposition of concepts, if necessary, and substantive paraphrase, with both processes controlled by information in the concept entries. Some collocational decisions are made using conditions on links between the semi-interlingual lexicon and the target lexicon using specifications related to lexical constructs, described by Mel'cuk and Polguere (1987).

Generation is essentially source-language independent. The organization of a target constituent depends on a combination of its relationship to its head, semantic content, and textual features. Producing target constituents involves such processes as as (a) making decisions as to the use of marked textual forms, (b) adding, in target language terms, obligatory lexical items, and (c) ordering dependents based on syntactic constraints and textual preferences.

4. Overview of Adaptation

To allow use of a Slot-Grammar-like formalism for both analysis and generation in above semi-interlingual context. several kinds of modifications are proposed. Some modifications support mapping from source-oriented to relatively neutral utterance representations. There are some mapping provisions in Slot Grammar; for example, those which extract substantive content from some parameterized slots, and those which obtain implied subjects of infinitive complements; these provisions are extended and integrated.

Other modifications are needed to allow generation decisions in the presence of alternatives, a function not needed in a grammar used only for parsing. Decisions are needed among the following:

- Alternative basic constituent forms involving different allocation of material among contained constituents (e.g., as in "I believe he is...," "I believe him to be....")
  Alternative expressions of individual logical complements (e.g. between the use of an indirect object or a prepositional phrase headed by "to."
* Alternative orderings of complements where many permutations are possible.

The proposed extensions to Slot Grammar relating to the "immediate dominance" area, that is, to mappings and to selections among alternatives not directly involving ordering, are addressed in the next section. That discussion is followed by an outline of how the extensions are used in the generation procedure. Next, the extensions relating to ordering decisions are presented. The final section, on further work required, discusses aspects of the above listed requirements not fully covered by the proposals.[1]

5. Immediate Dominance Modifications

This section details proposed modifications to lexicon entries and slot filler rules for purposes of (a) assisting in the selection of specific complement combinations, (b) identifying slots appearing only in surface representations, i.e.. which should not appear at the transfer interface, and (c) explicitly mapping between surface and transfer-interface representations of complements.

5.1 Complement Combination Selection

Figure 1 illustrates lexical provisions made for complement combination selection, using as an example the subentry for "ask someone for something," which allows such alternative forms as "He asked Mary for a book," and "He asked a book of Mary."

```
lex.ask.2  /* ask SOMEONE for SOMETHING */
      (cat:v,
      comps: ((obj).(reqp,of).(reqp,from)).    /* SOMEONE */
                  ((obj).(reqp.for))                    /* SOMETHING */
       prefs: (pref: (obj).(reqp,for)) cond : ...)...
```

Figure 1. Complement Combination Selection

Note first that the notation for lexical entries (and for other rules) is modified from the original formalism. A rule is identified by a compound name indicating the type of rule ("lex"), and instance ("ask.2," the second subentry for ask). The content of the rule is given as an explicit feature structure. This is intended both for clarity (at least during development), and to allow the use of abbreviations (not shown) for information shared among entries.

In the modified entry, slots are given by the attribute "romps," which is a list of lists. Each sublist gives the alternative slots for a particular logical complement. Some slots are parameterized, and the associated filler rules (see below) expect a parameter. Thus the slots which can be used for the first (canonical) complement (SOMEONE) in

[1] It should also be mentioned that modifications to the unilingual lexicons have been designed to allow fully reversible morphological processing. Those modifications, applicable in their current form to at least English and Romance languages, will not be discussed here.

the example are (obj).(reqp.of).(reqp.from). The last two slots are both "reqp" (required preposition) slots, respectively with parameters "of and "from."

The modified entry also contains a "prefs:" specification, which lists preferred complement combinations. Each list element gives one combination, together with an optional condition for the selection of that combination. The conditions, which are not shown here, are expressed as unifiers for the head, and involve variations such as "register" and presence or absence of specific optional complements. In the above example, the single preferred combination is specified as "ask x for y." We note that the selection of a combination may also be based on more general textual features. This latter subject is discussed briefly in conjunction with "directions for further work."

```
a.   lex.give.2   /* give up = surrender */
        (cat:v, comps: ((obj)).((advpart,up)))

b.  lex.wait.2 /* wait for */
        (cat:v, comps: ((reqp.for)))

c.  lex.take.3   /* take (good) care of */'
          cat:v, comps: (((pfnom,care)).((reqp,of)))
```

Figure 2. Unilingual Lexicon Subentries

Figure 2 shows some additional example lexicon entries, for "give up," "wait for," and "take care of," respectively, which will be related to filler/map rules below. For "take care of," the first complement slot is "pfnom" (pro-forma nominal), a complement noun appearing only in the surface structure, taking an optional adjective modifier.

