A Contrastive Study of Functional Unification Grammar for Surface Language Generation: A Case Study in Choice of Connectives

CUCS-487-89

Kathleen R. McKeown and Michael Elhadad
Department of Computer Science
450 Computer Science Building
Columbia University
New York, N.Y. 10027

August 1989

To be published in the proceedings of the Fourth International Conference on Natural Language Generation.

A Contrastive Evaluation of Functional Unification Grammar for Surface Language Generation: A Case Study in Choice of Connectives

CUCS-487-89

Kathleen R. McKeown and Michael Elhadad
Department of Computer Science
450 Computer Science Building
Columbia University
New York, N.Y. 10027

1 Introduction

Language generation systems have used a variety of grammatical formalisms for producing syntactic structure and yet, there has been little research evaluating the formalisms for the specifics of the generation task. In our work at Columbia we have primarily used a unification based formalism, a Functional Unification Grammar (FUG) [Kay 79] and have found it well suited for many of the generation tasks we have addressed. Over the course of the past 5 years we have also explored the use of various off-the-shelf parsing formalisms, including an Augmented Transition Network (ATN) [Woods 70], a Bottom-Up Chart Parser (BUP) [Finin 84], and a Declarative Clause Grammar (DCG) [Pereira & Warren 80]. In contrast, we have found that parsing formalisms do not have the same benefits for the generation task.

In this paper, we identify the characteristics of FUG that we find useful for generation. Of the following general criteria we have used in evaluating language generation systems, we focus on *order of decision making* and its impact on *expression of constraints*:

1. *Input Specification:* Input to a surface language generator should be semantic, or pragmatic, in nature. Ideally, few syntactic details should be specified as these should be filled in by the surface generator, which contains the syntactic knowledge for the system. Furthermore, some flexibility should be allowed in what must be provided as input; not all pragmatic or semantic features may always be available for each input concept and the surface generator should be able to function in their absence. Finally, input should be kept simple.
2. *Expression of constraints on decision making:* One main task of a language generator is to make decisions about the syntactic structure and vocabulary to use. Such decision making is done under constraints and the ability to clearly and concisely represent constraints is important [McKeown & Paris 87]. If these constraints can be represented declaratively, without duplication, clarity of the system is improved.
3. *Order of decision making:* The order in which decisions must be made and the interactions between them has an impact on representation of constraints. If decisions must be made in a fixed order, representation of constraints on those decisions may become more complex. The order of processing, bottom-up, top-down, left to right, or any other variation, can significantly influence how constraints interact.
4. *Efficiency:* As in any interactive system, an efficient, speedy response is desirable. At this point in time, most grammatical systems can provide a response in reasonable real time. In fact, in practice there doesn't appear to be significant differences in run time between a deterministic surface generator such as MUMBLE [McDonald 86] and unification based processors such as FUG [McKeown & Paris 87].

5. *Reversability:* Ultimately, a natural language system that uses only one grammar both for parsing and generation is desirable. Using a reversible grammar means that syntactic knowledge need not be duplicated for the two tasks. In reality, however, a grammatical formalism has usually been developed with one task or the other in mind. Due to the differences in focus between the two tasks, when a formalism is adopted for the other task, the match is often not ideal. For example, when FUG is used for interpretation, an additional, rather complex chart must be supplied [Kay 79]. On the other hand, when grammars originally developed for interpretation are used for generation, points 1-3 often can not be achieved easily, as we shall attempt to show.

Our claim is that order of decision making in FUG through unification allows for flexibility in interaction between constraints. This, in turn, allows for a more concise representation of constraints. To illustrate these properties of FUG, we use the task of selecting a connective (e.g., but, however, nonetheless, since, because, etc.) to conjoin two input propositions. Connective selection is a subset of the lexical choice problem. Lexical choice has been shown to require complex interaction between constraints [Danlos 87, Danlos 88], and connective selection in particular contains a number of challenges particular to generation.

Our goal in this paper will be to show the advantages of the following main features of order of decision making in FUG:

- *Not strictly left-to-right:* In FUG, all decisions that can be made at the top-level are made before producing constituents. These decisions can send constraints down to lower levels if necessary. Thus some decisions about later sentence constituents can be made before decisions about prior constituents in the sentence. This is important when a decision made early on in the sentence depends on a decision made later.
- *Bidirectional:* Specifying dependence of a decision on a constraint automatically specifies the inverse because of the use of unification: if the constraint is unspecified on input it will get filled in when the otherwise dependent decision is made.
- *Interaction between different types of constraints is determined dynamically:* How different constraints interact can be determined at run-time depending on the current context of generation. This means the grammar can be modularized by constraints with specific interaction left unspecified. In contrast, the parsing formalisms synchronize in lock-step the influence of different constraints as they proceed through the construction of syntactic structure making their representation difficult.

In the following sections we first give an overview of FUG, showing how decision making is carried out for a simple grammar. We then introduce the problem of connective choice, describing the constraints on choice and the type of decision making required. We show how the basic characteristics of FUG lend themselves to the implementation of connective choice.

Finally, we make comparisons with other formalisms. In particular, we note that control strategies developed for parsing formalisms lack the flexibility FUG provides. Our more general position is that, while reversability of grammatical processors is definitely a worthwhile aim, a syntactic processor that was originally developed for parsing may not be ideal for generation. This results partially from the fact that control of processing is driven in part by the input sentence, or word order, in interpreting language. Emphasis is on using the input to determine which grammatical rules to apply next. In contrast, in generation, there is no need to select words as they appear in a sentence. In fact, many systems determine the verb of the sentence first as this can control the assignment and syntactic structure of the subject and object (e.g., MUMBLE [McDonald 86]). Part of our goal in identifying the problems in using a parser for generation is to point out some of the

characteristics that are useful for generation so that they can be taken into account when future reversible syntactic processors are designed.

Despite our overall preference for FUG, there are certain tasks in selecting connectives that are difficult to represent in FUG, but which can be easily accommodated in other formalisms and we note these in our conclusion.

2 Overview of FUG

The main characteristic of FUGs ([Kay 79, Shieber 86]) is that all information is uniformly described in the same type of structure - the functional description (FD). An FD is a matrix of attribute-value pairs (called features). Both the input and the grammar are represented as FDs. The only mechanism allowed when dealing with FDs is unification. Intuitively, the unification of two FDs consists of building a larger FD that comprises both input FDs and is compatible with both. Crucial features of the process are that it is (1) independent of the order of features in the input FDs, (2) bidirectional, (3) monotonic and (4) completely declarative - a grammar being best viewed as a set of constraints to be added to or checked against an input.

The unification algorithm begins by selecting the syntactic category from the input and unifying the grammar for that category with the input. Unification is controlled by the grammar and basically consists of checking grammar attribute value pairs of this category against the input. If a grammar attribute does not exist in the input, the grammar attribute value pair is added to the input. If the attribute does exist, the grammar and input values for this attribute are unified, and the results added to the input. This stage of unification can be characterized as a breadth first sweep through the top level category adding restrictions governed by this category. Following this stage, each constituent of the resulting FD is in turn unified with the grammar in the same way. Thus at this next stage, unification results in successive refinement of embedded constituents. The constituents that are to be unified are specified by the special attribute CSET (for Constituent Set) and the order in which they occur need not necessarily be the order in which they will occur in the resulting sentence. Again, this means that decision making is top-down but not necessarily left-to-right. A further distinction is that all decisions get made at the top level before moving to embedded constituents.

To see how order of decision making occurs in FUG, consider the unification of a sample grammar (Figures 1, 2, and 3) and input (Figure 4).[1] This grammar is a small portion of the clause category of a larger grammar we are currently using [Elhadad 88] and is based on the systemic grammar described by Winograd [Winograd 83]. This portion will generate either action sentences (e.g., "John gives a blue book to Mary.") or attributive sentences (e.g., "This car is expensive."). Note that input to this grammar is specified semantically with the exception that the input must specify the type of phrase we are trying to generate.