5.2 Surface Slot Specification

The specification of entire slots appearing only in surface forms is done by "surfaceslots" rules, as illustrated in Figure 3.

```
surfaceslots.adjuncts (slots: det.aux....)
surfaceslots.comps (slots: advpart...)
```

Figure 3. Unilingual Slot Specifications

The rules state that fillers for the adjunct slots "det" and "aux," and for the complement slot "advpart" do not appear at the transfer interface. The use of these rules is discussed below in conjunction with explicit mappings, and with generation control.

5.3 Filler/Map Rules

Explicit mappings between surface and transfer interface forms are included in the grammar by recasting slot filler rules to serve both as conditions on surface slot fillers and as expressions of the necessary transformations. The filler/map rules are influenced by generalized unification structures but depart from them in significant ways.

Generalized unification is a popular method for mapping between "syntactic" and "semantic" representations, as described, for example, by Shieber (1986). Thus one might obtain a mapping for prepositional phrases containing required prepositions as shown in Figure 4.

```
filler.reqp (
            syn: (cat: p   lex: $parm   pobj:  < 1 > )
            sem:  < 1 > )
```

Figure 4. Mapping by Unification Only

In parsing the "syn" substructure would be unified with that representing the source prepositional phrase to obtain the "sem" substructure, etc. But restricting possible operations to unification, both in testing fillers and in mapping, can present some difficulties. For example, it is difficult to indicate that a given attribute must be present. Also, list handling tends to be difficult. For these reasons the notation adopted is a more flexible, procedural one.

The filler/map rules consist of lists of binary expressions

```
opndl operators opnd2
```

for example,

```
#syn.cat = p
```

where operands may be variables (#xxx) representing structures or atoms, attribute references (#xxx.yyy....), functions ($xxx, see below), or literals. The structure references of left-hand operands may be factored, so that

```
#xxx:(att1 - val1, att2 = val2)
```

is equivalent to

```
(#xxx.att1  = val1).(#xxx.att2 = val2)
```

Some variable names are reserved; others may be introduced as needed. Each expression may have one or two operators. If there is one operator, it applies both in parsing and generation. If there are two, the first applies in parsing and the second in generation.

The operators include " = " (unify), "!" (test for equivalence), and " <-" (assign). "<" indicates "contains," or, more precisely, "find a member of the list denoted by the first operand which either unifies with the one on the right (if the latter is an atom or variable), or for which the indicated operations succeed (if not)." " + " specifies that the second operand (if it is an atom or variable) be added to the list.

Examples of filler/map rules are shown in Figure 5.

```
a./* Adverbial Particle */
filler.advpart (#syn: (cat = particle lex = $parm)).

b./* Required Preposition */
filler.reqp (#syn: (cat=p, lex = $parm)). .
            (#syn: (dtrlist > + (compix = 1, value = #sem)))

c. /* A Class of Idiomatic Object */
filler.pfnom (#syn: (cat = n, lex = $parm, dtrlist > + (type = adjunct,
                                                value = #sem)).
              (#sem: (cat = adjective)).

filler.pfnom (#syn: (cat = n, lex = $parm,  dtrlist !<- $empty). .
               (#sem < -! $empty).
```

Figure 5. Filler/Map Rules

The filler/map rules are used in three different ways. During the initial parse they are used to check the candidate slot filler. To do this, the variable #syn is set to the value of the candidate slot and #head to the current head. Then the expressions are evaluated in sequence up to an empty expression denoted by two successive "." (i.e.. . .). During the evaluation of a filler rule, the function $parm gives the value of the slot parameter, if any. Thus part (a) of Figure 5 specifies that the filler for an "advpart" slot must be a particle whose lexical head (lex =) is equal to the value of $parm.[2]

[2] In general, functions represent either derived attributes, in which case their implicit argument is the structure imme-

After the parse, all expressions in the listed sequence are evaluated, including those beyond the empty expression (. .), with the aim of finding a value for #sem, the transfer interface representation for the slot. If no value is found for #sem, and the slot has been identified as surface only, or #sem is explicitly set equal to $empty, the slot is removed from the transfer input representation[3]. When the surface and transfer-interface representations of a slot are equivalent, the filler/map expression should be "(#sem = #syn)."