The grammar for the clause category is divided into three sections. The first section (Figure 1) specifies how syntactic features get added to semantic roles depending on the semantic type of the clause being generated. Thus, in the sample grammar we see that the protagonist role (prot) of an action sentence is specified as an np, while the attribute role of an attributive sentence is specified as either adjective or np. The second section (Figure 2) identifies the voice-class of the verb and, according to the chosen voice, the grammar determines how the semantic cases are mapped into the syntactic roles, subject, object, and indirect object. Finally, in the

[1] See [Kay 79], [McKeown 85], [Appelt 85] for more details on FUG.

```
;;=============================================================
;; 01 CAT CLAUSE : clause ---------------------------------------
;;=============================================================
((cat clause)
 (alt
  (
   ;; Process-type: action, mental, or relation
   ;; ---------------------------------------

   ;; Process 1: Action --> actions, events, natural phenomena
   ;; inherent cases    --> prot, goal, benef.
   ;; all are optional, but at least one of goal or prot must be present.
   ;; this will be dealt with in the voice alternation.
   ((process-type actions)
    (prot ((cat np) (animate yes)))
    (goal ((cat np)))
    (benef ((cat np)))
    (verb ((process-class actions)
           (lex any))))                    ;there must be a verb given

   ;; Process 3: relation --> equative, attributive
   ;; there need not be a verb, it will be determined by the
   ;; epistemic modality features among the possible copula.
   ((process-type attributive)
    (verb ((process-class attributive)))
    ;; so far all we do if the verb is not given use "be"
    ;; later use modality...
    (opt ((verb ((lex "be")
                 (voice-class non-middle)
                 (transitive-class neutral)))))
    ;; inherent cases     --> carrier, attribute
    ;; both are required.
    (carrier ((cat np) (definite yes)))
    ; attribute can be a property or a class
    ; like in "John is a teacher" or "John is happy".
    (alt
      (((attribute ((cat adj) (lex any))))
       ((attribute ((cat np)  (definite no)))))))))
```

Figure 1: Sample FUG -- Section 1

third section (Figure 3), the syntactic roles are arranged linearly through the use of patterns.

These sections are represented by three large alternatives (alt) in the grammar.[2] Output is produced by successively unifying each of these alternatives with the input, thus adding the constraints from each section. This grammar thus implements Kay's [Kay 79] suggestion that the semantic and syntactic grammar be represented separately and unified to produce output.

In unifying input T2, Figure 4, with the clause grammar, section 1 which specifies constraints associated with the clause's semantic category, is unified first. Since it consists, itself, of alternatives representing each

[2]Alternatives are a special construct of FUGs. They represent a disjunction of possibilities. To unify an FD F with an alternative (alt (fd1 fd2 ... fdn)), the unifier tries to unify each of the "branches" of the alternative (*i.e.*, the fdis). The result is the disjunction of the successfully unified branches. In our implementation, only the first branch that can be unified with F is returned.

```
;; Voice choice --> operative, receptive, middle.
;; Operative is rougly  active.  Receptive, roughly passive.
;; Middle     = sentences with only one participant ("the sun shines")
;; Choice middle/non-middle is based on verb classification
;; Choice receptive/operative is based on focus (using pattern).
;; The voice alternation does the mapping semantic case -> syntactic roles
(alt
   (((voice operative)
     (verb ((voice-class non-middle)
            (voice active)))
     (alt
       (((process-type actions)          ;; The notation (subject (^ prot))
         (subject (^ prot))              ;; is to be read as the equation:
         (object (^ goal))               ;; the value of subject must be the
         (iobject (^ benef)))            ;; same as the value of the path
        ((process-type attributive)      ;; (^ prot).
         (subject (^ carrier))           ;; (^ prot) is a relative path, giving
         (object  (^ attribute))         ;; the address of another pair starting
         (iobject none)))))              ;; at the current position. ^ means
     ((voice receptive) <...>)           ;; go up one level (to the embedding
     ((voice middle)    <...>)))         ;; pair).
```

Figure 2: Sample FUG -- Section 2

```
;; General things: arrange syntactic roles together
;; and do the agreements.
;; The patterns are here of course.

; Focus first (change when add modifiers)
(pattern ((* focus) dots))

; Arrange order of complements
(pattern (subject verb dots))
(alt
  ; VERB VOICE ACTIVE
  (((verb ((voice active)))
    (alt
      (((object none)
        (iobject none))
       ; John gave the book
       ((verb ((transitive-class bitransitive)))
        (iobject none)
        (pattern (dots verb object dots)))
       ; John gave Mary the book
       ((verb ((transitive-class bitransitive)
               (dative-prep none)))
        (pattern (dots verb iobject object dots)))
       ((iobject none)
        (pattern (dots verb object dots)))
       ((verb ((dative-prep any)))
        (dative ((cat pp)
                 (prep ((lex (^ ^ ^ verb dative-prep))))
                 (np (^ ^ iobject))))
        (pattern (dots verb object dative dots))))))
```

Figure 3: Sample FUG -- Section 3

possible semantic clause type (in this grammar, either action or attributive process types), the first step is

```
T2 =

((cat clause)
 (process-type actions)
 (prot ((lex "John")
        (np-type proper)))
 (goal ((lex "book")
        (np-type common)
        (definite no)
        (describer === "blue")))
 (benef ((lex "Mary")
         (np-type proper)))
 (verb ((process-class actions)
        (voice-class non-middle)
        (transitive-class bitransitive)
        (dative-prep "to")
        (lex "give"))))
```

Figure 4: Sample Input

selecting one of these alternatives. Since the input includes the attribute value pair **(process-type action)**, the first alternative matches. Unification of this alternative results in the addition of the italicized lines in the FD shown in Figure 5. At this point the syntactic categories of each semantic role have been determined and some further features added. The unifier now proceeds to the second section based on voice class. At this point, **(voice operative)** will be selected because no voice is specified in the input and there are no incompatibilities between the **(voice operative)** alternative and the input. Later on, this choice will be confirmed or rejected by the focus constraint. The result from this unification is the addition of the underlined lines in Figure 5 and the FD now contains the mapping of semantic roles to syntactic roles. Finally in unifying the third section of the clause grammar with the input, order of syntactic constituents is determined. This is done in two steps. First the constraint from focus is added **(pattern ((* focus) dots))**, stating that focus must occur first.[3] In the second step, syntactic constraints on order are added, namely that subject must occur first **(Pattern (subject verb dots))**. At this point, subject is unified with focus and if they are not the same, the unifier would retract its earlier decision of **(voice operative)** and select **(voice receptive)** instead. In this example, subject and focus both refer to the protagonist and the remaining syntactic details for the active voice are filled into the grammar, specifying the order of the object and indirect object. This results in the addition of the last lines in small caps to the FD in Figure 5 and the FD is linearized as "John gives the blue book to Mary.", following unification of its constituents, **prot, goal**, and **benef.**[4]

[3]The * indicates that this element of the pattern must be unified with the element of some other pattern. This feature is not standard in Kay's formalism and was added to increase efficiency.

[4]Note also that the three patterns in small caps are actually unified by a special pattern unifier to produce the single pattern: `(pattern (subject verb object dative dots))` plus the following conflation constraint `(focus (^ subject))` derived from the unification of the first two patterns.

T2 after unification with the grammar:

```
((cat clause)
 (process-type actions)
 (prot ((lex "John")
        (np-type proper)
        (cat np)
        (animate yes)))
 (goal ((lex "book")
        (np-type common)
        (definite no)
        (describer === "blue")
        (cat np)))
 (benef ((lex "Mary")
         (np-type proper)
         (cat np)))
 (verb ((process-class actions)
        (voice-class non-middle)
        (transitive-class bitransitive)
        (dative-prep "to")
        (lex "give")
        (voice active)))
 (voice operative)
 (subject (^ prot))
 (object (^ goal))
 (iobject (^ benef))
 (PATTERN ((* FOCUS) DOTS))
 (PATTERN (SUBJECT VERB DOTS))
 (DATIVE ((CAT PP)
          (PREP ((LEX (^ ^ ^ VERB DATIVE-PREP))))
          (NP (^ ^ IOBJECT))))
 (PATTERN (DOTS VERB OBJECT DATIVE DOTS)))
```

Figure 5: After unification of the clause level

3 Choice of Connective: an Example

Choosing a connective (e.g., "but," "although") is a task particular to language generation that requires flexibility in the order that decisions are made and thus, we argue FUG is well suited to represent constraints on connective choice. The need for flexibility arises because the features used to select a connective also have an influence on other aspects of generation. Therefore, there is interaction between the selection of a connective and the generation of the connected propositions. There are two types of interaction that can occur:

- *External:* mutual interaction is necessary between the deep and surface components of the generation, and the order of decision making must be left as flexible as possible between a surface generator and its environment. For example, choice of a connective can influence what can be said next. Conversely, what must be said next can influence the choice of a connective. Similarly, [Danlos 88, Danlos and Namer 88] argues that the morphological component of a generation system may have to interact with a deeper component to decide on pronominalization. For example, in French, when deciding whether to pronominalize "Marie" in "Jean aime Marie," the fact that the pronoun "la" would be elided in "Jean l'aime" and therefore loses the distinctive marks of gender, may introduce an ambiguity in the reference. It is therefore necessary to allow the morphological component to interact with other components, dealing with semantic and rhetoric issues.

- *Internal:* there is complex interaction between surface decisions. Therefore order of decision *within* the grammar must be left as flexible as possible. For example, the choice of an adjunct can precede and influence the choice of the verb in a clause, and vice-versa.

In this section, we illustrate interaction between internal decisions of the grammar through an example of lexical choice. In this case, lexical choice in an embedded clause can influence the choice of connective and choice of connective can in turn influence lexical choice of the clause. Thus we have a propagation of constraints from connective down to the clauses it connects and as well, we have a propagation of constraints from the decision made in the clause back up to choice of connective. Constraints between the two constituents are bidirectional and decision making must not be strictly left to right. We illustrate external interaction by showing how different connectives chosen to conjoin the same two clauses allow for different follow-up sentences. To illustrate these two cases, we describe the features that play a role in connective selection, give an example of the two cases of interaction, and describe our implementation of connective selection in FUG through this example.

However, since choosing a connective to link two propositions is a subset of the problem of lexical choice, a problem in language generation that has raised questions about modularization and order of decision making in generation, we first briefly survey previous work on lexical choice.

3.1 Previous Work in Lexical Choice

The place of word selection within a complete generation system is a controversial topic. Several options have been put forth in recent work:

During Surface Generation: Many previous systems position the task of lexical choice as part of the component that does surface language generation. One class of such systems [McDonald 86, McKeown 85, Paris 87] use a dictionary based on Goldman's [Goldman 75] system. The dictionary is keyed by internal concepts for which a word or phrase must be chosen (for Goldman these were conceptual dependency primitives such as *ingest*) and each entry contains a discrimination net which makes tests on various features to determine the word or phrase to use. In MUMBLE [McDonald 86], the dictionary is accessed from the grammar in the process of building syntactic structure. Thus lexical choice is interleaved with syntactic choice. Since syntactic structure is built by constructing and traversing the syntactic tree in depth-first traversal, words will typically get selected in left-to-right order. There are some exceptions. For example, MUMBLE selects the verb of the sentence first. In other systems (e.g., [McKeown 85, Paris 87]) all necessary dictionary entries are accessed and lexical choices made before the grammar is invoked.

In NIGEL [Mann & Matthiessen 83], the lexicon is only accessed after the grammar has completed its task. Sets of semantic features are used where lexical items would occur and are sufficient for making syntactic choices. Semantic features get added as the grammar systems make choices. After all syntactic choices have been made, the lexicon is accessed to replace each set of features with a lexical item. A lexeme may be preselected (by the deep generator for example) or directly chosen by the grammar (through a `lexify` realization statement). In the latter case, it would provide constraints on other choices. Systemicists term lexical choice as "the most delicate" of decisions as it is represented at the leaves of grammatical systems.

As part of content decisions: Another class of generation systems positions the task of lexical choice as occurring somewhere during the process of deciding what to say, before the surface generator is invoked. This

positioning allows lexical choice to influence content and to drive syntactic choice. Danlos [Danlos 87] chooses this ordering of decisions for her domain.[5] She makes use of a *discourse grammar* that identifies possible discourse organizations along with the lexical choices that can be used for each organization. Thus lexical choice and order of information are decided simultaneously before other decisions, such as syntactic choice, are made. Systems using phrasal lexicons (e.g., [Kukich 83, Jacobs 85]) are similar in that they select whole phrases fairly early on in the generation process and the phrases in turn control syntactic choice. In these approaches, emphasis is on idiomatic phrases whose usage is very tightly tied to content in a particular domain. For example, in the stock market domain in which Kukich works, the use of a particular phrase has a very specific meaning and thus choice of a phrase can determine the content conveyed.

Other researchers advocate folding the lexicon into the knowledge representation. In this approach, as soon as a concept is selected for the text, the lexemes associated with it in the knowledge base would automatically be selected as well. One variation on this approach is presented by [Matthiessen 81] who represents the semantic structure of the lexicon as intensional concepts in a KL-ONE [Brachman 79] style knowledge base. His approach provides for links between the syntactic structure of the lexicon and the semantic structure, showing how, for example, the semantic role of AGENT might function as the syntactic role ACTOR, if the semantic concept for SELL were lexicalized using the verb "to sell."

Specifying Interaction: More recent work aims at specifying the type of interaction that can occur between the two components, rather than merging them. For example, Hovy [Hovy 86] specifies five points of interaction between conceptual and linguistic decisions; processing is controlled primarily by the surface generator, with the conceptual component being invoked at predetermined points. Work presented at this workshop [Rubinoff 88, Iordanskaja et al 88, McDonald 88] also looks at the types of interaction that must occur.

3.2 Constraints on Connective Selection

Before presenting our implementation of connective choice, we first describe the information involved in the decision to use a connective and give an abstract model of the selection procedure. Connectives are functionally defined as the class of words that express a relation between two (or more) utterances or discourse segments. There are many types of relations that can hold between discourse segments, and a given connective can often express more than one relation. In this paper, we look at two relations that connectives can express: a relation between the *argumentative orientation* of the conjoined utterances and the *functional status* of the utterances. We limit our discussion to the connective "but."

Traditional definitions of "but" [Quirk *et al* 72] indicate that the complex "*p but q* " expresses opposition between *p* and *q* as illustrated in (1) below. However, whatever semantics is given to the concept of *opposition*, it seems unlikely one would maintain that *p* and *q* of (2) can be in opposition, for it is well accepted in our society that beauty deserves a high price. We are likely to agree that the fact that an object is beautiful implies that it is expensive, thus indicating that *p* and *q* of (2) are more in "agreement" than in "opposition."

---

[5]Note, however, that her analysis of interactions between constraints on lexical choice and other decisions leads her to conclude that no general principles specify interaction between conceptual and surface decisions. For each new domain a new ordering must be developed.

(1) I want to buy it, but it is expensive.
(2) It is beautiful, but it is expensive.

Although we do maintain that "but" expresses an opposition between the two units it connects, the problem is to determine exactly what is opposed in the two utterances and to define precisely what is meant by "opposition". Ducrot [Ducrot 83] offers some clues. In *p but q*, *p* is presented as an argument for a certain conclusion *c1* and *q* as an argument for another conclusion *c2*. It is these two conclusions that need to be in opposition.

In our example, the opposition between *p* and *q* is **Indirect** and requires the identification of **Implicit conclusions**. Such conclusions could be:

A: It's beautiful → *I want to buy it*
B: But it's expensive → *I don't want to buy it*

We call the set of conclusions compatible with an utterance its **argumentative orientation (AO).**[6] It is now possible to rephrase the description of 'but' as: 'but' indicates an opposition between the AO of the units it connects. If we consider the conclusions aimed at by utterances as formulae of a first order language[7], we can define opposition between the conclusions *c1* and *c2* each represented by single formula as simply: $oppose(c1,c2) \equiv (c1 \rightarrow \neg c2) \equiv \neg(c1 \wedge c2)$ and between the conclusions *AO1* and *AO2* each represented as sets of formula: $oppose(AO1,AO2) \equiv (AO1 \cup AO2)$ *is inconsistent*

In order to rank arguments so that comparisons can be made, we use the notion of **argumentative scale** [Anscombre & Ducrot 83]. An argumentative scale is simply an ordering between propositions that can be dynamically established during discourse. Naturally, several scales exist in a given situation. To allow comparisons across different scales, Anscombre and Ducrot [Anscombre & Ducrot 83] use the notion of *topos* originally proposed by Aristotle. *Topoi* can be viewed as conventional argumentative scales underlying communication. They can be represented as gradual inference rules of the form *"the more/less P, the more/less Q"* where P and Q are arbitrary formula along a scale. Using these tools, *Q* can be determined to provide a better argument than *P* if it falls higher on the scale. A very similar formal apparatus is described in [Kay 87].