In generation the expressions are evaluated in the reverse order, with the goal of finding a value for #syn, the surface representation. The use of the filler rules in generation differs depending on whether the slot is a complement or adjunct, and whether it is surface-only. For complements present in the transfer output, #sem is set equal to the interface representation, and then filler/map rules for the different potential slots associated with the complement (i.e., with the complement index) are evaluated. For surface-only complement slots indicated by the (unilingual) lexical entry for the head, the filler/map rules are also evaluated, to find a value for #syn if appropriate (but #sem is naturally not set). For adjuncts present in the transfer output, #sern is set to their interface representation, and filler/map rules for adjunct slots associated with the head category are evaluated to find an appropriate slot and surface representation (#syn). Finally, for all surface-only adjunct slots (e.g., aux) associated with the head category, the associated filler/map rules are also evaluated, in this case to determine (usually via head features) if the slots are needed as well as to find their surface representation.

To discuss the examples a bit further, the filler/map rule in part (b) of Figure 5 deletes or adds an obligatory preposition: after parsing, evaluating the expressions in sequence produces a value for #sem equal to the object of the preposition. In the inverse direction the obligatory preposition is added. Part (c) contains two alternatives, depending on whether the complement contains an optional adjective ("take GOOD care of"). If it does, the adjective becomes the value of the transfer input representation of the complement. Otherwise, that value is null 4. Similar, but more complex, mappings are used to add logical subjects to the semantic representations of infinitive complements, with different slots used for different kinds of control (e.g., subject control).

Summarizing to this point, the extensions proposed in the area of "immediate dominance" concern selection among surface forms, and mappings between surface and transfer representations. The latter include simple addition/deletion of complements and adjuncts, as well as more complex transformations. Not discussed are a few additional modifications made to Slot Grammar specifications of Iexicalized compounds and non-compositional structures.

6. Generation Control

The discussion in this section is somewhat tentative. It should be considered more as an indication of how the rules interrelate for generation purposes than as an algorithm, for which further development is required.

The overall sequence of generation is top-down, and must have considerable backtracking capability. At the beginning of the expansion of each constituent the following situation exists:

> The head of the constituent is known, and some additional features of the head have been instantiated by prior processing.

- One of those features is the specification of dependent slot combination alternatives. This may be just a copy of the "comps:" list from the lexical entry, but it may also be a result of applying lexical transformations, e.g., to modify the slot alternatives for passives.

---

diately containing the reference, or general functions, which may have either explicit or implicit arguments. $parm is a function whose implicit argument is the most current filler/map rule parameter.

[3] Note that to allow complements to be properly numbered at transfer interfaces, unilingual complements (such as advpart) should be given the highest complement indices in the ordering.

[4] It is currently customary for transformations from surface to internal forms to be accomplished by specifying separate "syntactic" and "semantic" subframes, and relating them by structure sharing. We depart from this somewhat in that syntactic and transfer-interface representations can be related by transformation rather than by separate frames, to avoid the necessity for inordinate care in segregating "syntactic" and "semantic" features.

The constituent has an associated list of dependents in transfer-output form. Each is labelled as complement or adjunct and, if it is a complement, the attribute "compix" gives the complement index.

In general, expanding a constituent consists of (a) selecting a combination of slots for its dependents, (b) validating that combination by finding appropriate filler/map rules, which serve to set head features for the dependents, and (c) ordering the dependents. If a combination fails another must be tried until a successful one is found. After dependents are ordered, the process continues by expanding those dependents, etc.