Functional status is another feature that we have found plays a role in connective selection. It indicates whether the unit is directive (i.e., makes the primary point of the complex clause) or subordinate. The functional status of individual units is an essential component of the representation of discourse structure as it indicates how units are related. Different connectives constrain the functional status of the units they conjoin in different ways. For example, in "P but Q" Q is the directive act, in "Although P, Q" Q is the directive act,

[6]To say that all utterances have an AO is not a proposal to cast all discourses as argumentative. The AO can be thought of as the set of inferences that can be drawn from a given proposition. For some utterances, the AO can be unconstrained by the linguistic form. The point is that there are certain linguistic devices whose primary function is to constrain the AO of an utterance. Therefore, the notion of AO is necessary to describe the semantic value of these devices. For example, the role of words like 'even' [Fraser 71, Anscombre 73, Nolke 83, Kay 87],'almost' [Sadock 81], 'only' [Horn 69], 'let alone' [Fillmore, Kay & O'Connor 87] or of many of the connectives we have studied can be described as adding constraints on the AO of the sentences they modify.

[7]Note that conclusions are not utterances or sentences of a natural language. They are part of the meta-language used to describe the meaning of an utterance.

and in "P because Q," P is the directive act.[8] Other features also play a role in the selection of a connective, but we do not discuss them here. See [Elhadad & McKeown 89] for a full description.

We represent a connective as a *relation between the features of two consecutive utterances*. Therefore, in order to produce a connective, a generator is provided with a set of features for each utterance as input. If these utterance features satisfy a relation, the corresponding connective can be produced. It must be noted that in this model, we go from connectives to relation, and not from relations to connectives: we do not need to establish a classification of possible relations between discourse segments, but consider only those relations that can be realized by certain connectives. Note also that this approach provides a *multi-dimensional* description of the types of relations that can be expressed between discourse segments.

As an example, consider the description of 'but'. It is a set of constraints between the features of two utterances *P* and *Q*. For the argumentative and functional status part, it specifies that:

- The argumentative orientations of *P* and *Q* must include ordering constraints involving the same scale, and the proposition mentioned in *P* must have a lesser degree on this scale than the one mentioned in *Q*. $(P_i \text{ of } \sigma) \in AO(P)$ *and* $(Q_j \text{ of } \sigma) \in AO(Q)$ *and* $(P_i <_\sigma Q_j)$ *where* σ *is a scale.*
  If this is not the case, it is difficult to explain why the locutor supports the conclusions of *Q*. For example, if there is no scale in common between the argumentative orientations of *P* and *Q*, the opposition is difficult to understand, as in "John is hungry but he is short." If there is a scale in common, but *Q* has a lesser degree than *P* then the preference of the locutor is difficult to directly understand, like in "John is starving but Mary is hungry."[9]
- The topoi used for *P* and *Q* must have their right-hand sides of different polarities: if $Topos(P) = (..., +\sigma)$ then $Topos(Q) = (..., -\sigma)$ and *vice-versa*. This explains the opposition between the argumentative orientations of *P* and *Q*. For example, in "this car is nice but it is expensive," one interpretation would use the topoi *"+nice,+desirable"* and *"+expensive,-desirable"*.
- *P* must have a subordinate status and *Q* a directive status. This constraint accounts for the fact that one must link on *Q* and not on *P* after the complex *P but Q*. For example, the combination "this car is nice, but it is expensive. Therefore I will buy it" is (in most situations) not acceptable, because "therefore" links on the argumentative features of *P* and not on *Q*.

3.3 An Example of Constraint Interaction

An example of interaction between internal constraints occurs in lexical selection. Many adjectives are conventionally associated with argumentative scales (*e.g.*, "small" is associated with the scale of size). Similarly, verbs often 'project' an argumentative aspect on their actants (*e.g.*, "steal" positions its actor on the scale of honesty, as described in [Raccah 87]). These features of words are described in a lexicon. When a connective is chosen, the values of the argumentative features (AO and Topos) are constrained. As a consequence, the verbs and adjectives chosen in the connected clauses are also constrained.

---

[8]Note that the notion of directive vs. subordinate does not coincide with the more classical notions of coordination vs. subordination (cf [Quirk *et al* 72] for a grammatical definition of the gradient coordination/subordination), or to the systemist notion of taxis [Halliday 85]. For example, the complex "p but q" is grammatically a conjunction and is defined in [Halliday 85] as a paratactic relation. But in our analysis, in "p but q," p and q do not have the same functional status. In the structure of the discourse, q is more accessible than p - and for example, "p but q, therefore c" is only possible if c is argumentatively compatible with q, not with p.

[9]The opposite case "Mary is hungry but John is starving." could be interpreted using a scale along the degree of hunger. The two propositions would be connected using a topos such as "the more X is hungry, the more X has priority for food" and the opposition appears between the two specializations of the right-hand side "John has priority for food" and "Mary has priority for food."

For example, consider the case where a text generation system wants to convey that a transaction has occurred between two participants, a construction company and the mayor, involving an exchange of money in return for a permit to build. Suppose, furthermore, that the system wants to convey its opinion that the two participants acted dishonestly (i.e., that the transaction was in some way not legal). The propositional content for such an utterance and the AO might be represented as shown in Figure 6. If this utterance is to be generated in isolation, the AO must be realized through appropriate lexical choice, for example resulting in the choice of "bribe" for the *exchange* predicate as in sentence (3).

(3) A&C Builders bribed the Mayor with $10,000 to receive a license for the new construction site.

On the other hand, if this utterance is to be conveyed as part of a discourse segment where the AO is attributed to someone other than the system (i.e., someone thinks the participants have acted dishonestly[10]), and is followed by the statement that the transaction is, in fact, legal, then the system can choose to express the AO through the selection of the connective "but". In this case, the lexical choice for *exchange* is no longer constrained to be semantically loaded. The system can choose a more neutral verb to express the exchange such as "buy" as in sentence (4).

(4) A&C Builders bought a license for the new construction site from the Mayor for $10,000, but their intentions were honest.

In this sentence, the inference that the exchange could be considered illegal is triggered by the use of "but" since this connective indicates opposition between the AOs of P and Q.

Lexical choice in P is therefore affected by the decision to use a certain connective. The decision about what verb to use in an embedded clause is determined in part by the decision about a constituent to the right of the verb. Thus decision making must not be purely left to right. Conversely, if it happens that the lexical item "bribe" must be used (e.g., the propositional content of the clause contains the predicate *illegal-exchange* in place of *exchange*), that can in turn have an influence on the decision to generate a complex clause or two single ones. Thus constraints between these two constituents are bidirectional.

Furthermore, constraints made in selection of the connective may in turn place constraints on generation of content by a deep planner. For example, the use of "but" in the previous example allows the generation of different follow-up sentences than would have been the case if "although" had been generated. Since Q is directive in "P but Q", we can use a follow-up sentence such as "We should use them." following (4) above. In contrast, P is directive in "P although Q" and this explains the awkwardness of the sequence "A&C Builders bought a license for the new construction site from the Mayor for $10,000 although their intentions were honest. We should use them." A comprehensive analysis of the interaction between the features we use for connective selection and both deep and surface generation remains to be done. The point is that **a given linguistic device (*e.g.*, a connective) introduces more constraints on a discourse than those that motivated its use. Thus**, in selecting a linguistic device, a surface generator must be able to generate constraints on content that will be fed back to the deep generator.

---

[10]Other features of our definition of "but" account for the fact that the propositional content of *P* in *P but Q* can be attributed implicitly to someone other than the speaker(see [Elhadad & McKeown 88] for details).

```
Propositional content for (3):
   (pc ((cat clause)
        (process-type action)
        (concept Exchange)
        (benef/init ((lex "A&C Builders")
                     (concept A&CB)
                     (np-type proper)))
        (benef/react ((lex "Mayor")
                      (np-type common)
                      (concept Mayor)
                      (definite homophora)))
        (medium/init ((concept cash)
                      (quantity 10000)
                      (unit $)))
        (medium/react ((head ((lex license)
                              (concept license)
                              (np-type common)))
                       (qualifier ((prep === for)
                                   (concept CSite)
                                   (head === site)
                                   (classifier === construction)
                                   (describer ((lex new)
                                               (defining yes)))))))))


Argumentative Orientation for (3):
   (ao ((scale dishonest)
        (conclusion
         ((process-type attributive)
          (carrier ((concept A&CB)))
          (attribute ((concept dishonest)))))))
```

Figure 6: Sample input with argumentative constraint

3.4 Implementation

To illustrate how our FUG implementation accounts for this interaction among constraints, we present a simplified version of our full grammar, restricted to 'but' and 'although'. It expects as input an FD of category **discourse-segment**. A discourse segment, following [Roulet *et al* 85, Sinclair & Coulthard 75] is represented as a hierarchical structure, characterized by a directive act and subordinate acts. The directive act is a single utterance, while the subordinate acts recursively form a complex discourse segment. In this paper, for simplicity, we restrict subordinate acts to simple sentences.

The connective grammar is shown in Figures 7-10. In Figure 7, the grammar for discourse segments is shown and it specifies the possible orderings of the clauses and connectives. Whether a connective can be used freely in the initial position (*e.g.*, "although P, Q") is a property of the particular conjunction used. This is represented in the part of the grammar handling conjunctions (Figure 8). The feature **position** has a value of **middle** when the conjunction must be in between the two clauses it connects (*e.g.*, 'but') and of **free** otherwise. In the grammar for discourse segments, the feature **connective** is introduced and specified as category **connective**. This category is in turn defined in Figures 9 and 10. It expects two utterances P and Q as features, and describes the relation that must hold between them when the complex PcQ can be realized. It

is at the heart of our selection procedure. The clause section presented in section 2 is used for the generation of each simple clause.

To describe the relation that holds between P and Q, the connective grammar contains a separate alternative (alt) for each class of features. Functional Status (FS) forms one class and argumentative orientation (AO) and Topos forms the class, argumentation. The use of alternatives encodes the fact that there is no natural priority in the model between the different features. Furthermore, constraints from one class (e.g., FS) can be stated independently of those from another (e.g., argumentation). Again, the FUG implementation allows us to distinguish between the different types of constraints, and to localize related constraints in the same alt. This separation of constraints of different natures in different regions of the grammar is similar to the distinction we have in the **clause** grammar presented in section 2 between syntactic and semantic features. It allows internal flexible ordering of decision making mentioned on page 8. Constraints from functional status are represented in the first alt (Figure 9) and these generate most of the constraints on the ordering of the complex clause (i.e., whether P or Q is the first embedded clause). While the **discourse-segment** section of the grammar expresses all possible orderings, this section selects one based on which clause can be directive as governed by the particular conjunction. Conversely, if constraints on the ordering are generated by the deep planner, they are taken into account by the FS feature constraints. Figure 10 presents the part of the grammar handling argumentative features. We represent opposition between two AOs using topoi: two AOs are opposed if their respective scales appear in a topos with opposite signs (the feature **sign-right** of P must be the opposite of **sign-right** of Q).

```
;; ================================================
;; CAT DISCOURSE-SEGMENT ------------------------------------
;; ================================================
((cat discourse-segment)
 (directive ((cat utterance) (FS directive)))
 (subordinate ((cat discourse-segment) (FS subordinate)))
 (alt
   (((connective  ((cat connective)
                   (P (^ ^ directive))
                   (Q (^ ^ subordinate))))
     (alt
       (((pattern (directive connective subordinate)))
        ((pattern (connective subordinate directive))
         (c ((position free)))))))
    ((connective ((cat connective)
                  (P (^ ^ subordinate))
                  (Q (^ ^ directive))))
     (alt
       (((pattern (subordinate connective directive)))
        ((pattern (connective directive subordinate))
         (c ((position free)))))))))
```

Figure 7: The CATegory Discourse-segment

```
;; ==========================================================
;; CAT CONJ  ---------------------------------------------
;; ==========================================================
((cat conj)
 (alt
   (((position free)
     (alt
       (((lex "although"))
        ((lex "since"))
        ((lex "because")))))
    ((position middle)
     (lex "but")))))
```

Figure 8: Grammar for CONJunctions

```
;; ==========================================================
;; CAT CONNECTIVE ----------------------------------------
;; ==========================================================
((cat connective)
 ;; The parts common to all connectives
 (pattern (c))
 (c ((cat conj)))
 ;; Themes must intersect
 (TEST (FD-intersection @(^ ^ P Th) @(^ ^ Q Th)))

 ;; First alt: Functional Status
 ;; For but: S-D order, all other, D-S order.
 (alt
   (((P ((FS subordinate)))
     (Q ((FS directive)))
     (c ((lex "but"))))
    ((P ((FS directive)))
     (Q ((FS subordinate)))
     (alt (((c ((lex "although"))))
           ((c ((lex "because"))))
           ((c ((lex "since")))))))))
```

Figure 9: The Connective FUG -- Section 1

A sample input to the grammar is shown in Figure 11. Note that this input contains an argumentative constraint stating that the clause realizing this proposition must argue for the conclusion that A&C Builders and the Mayor were involved in a dishonest transaction. The rest of this input FD is primarily a description of the propositional content (the value of the feature pc). Note that the lexical specification for the verb is not given, but only specifies that the concept to be expressed is "exchange." Part of the task of the grammar is to choose a verb that will express this concept.

Figure 12 shows a fragment of the lexicon that helps map concept to verb. The fragment shows that the verbs

```
;; Second alt: Argumentation
;; AO has 2 (main) features: conclusion and scale.
;; Topos has 4 (main) features: sign and scale/left and right.
;; For ''although'', Topos(P) must be none,
;; because the opposition between P and Q must be direct,
;; following a pattern (P, P arg -Q) and not through an implicit
;; conclusion reached through a topos in P like in (P arg C, Q arg -C)
;; which is possible with ''but''

(alt
  (((P ((Topos none)))
    (alt
      (((Q ((Topos ((scale-left  (^ ^ ^ Q AO scale))
                    (sign-right -)
                    (scale-right (^ ^ ^ P AO scale))))))
        (c ((lex "although"))))
       <...other connectives...>)))
   ((P ((Topos ((scale-left  (^ ^ ^ P AO scale))
                (scale-right (^ ^ ^ Q Topos scale-right))))))
    (Q ((Topos ((scale-left  (^ ^ ^ Q AO scale))))))
    ;; sign-right of Topos(P) and Topos(Q) must be opposed
    (alt
      (((P ((Topos ((sign-right +)))))
        (Q ((Topos ((sign-right -))))))
       ((P ((Topos ((sign-right -)))))
        (Q ((Topos ((sign-right +))))))))))))
;; The AO of P and Q is justified by the use of the connective
(P ((AO ((justified yes)))))
(Q ((AO ((justified yes)))))
```

Figure 10: The Connective FUG -- Section 2

"buy,""bribe" and "sell" all can express the concept of "exchange", but "bribe" adds the desired argumentative orientation (i.e., the transaction was dishonest) to the clause. Figure 13 shows how the argumentative constraint given in the input can be satisfied by the choice of verb. The first fragment is taken from the grammar for verbs. It shows how the verb's argumentative feature from the lexicon, when there is one, is sent up to the clause using the feature `(justified yes)` in the AO description. The second fragment, taken from the clause grammar, indicates that the AO feature of a clause **must** eventually be justified.

When the simple clause C1 is unified with the grammar, the concept "exchange" is first mapped to the verb "buy." But the entry for "buy" contains no AO and the feature `justified` remains unbound. Thus, this first unification fails, the unifier backtracks and tries the verb "bribe." Since the lexical entry for "bribe" contains an AO, the feature `justified` is set to `yes`, and the argumentative constraint of the input is satisfied. The grammar eventually produces the sentence "A&C Builders bribed the Mayor with $10,000 to receive the license for the construction site."