The process is initiated using a dummy constituent "top," whose single dependent is the internal representation of the entire sentence. That dependent has a single slotname alternative, "sentence." Expansion of the dummy constituent thus consists of (a) selecting the slot "sentence," and (b) finding and applying an appropriate filler rule for the slot "sentence." An abbreviated set of such rules is sketched in Figure 6.

```
filler.sentence
      (#syn:
            ( voice = active,
              dtrlist > + (compix  = #ix, theme!true)
              $possubj  > #ix)).
      (#syn: (slots = $activeslots)).
      (#sem = #syn).

filler.sentence
      (#syn:   ( voice = passive .....
```

Figure 6. Filler Rules for Sentence Slot

To evaluate the first rule, #sem is given the value of the internal representation, and then #syn is given the same value. Then a slot alternative list is established by "slots = $activeslots," where $activeslots is assumed to be a function returning the unmodified content of the "comps" attribute in the lexical entry associated with the head. Following this,[5] the voice is set to active, and a check is made to see that, given the "activeslots" alternatives, there is a complement in the input which (a) can be a subject and (b) has been identified as a theme.[6] If these tests fail, a passive form is tried, etc.

After a filler rule for a slot is used in generation, if the head feature "slots" of the slot filler has not been set by that rule, the "comps:" value from the lexical entry is used by default.

To specify the expansion of a constituent in somewhat more detail, the individual steps used are roughly as follows:

1. Select a complement slot combination covering both standard and surface-only complements from the sets of alternatives (slots:), using preference information (pref: from the lexical entry), and global constraints (e.g., there can be only one object).

2. Validate the selection, at least at the first level, by finding an applicable filler/map rule for each slot in the combination. If the selection is invalid, find another.

3. Find a slot for each adjunct in the input.

4. Add additional adjunct slots (e.g. aux) using "surfaceslots" lists for the head category, and filler/map rules to determine the need for those slots.

.5. When a successful combination (to this point) is found, order the result using the fixed ordering rules in combination with the preferential ones described in the next section.

---

[5] While the "sentence" filler rule is not used in parsing, other filler rules referencing $xxx built-in functions are. Because of this the functions must generally operate differently in the two directions.

[6] Because the check on possible subjects must follow the establishment of a trial slot alternative list, there are two pairs for #syn.

7. Ordering Extensions

7.1 Use of Zones

Constituent ordering rules in Slot Grammar relate only to grammaticality. Additional rules are needed to determine the generation order of constituents susceptible to many syntactically acceptable placements. Such rules must be given in terms of relative acceptability, based on factors such as conventional positioning for certain kinds of dependents, the need to limit ambiguity, and the means provided by the language for expressing topic and focus. The rules, if symmetric, should also be usable in parsing to assist in disambiguation and to determine, to the extent possible, textual features implied by the ordering. The latter is a rather difficult problem for written text, but important criteria are given by Hajicova (1989) and Sgall et al. (1986) for Czech and English, and by Wesche and Renz (]987) for German.

The added preferential ordering rules are expressed in terms of (a) preferred placements of constituents within *zones* of their containing constituents, and (b) relative orderings within those zones. Some zones are already used in Slot Grammar, namely "left-of-head" and 'right-of-head". But finer decompositions are useful both conceptually and in writing rules. Quirk et al. (1972) identify zones for English clauses, such as *initial* (pre-subject), *medial-1* and *medial-2* (two different zones if auxiliaries are present, collapsing to one if not), and e*nd* (post-verb).[7] Then preferred placements of English adverbials are described in terms of these zones. Thus *viewpoint* adjuncts, such as "visually" and "technically," prefer *initial* placement:

2 (a) Visually, it was very impressive.
2(b) It visually was very impressive.

It is not only convenient to think of constituent placement in terms of zones, but also to express placement rules in those terms. Consider the alternative. To express the preference of "viewpoint" adverbs using only "slot/slot" rules might require something like the following:

```
adjunct < subj left(class = viewpt) head(mood = indic) pref(1)
```

This is problematical in that there is no indication of *which* constituent the preference is stated about (i.e., the adjunct or subject) and thus what the alternatives (presumably with other preference numbers) to that preference are. Also, placing a constituent in non-initial and non-final zones requires two pairwise ordering rules (the first specifying that the constituent succeeds the left boundary of the zone, and the second that it precedes the right boundary), which must somehow be bound together as expressing a single preference.

Using explicit zones also allows the imposition of constraints on zones per se. That is, they provide a convenient way of coping with zone co-occurrence restrictions. For example, the medial zone might be constrained to a single non-negative adverbial, allowing 3(a) to be chosen over 3(b).[8]

3(a) Generally, John easily passes exams.
3(b) John, generally, easily passes exams.