Figure 14 now shows the same proposition, with the same argumentative constraint but embedded in a complex clause. This complex input represents a type of concessive move: the locutor concedes that A&C Builders "exchanged" $10,000 for a license and that this exchange can be an argument for dishonesty of

```
C1 =
((cat discourse-segment)
 (directive
  ((th ~(A&CB License  CSite Mayor $10,000 Exchange))
   (if ((force assert)))
   (ao ((scale dishonest)
        (conclusion ((process-type attributive)
                     (carrier ((concept A&CB)))
                     (attribute ((concept dishonest)))))))
   (pc ((cat clause)
        (process-type action)
        (concept Exchange)
        (benef/init ((lex "A&C Builders")
                     (concept A&CB)
                     (np-type proper)))
        (benef/react ((lex "Mayor")
                      (np-type common)
                      (concept Mayor)
                      (definite homophora)))
        (medium/init ((concept cash)
                      (quantity 10000)
                      (unit $)))
        (medium/react ((head ((lex license)
                              (concept license)
                              (np-type common)))
                       (qualifier ((prep === for)
                                   (concept CSite)
                                   (head === site)
                                   (classifier === construction)
                                   (describer ((lex new)
                                               (defining yes))))))))))))

((cat discourse-segment)
 (directive
  ((th ~(A&CB Exchange Honest))
   (if ((force assert)))
   (ao ((scale honest)
        (conclusion ((process-type attributive)
                     (carrier ((concept A&CB)))
                     (attribute ((concept honest)))))))
   (pc ((cat clause)
        (process-type attributive)
        (carrier ((head ((concept intention)))
                  (number plural)
                  (determiner ((possessive yes)
                               (np-type pronoun)
                               (pronoun-type personal)
                               (concept A&CB)
                               (number plural)
                               (person third)))))
        (attribute === honest)))))
```

Figure 11: Sample input with argumentative constraint

A&C, but states a stronger belief that A&C acted honestly in the directive move.

Lexicon =

```
(alt (((concept Move) ...)
      ...
      ;; Different verbs to express the Concept Exchange
      ;; For each verb, map the concept specific roles to more
      ;; general semantic roles accepted by the grammar.
      ;; Also preselect the governing prepositions if needed.

      ;; to buy: BI buys MR from BR for MI.
      ((concept Exchange)
       (lex "buy")
       (agent (^ benef/init))
       (medium (^ medium/react))
       (instrument (^ medium/init))
       (benef (^ benef/react))
       (instrument ((prep === for)))
       (benef ((prep === from))))

      ;; to bribe: BI bribes BR with MI in order to possess MR.
      ;; is also marked on its argumentative function
      ((concept Exchange)
       (lex "bribe")
       (AO ((conclusion
             ((process-type attributive)
              (carrier (^ ^ ^ ^ pc agent))
              (attribute dishonest)))
            (scale dishonest)))
       (agent (^ benef/init))
       (medium (^ benef/react))
       (instrument (^ medium/init))
       (purpose ((cat clause)
                 (concept Possess)
                 (agent (^ ^ agent))
                 (medium (^ ^ medium/react))))
       (instrument ((prep === with))))

      ;; to sell: BR sells MR to BI for MI.
      ((concept Exchange)
       (lex "sell")
       (agent (^ benef/react))
       (medium (^ medium/react))
       (instrument (^ medium/init))
       (benef (^ benef/init))
       (instrument ((prep === for)))
       (benef ((prep === to))))
      ...))
```

Figure 12: A fragment from the lexicon

The unification of C2 leads to the generation of "A&C Builders bought a license for the construction site from the Mayor for $10,000, but their intentions were honest." The first step of the unification will go through the `discourse-segment` category of Figure 7 where the pattern SCD will be chosen. Next, the unifier applies the constraints from functional status and argumentation. At the end of this first sweep through the connective category, all the constraints that can be derived from the input on the features have been verified or added to

In grammar for verb:

```
((cat verb)
 ...
 (alt (((ao (^ ^ ^ ao))
        (ao ((justified yes))))
       ((ao none)))))
```

The AO of the verb can justify the AO of the clause if the verb is argumentatively loaded.

In grammar for clauses:

```
((cat clause)
 ...
 (ao ((justified any))))
```

The AO of a clause must be justified by one of the linguistic devices realizing the clause.

Figure 13: Fragments from the grammars for verbs and clauses

C2 and the modified input will be as shown in Figure 15. The unifier then proceeds to the unification of the clauses. The argumentative constraint given in input to the subordinate clause is now satisfied by a constraint coming from the choice of the connective "but." When the lexicon is reached, the default verb "buy" is chosen, and the choice need not be reconsidered, since the input constraint is already satisfied.

This example demonstrates how complex interaction between lexical choice in the clause and connective selection can be implemented by the FUG without requiring the grammar writer to explicitly express the interaction. Similarly, if the deep planner had any constraints on lexical choice, they would be included in the PC feature of the discourse segment. The argumentative constraints implied by the lexical choice would be reflected at the `Discourse-segment` level by the argumentative part of the category `Discourse-segment` in the grammar (not presented in the figure). Therefore, if one of the features is "preselected" at any level, the constraint it implies are enforced at the highest possible level immediately.

4 Comparison with Other Formalisms

In this section we compare order of decision making in FUG with order of decision making in two of the parsing formalisms we have used, the ATN and the DCG. Because there are a number of similarities between the two, we focus on the ATN showing how order of decision making would occur for both the sample syntactic grammar and connective choice. We then point out where processing in the DCG diverges from the ATN, allowing an added degree of flexibility. Both the ATN and the DCG, however, favor a syntagmatic mode of grammar organization, while FUGs allow a paradigmatic organization. Generation is more concerned with choice based on the paradigmatic axis. Therefore, when order of decision making follows the syntactic structure of the utterance being produced, we run into problems both in the degree of flexibility and in the representation of constraints.

Finally, we turn to two formalisms that have been used for generation, the systemic formalism [Mann 83, Patten 88] and MUMBLE [McDonald 86].

```
C2 =
((cat discourse-segment)
 (subordinate
  ((directive
    ((th ~(A&CB License  CSite Mayor $10,000 Exchange))
     (if ((force assert)))
     (ao ((scale dishonest)
          (conclusion ((process-type attributive)
                       (carrier ((concept A&CB)))
                       (attribute ((concept dishonest)))))))
     (pc ((cat clause)
          (process-type action)
          (concept Exchange)
          (benef/init ((lex "A&C Builders")
                       (concept A&CB)
                       (np-type proper)))
          (benef/react ((lex "Mayor")
                        (np-type common)
                        (concept Mayor)
                        (definite homophora)))
          (medium/init ((concept cash)
                        (quantity 10000)
                        (unit $)))
          (medium/react ((head ((lex license)
                                (concept license)
                                (np-type common)))
                         (qualifier ((prep == for)
                                     (concept CSite)
                                     (head == site)
                                     (classifier == construction)
                                     (describer ((lex new)
                                                 (defining yes)))))))))))))
(directive
 ((cat discourse-segment)
  (directive
   ((th ~(A&CB Exchange Honest))
    (if ((force assert)))
    (ao ((scale honest)
         (conclusion ((process-type attributive)
                      (carrier ((concept A&CB)))
                      (attribute ((concept honest)))))))
    (pc ((cat clause)
         (process-type attributive)
         (carrier ((head ((concept intention)))
                   (number plural)
                   (determiner ((possessive yes)
                                (np-type pronoun)
                                (pronoun-type personal)
                                (concept A&CB)
                                (number plural)
                                (person third)))))
         (attribute == honest)))))))
```

Figure 14: Sample input for connective

Argumentative features added to C2:

```
((directive
   ((AO ((conclusion ((process-type attributive)
                      (carrier ((concept A&CB)))
                      (attribute ((concept dishonest)))))
         (orientation -)
         (scale dishonesty)))
    (Topos ((scale-left honesty)
            (sign-left +)
            (scale-right dishonesty)
            (sign-right -)))))
 (subordinate
  ((AO ((conclusion ((process-type attributive)
                     (carrier ((concept A&CB)))
                     (attribute ((concept dishonest)))))
        (orientation +)
        (scale dishonesty)))
   (Topos ((scale-left Bribe)
           (sign-left +)
           (scale-right dishonesty)
           (sign-right +))))))
```

Figure 15: Argumentative features added to C2

4.1 Using an ATN for Generation

Difficulties arise in using the ATN for language generation given the depth first traversal of the network and the need to synchronize in lock-step the influence of different constraints as syntactic structure is constructed. The traversal algorithm means that production of leftward constituents can not be influenced by decisions made in producing constituents towards the end of the sentence. Furthermore, it can be difficult to allow for complex interaction between constraints since different constraints must be coordinated as the system traverses the network. To show why this is the case, we first describe how generation is done in the ATN and then show through examples of a simple grammar and a grammar for connectives how decision making is constrained.