Before detailing proposed zone-related extensions, one other addition is introduced to simplify these and other rules, namely, the use of a slot lattice, i.e., where some slots can be declared superordinates. This allows more generalized placement rules. For example, one might like to say (in the absence of considerations with higher priority), that in the "end" (postverb) zone of a clause, complements are positioned before adjuncts, independent of the specific complement slot type.

---

[7] Jacobson (1964) identifies a more elaborate set of zones for English clauses.

[8] Note that these examples assume very flat syntactic structures, as in Kartunnen (I986). In the statement of zone allocation and other ordering rules this allows dependents assigned to different pre-main-verb zones to be considered logical siblings. Thus auxiliaries are treated as adjuncts and, unlike most current approaches, fronting is not viewed as an example of long-distance dependency (except for fronting of components of complements or adjuncts).

7.2 Zone Definition and Allocation to Zones

Figure 7 illustrates proposed rules for defining zones and allocating items to those zones.

(1) Zone definition rules

```
zone.initial (head:($clausetype > mainclause),
                lbd:($slot > begin), rhd:($slot > subj))
zone.medial (head:($clausetype >  mainclause),
                lbd:($slot > subj). rbd:($curhead = true),
                limits:(cond: ($slot -  >  (or.aux.neg)) limit = 1)...
```

(2) Zone Placement Rules

```
zput.adjunct (cond:($domain > viewpoint),
              prefs:(initial,4). (medial,3). (final,1))
```

Figure 7. Zone Definition and Placement Rules

A *zone definition rule (1)* names a zone, describes the conditions under which it exists, and identifies its boundaries. It also describes constraints on the contents of the zone. Referring to the example, "head:" indicates existence conditions in terms of constraints on the head. "$Slot" is a function identifying the slotnames associated with a constituent. "Lbd:" and "rbd:" identify the left and right boundaries of a zone in terms of constraints on the boundary constituents. "Begin" and "end" are built-in slotnames representing the beginning and end of a constituent, respectively. "Limits:" is a list indicating conditions on sets of contained constituents; that shown limits non-auxiliary and non-negative complements within the medial zone to a maximum of 1.

Zone implication specifications, which relate nested ones to their containing ones, are not shown. They are used to specify combined content limits, and to infer preferences for larger zones where smaller ones do not apply.

Zone placement rules (2) specify one or more possible zone placements for the named slot fillers, in order of relative preference. The preferences can be weighted, with the weights used (at least) to indicate relative preferences beyond simple ordering, e.g., that a particular placement is acceptable but unfortunate.

These rules can be used in generation as follows. First, absolute ordering rules (equivalent to those in Slot Grammar) are employed to place some constituents right or left of head, and to order them. This will not, in general, produce a total ordering, but there must be sufficient absolute rules to totally order any zone boundary slots present. Then the most deeply nested zones present are found. Placement roughly proceeds by performing initial allocations based on first preferences until a constraint of a zone or containing zone (limits are stated on both) is violated. At this point the search "backtracks" (logically), while attempting to place a constituent in a less preferred zone.

Zone placement rules are also to be used in parsing, for two purposes: (a) disambiguation, and (b) discovering textual features. Zone rules can assist in disambiguation by helping to choose between alternative parses. I or example, the most probable meaning of "happily" in 4(a) might be selected by noting that "attitudinal" adverbs prefer an initial position, while manner adverbs do not except in certain kinds of texts.

4(a) Happily John did the homework.
4(b) John happily did the homework.

The process of discovering textual features, using both zone placement rules and relative ordering rules (discussed in the next section), should be a process of identifying which rules were applied to obtain the input ordering. For example, given the sentence:

5 ***That house Mary never saw.***

and the rule

```
zput.comp (cond:(cat = n . $slot - > subj. textfeat + > markedtheme)
            prefs:(initial))
```

Figure 8. A Placement Rule for Fronted NP Complements

one might determine that "that house" was a marked theme. In general, however, the process of finding a set of consistent conditions which together could have determined an ordering is a very difficult one. This area requires considerable additional work, and will probably require some modification of rule structure.

7.2 Ordering Within Zones

In generation, the allocation of movable constituents to zones is followed by ordering those constituents within the zones. In this section we describe a possible approach to the latter process. While it should be adequate for many purposes, it has some limitations which are discussed further on.

```
porder.end. vdep.comp (weight :4,
                       right:($large = true)   /* e.g., heavy np shift */
porder.end.vdep.vdep (weight:3, right:($domain > loc),
                       left.:(domain ¬ > loc))   /* final locatives */
porder.end.vdep.vdep (weight:2.5, left:(textfeat ¬ > focus),
                       right:(tcxtfeat > focus))
porder.end.comp.adjunct (weight:2)
porder.end.vdep.vdep (weight: 1.5, rules:roleprec) /* semantic role based
ordering */
```

Figure 9. Preferential Ordering Rules

Some simplified preferential ordering rules for "end" zones (right of verb) are illustrated in Figure 9. In the first rule, the rule tag, i.e., "porder.end.vdep.comp" specifies the rule type, the zone ("end"), and left and right slots ("vdep" - the superordinate slot "verb dependent," "comp" - complement). The body of the rule places applicability conditions on the left and right slot fillers, and gives a weight, which should specify the relative importance of the rule. In the last rule of the example, a separately coded rule "roleprec" (role precedence) is referenced. This would be supplied with default arguments (the head and the two dependents being compared), and in this case check whether the dependents are in a sequence consistent with the "systemic" ordering of dependent roles for the language, as discussed by Hajicova (1989).

These rules can be applied in order of decreasing weight, to avoid (significant) conflicting orderings. Before each rule is applied, the already established pairwise orderings within the zone are analysed to determine whether they imply a narrowing of areas of indecision to some nested subzones. This would be true if one or more constituents have been found which are totally ordered relative to each other and to the zone boundaries, thus establishing some subzone boundaries, and extant pairwise orderings place all other constituents within one of those subzones. This information is obtained via graph analysis algorithms similar to those widely used for program flow analysis in compilers (Hecht and Ullman 1975).

After this analysis, the next most important preferential rule is then applied, but only to pairs which: (a) lie in the same subzone, and (b) have not yet been ordered relative to each other. The new relationships established are again analyzed to find smaller subzones. This analysis can result in some contradictory orderings, but these can be diagnosed and the conflict removed. When no further rules apply, remaining partial orders are converted to absolute ones, arbitrarily.

While this method should obtain acceptable orderings, it is not really satisfactory. For one thing, the rules are stated independently and some placement decisions may impinge on others. More generally, the sequentiality is suspect. A preferable approach would take into account multiple weighted considerations, and find an ordering which best satisfies all of them. Further work is needed in this area.

The same could be said for the use of these preferential ordering rules in parsing. As discussed in connection with zone placement rules, the basic approach to be taken is to find a consistent subset of the rules which together "explain" the placements, and, in so doing, determine some important topic/focus aspects of the sentence.

Before completing the discussion of relative ordering, it should be mentioned that a need has been found for strict adjacency as well as precedence rules, at least for handling focus adjuncts ("only," "just").

8. Directions for Further Work

In the preceding sections we have identified a number of directions for further work relating to ordering, in connection with obtaining "optimal" orderings by generation, and with using preferential ordering rules during analysis for disambiguation and textual feature identification. In addition, many aspects of Slot Grammar beyond the basic ones discussed here must be analyzed for bi-directional use, such as provisions for coordination. Also, issues such as pronominalization, tense/aspect mapping, and quantifier scoping, which do not have an explicit counterpart in Slot Grammar (although there has been Slot Grammar related work in these areas) should be examined in the SSG context.

One other major aspect not yet discussed should be addressed: the selection of alternative basic clause structures based on textual considerations, including, for example, decisions with respect to clefting, and subject raising ("John seems"). These require additions to the mechanisms so far discussed, as they imply changes to the distribution of material among complements.

This subject has been treated extensively by Johnson (1988a, 1988b), and it should be possible to adapt the mechanisms described to the current framework. A partial approach is to link lexicon subentries for such alternatives (e.g., for "seems"), with one subentry identified as canonical (i.e., to be used at the transfer interface), and with the other subentries containing mappings to/from the canonical form.