The ATN generator we use makes the following assumptions:

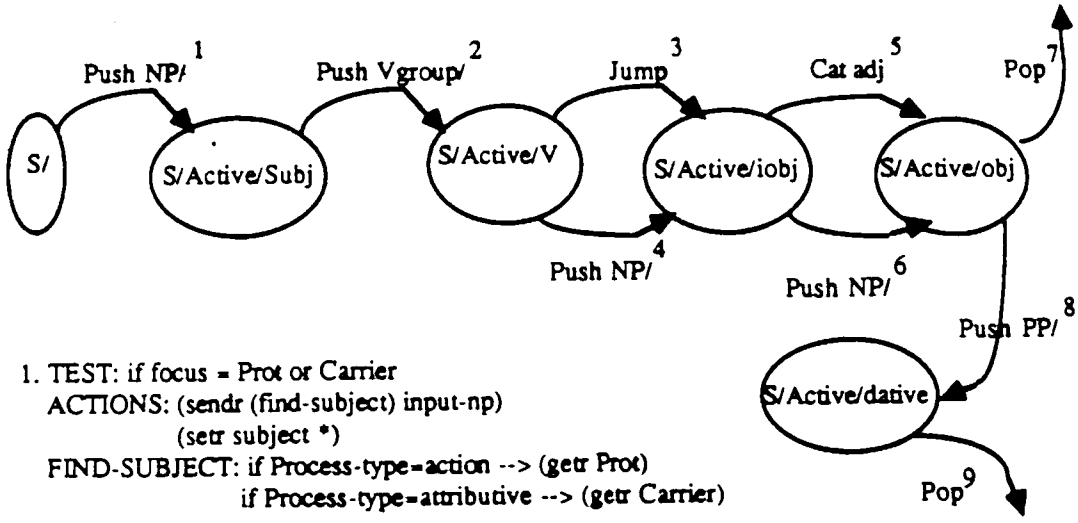
- Input to the ATN interpreter is a list of case roles, such as *prot*, *goal*, *predicate*, etc. Registers are initially filled with the values for these case roles and can be accessed when traversing the grammar.
- The generator works by traversing the net, producing a word whenever it encounters a cat arc. On traversal of a **cat** arc, the special register * is set to contain the word produced.
- The same grammar can be used for both interpretation and generation provided:
  1. Rather than building a tree structure from registers at pop arcs, the grammar strings registers together in list form to construct the sentence produced.
  2. The grammar writer provides arbitrary LISP functions associated with each category that access specified registers to determine the word or words to generate for the category at this point in the sentence. A cat arc can be traversed if the associated LISP function can select a word for the given category. For example, the grammar writer might provide functions *produce-adj*, *produce-det*, and *produce-noun* which would access specified input registers when the **cat det, adj**, and **noun** arcs are traversed to determine if a word in

those categories can be produced.

3. The grammar writer may choose to add actions to any arcs to manipulate input registers. For example, when attempting to construct the subject of a sentence from the input prot register, one common action would be to send the value of the prot register down to the np network. Similarly, one could send the value of the goal register down to the NP network when constructing the object of the sentence. This allows the NP network to access a single register when constructing a NP whose content varies depending on its context in the sentence.

The ATN interpreter for language generation makes decisions in two ways. Decisions are made about what to produce each time the system has a choice of arc to take next. Constraints on this type of choice can be represented as arbitrary LISP tests on the arc. Alternatively decisions can be made on traversal of a *cat* arc by its associated function. This function may decide whether a word of the specified category may be produced at all, and if so, what word will be produced.

4.1.1 Simple Syntactic Grammar



1. TEST: if focus = Prot or Carrier
   ACTIONS: (sendr (find-subject) input-np)
   (setr subject *)
   FIND-SUBJECT: if Process-type=action --> (getr Prot)
   if Process-type=attributive --> (getr Carrier)
2. ACTIONS: (sendr Process-type) (sendr Verb) (setr Verbgroup *)
4. TEST: if Verb-class=bitransitive and if no Dative-prep in Verb
   ACTIONS: (sendr (find-iobject) input-np) (setr iobject *)
   FIND-IOBJECT: if Process-type=action --> (getr Benef)
5. TEST: if Process-type=attributive ACTIONS: (setr object *)
6. ACTIONS: (sendr (find-object) input-np) (setr object *)
   FIND-OBJECT: if Process-type=action --> (getr Goal)
   if Process-type=attributive --> (getr Attribute)
7. RETURN: Subject Verbgroup Object if Transitive
   Subject Verbgroup Iobject Object if Bitransitive
8. TEST: if Dative-prep in Verb
   ACTIONS: (sendr (find-iobject) input-np) (sendr Dative-prep input-prep) (setr Dative *)
9. RETURN: Subject Verbgroup Object Dative

Figure 16: Sample network

To compare order of decision making in an ATN with order of decision making in FUG, consider a sample network shown in Figure 16 which is one way to translate the sample FUG of Figure 1. Syntactic structure of the constituents of the sentence is built by traversing subnetworks. Order of the constituents is also determined by order of arc traversal and by the **buildq** action on pop arcs.[11] Assignment of semantic roles to syntactic ones is done by complex actions on the arcs.

The important point to note in this grammar is that decisions are made in building the syntactic structure of the sentence, top-down and left-to-right. Mapping of semantic roles to syntactic roles, building of syntactic structure, and ordering of roles all occur simultaneously. For example, using `T2`, Figure 4 as input, traversal of the network would begin with arc 1 since **focus** is on the **prot**. At this point, assignment of the semantic role **prot** to the syntactic role **subject** would be made as part of the **sendr** action. In addition, a decision to produce an active sentence would have been made by the test. This arc would produce the NP for the subject, "John" and place it in the **subject** register.

This ATN grammar is but one way of translating the FUG. There are a variety of other possibilities. Since the ATN is turing machine equivalent, it would be possible to follow the FUG more exactly. One could use 3 stages in the ATN, where the first two stages resulted in the setting of registers used for features corresponding to FUG attributes and in the final stage only, would the sentence actually be produced. This grammar would only use **test** and **jump** arcs and would not correspond to the normal use of ATNs. Most of the work for generation would be done in the LISP functions used as actions or tests on the arcs. Clearly, this is not a desirable solution.

4.1.2 Characterization of Differences

In the ATN version of the FUG that we presented here, the ordering of constituents in the resulting sentence is conflated with the assignment of syntactic structure to constituents while in FUG these tasks were represented in two separate sections of the grammar. The necessity to mix different sorts of information as well as the depth-first traversal algorithm of the ATN results in these primary differences:

Left-to-right Traversal: In the ATN version, decisions about embedded constituents will get made before some top-level decisions. In the example, the subject "John" is fully determined before syntactic ordering decisions such as where the indirect object is placed. In FUG, all decisions that can be made at the top level are made before producing constituents. The ATN's order of decision making will cause problems for the case of connectives where a decision made early on in the sentence depends on a decision further to the right.