But the full decision-making process is a combinatorial one, in that these basic decisions can interact with decisions about the expression of individual complements, and about ordering, to produce many different textual effects. An important design question relates to the amount of anticipation to be attempted, i.e., whether the rules governing these first-stage decisions should try to anticipate related considerations, or whether several possibilities should be explored, together with some method of ranking the results. A somewhat tantalizing possibility is to exploit the bi-directionality of the grammar by parsing alternative generation results and measuring the closeness of the textual features obtained to those of the input.

9. Concluding Remarks

We have outlined parts of an adaptation of Slot Grammar for bi-directional use. The major adjustments relate to mappings between surface and transfer input/output forms, and to provisions for ordering based on preference rules. The adaptation is incomplete and insufficiently tested. Nevertheless, it may serve to illustrate the potential of Slot Grammar, and of head-driven grammars in general, for bi-directionality and to suggest possibilities for further work in this area.

10. References

1. Bresnan, J., "The Passive in Lexical Theory," in J. Bresnan, ed., *The Mental Representation of Grammatical Relations,* MIT Press (1982)
2. Buseman S., C. Hauenschild, "A Constructive View of GPSG or How to Make It Work," *Proc COLlNG 88,* 77-82
3. Caraceni, R., O. Stock, "Reversing a Lexically Based Parser for Generation," *Applied Artificial Intelligence,* vol. 2, # 2 (1988) 149-74
4. Dymetman M., P. Isabelle, "Reversible Logic Grammars for Machine Translation," *Proc 2nd Int'l Conf on Theoretical and Methodological Issues in the Machine Translation of Natural Languages* (1988)
5. Gazdar, G., E. Klein, G. Pullum, I. Sag., *Generalized Phrase Structure Grammar,* Basil Blackwell (1985)
6. Hajicova *E.* "A Dependency-Based Parser for Topic and Focus," *Proc. Intn'I Workshop on Parsing Technologies* (1989) 448-457
7. Hecht, M.S., and J.D. Ullman, "A Simple Algorithm for Global Data Flow Analysis Programs," *SIAM J. Computing,* Vol 4 No 4, (1975)
8. Jacobsen, S., *Adverbial Positions in English,* AB Studentbok, Stockholm, 1964
9. Johnson, D. "The Design of Post-Analysis in the JETS Japanese/English Machine Translation System," *Proc. Intn'l Conf on Fifth Generation Computer Systems* vol. 3 (1988a) 1150-1158

10. Johnson, D. "Genie: a Transportable English Generator for Machine Translation." IBM Tokyo Research Laboratory Report TR87-1023 (1988b)
11. Kasper, R. T. "An Experimental Parser for Systemic Grammars," *Proc COLING 88,* 309-312
12. Kartunnen L., "Radical Lexicalism," CSLI Report CSLI-86-68 (1986) *Proc COLING 88,* 359-364
13. McCord, M.C., "Slot Grammars," *Computational Linguistics,* vol 6, 31-43 (1980)
14. McCord, M.C. "A New Version of Slot Grammar," IBM Research Report RC 14506 (1989a)
15. McCord, M.C. "A New Version of the Machine Translation System LMT," IBM Research Report RC 14710 (1989b), to appear in *Proc. Internationa/ Scientific Symposium on Natural Language and Logic,* Springer Lecture Notes in Computer Science
16. Mel'cuk, I., A. Polguere. "A Formal Lexicon in Meaning-Text Theory (or How to Do Lexica with Words)," *Computational Linguistics,* Vol 13, Nos 3-4, July-Dec 1987
17. Newman, P. "Combinatorial Disambiguation," *Proc. ACL Conf. on Applied NLP* (1988)
18. Newman, P. "Towards Convenient Bi-Directional Grammar Formalisms," To appear in *Proc. COLING 90*
19. Pollard, C. and I. Sag, *Information-Rased Syntax and Semantics,* Center for the Study of Language and Information, Stanford, 1987
20. Quirk, R., S. Greenbaum, G. Leech, J. Svartnik, *A Grammar of Contemporary English* Longman (1972)
21. Sgall, P, E. Hajicova, J. Panevova, *The. Meaning of the Sentence in its Semantic and Pragmatic Aspects,* Reidel (1986)
22. Shieber, S. *An Introduction to Unification Rased Approaches to Grammar* CSLI (1986)
23. Wesche, B., I. Renz, "Word Order and Focus Projection," LILOG Report 13 (1987)