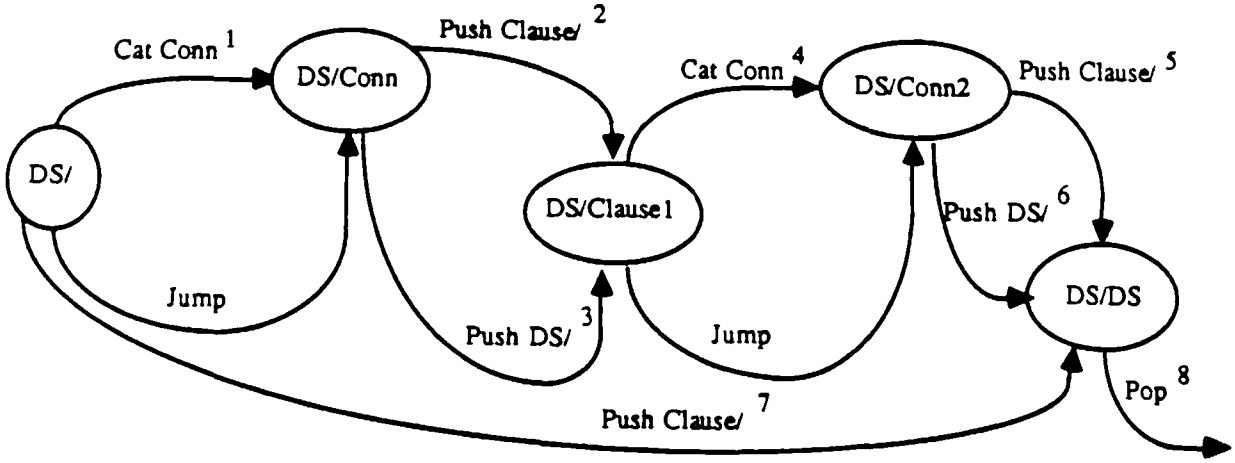
Synchronization of different types of constraints: An ATN must synchronize in lock-step the influence of different constraints as it proceeds through the construction of syntactic structure of the sentence. It can be difficult to coordinate these different constraints and it means that the grammar writer must know in advance exactly when these constraints will come into play in producing the sentence.

---

[11] Actually, a more sophisticated ATN interpreter might construct linear order of constituents in the sentence simply by tracking order of arc traversal thus allowing the user to omit the **buildq** statements.

4.1.3 Implementation of Connective Choice

Since the form of an ATN must follow the syntagmatic structure of the sentence, the network to perform connective choice is made of roughly four parallel paths corresponding to the orders ScD, cDS, DcS and cSD. The input to the ATN interpreter is a list of registers, containing the values of all features present in a `discourse-segment`. Note that the input does not have the structured aspect of an FD, as all features, at all levels, must be put in different registers.



1. (setr Conn *)
2. (sendr Directive input-clause) (setr Clause1 *) (setr Dir T)
3. (sendr Subordinate input-clause) (setr Clause1 *) (setr Sub T)
4. IF Conn empty, (setr Conn *) (setr Middle T)
5. IF Sub, (sendr Directive input-clause) (setr Clause2 *)
6. IF Dir, (sendr Subordinate input-clause) (setr Clause2 *)
7. (setr Simple T) (sendr Directive input-clause) (setr Clause1 *)
8. IF Simple,. return Clause1, ELSE IF Middle, return Clause1 Conn Clause2,
   ELSE return Conn Clause1 Clause2

Figure 17: Top level Network for Complex clause

Figure 17 shows the top level of an ATN that could be used for connective selection. For an ATN interpreter, the Utterance category is realized as a clause. Therefore, on the arcs **directive** and **subordinate** we push to the **clause** subnetwork, and there is no **utterance** subnetwork. All the actions performed by the category utterance in the FUG implementation must therefore be done on the actions of the arcs.

The most natural way to perform connective selection is to associate a procedure to each connective (*e.g., produce-but, produce-although*). The procedure will do all the testing required by the description of the conjunction. This means that the selection procedure must be implemented in Lisp. An alternative approach is

to write only one procedure implementing the whole selection procedure.

It is therefore the responsibility of these procedures to behave either as testers or as generators when one of the registers being tested turns out to be empty. If the procedures are to work as generators, the grammar writer must basically rewrite the unification algorithm in each procedure.

The difference between the two approaches is that in the FUG implementation, the part of the grammar handling connective selection does not need to be aware of the source of a constraint on a feature: it just tries to unify the feature with possible values. In contrast, the ATN grammar writer must explicitly describe the types of interaction that can occur: what register can affect the value of each feature, and test for all of them. This complexity is derived from our desire to leave order of decision making unconstrained. If, on the other hand, we accept a specification of priorities between the features involved in the selection, then the ATN implementation can be made much simpler. All the procedures can work as tests only, testing more 'primitive' features first, and assigning values to less 'primitive' values as a result. This is, however, the type of rigid interaction we want to avoid.

4.2 DCG

DCGs share with ATNs the characteristic of favoring a syntagmatic mode of grammar organization. Typically, a DCG encodes the structural properties of a language in context free rules having both a left and right hand side. These are augmented by extra conditions on the rules. For generation, these tests would make any context sensitive tests required. For example, a rule stating that OBJ --> ADJ might have the test that process-type is attributive since action process-types do not have adjectival objects. In addition, pragmatic information could be tested to determine certain syntactic choices. For example, focus might be tested as part of a rule that determines the active form be used.

Mapping of semantic roles to syntactic roles is achieved through the use of arguments to rules. For example, the DCG we use in one of our generation systems [Derr and McKeown 84] contains the rule shown in Figure 18. This rule builds syntactic structure for a sentence (nplist followed by vb_phrase) and maps the semantic roles provided in input (verb, prot, goal, beneficiary, focus) to syntactic roles such as subject and object by passing them to subconstituents of the sentence (e.g., focus is passed to nplist to be used as the subject of the sentence).

```
sentence(clause(Verb,Prot,Goal,Bene,Focus,Advs,Mods)) -->
                        {trimcore},
                        nplist(Focus,subj),
                        vb_phrase(Verb,Prot,Goal,Bene,Focus,Advs),
                        mods(Mods).
```

Figure 18: Sample DCG Rule

As in the ATN, then, the result is a conflation of different types of constraints into individual rules. Mapping of semantic roles to syntactic roles, building of syntactic structure, and ordering of syntactic roles based on pragmatic constraints are represented as synchronized decisions that occur in lock-step as the sentence is produced. Furthermore, as in the ATN, processing of rules is top-down, left-to-right meaning that a constituent

to the left in the sentence will be fully determined before other decisions get made.

Unlike the ATN, the DCG uses unification as the mechanism for processing rules and this results in two types of added flexibility. Through unification of arguments, some constraints can be expressed at a higher level and passed down to lower level constituents. More importantly, constraints are necessarily bidirectional precisely because of the use of unification.

4.3 MUMBLE

Order of decision making in MUMBLE [McDonald 86] is quite different from order of decision making in FUG. First, it is determined by the incoming message and not by the grammar. To produce a sentence the incoming message is traversed, replacing each plan unit of the message with a possibly partial syntactic tree structure, until a full tree is produced. The second main difference is McDonald's commitment to a linear algorithm. All decisions are indelible. As far as we can tell, this means that bidirectional constraints can not be accounted for.

To compare order of decision making more directly, consider how MUMBLE would generate our example sentence "John gives a blue book to Mary." MUMBLE expects as input a *realization specification* for the text, which is a plan represented in semantic and pragmatic terms. A realization specification for this sentence might be as shown in Figure 19[12].

```
(transfer-event
        (main-event #<transfer John Mary indefinite-book>)
        (particulars #<attribute book color blue>))
```

Figure 19: Realization Specification for "John gives a blue book to Mary"

MUMBLE's generation process consists of three subprocess: Attachment, Realization, and Phrase Structure Execution (PSE). While these levels are organized as separate modules, they are not strictly ordered in the overall process. Rather, they are interleaved processes that pass information and partially refined structures between themselves. Flow of control is in part dictated by the input plan and in part by programmed knowledge dictating when to invoke the next component. Attachment is responsible for assigning plan units to positions within the surface structure tree that MUMBLE builds. At any point in the process, the surface structure contains "attachment points" to which new structures can be added. Initially, the first plan unit is assigned to the only available attachment point, the node dominating the first sentence. Attachment is interleaved with PSE. As soon as a partial tree is constructed, PSE takes over and does a depth-first traversal of the tree. PSE invokes procedures indicated by labels of the tree to perform transformations or enforce syntactic constraints. Words undergo morphological analysis and are produced when the leaves of the tree are reached. PSE re-invokes Attachment if it arrives at a node that is an attachment point (e.g., any noun phrase would allow the attachment of qualifying clauses) to check if there are additional plan units that can appear at

[12]We hypothesize about the primitives of this example. Papers on MUMBLE do not specify the primitives for the specification language. We base our choice of primitives on an example from [McDonald and Pustejovsky 87]

this point. PSE invokes Realization when it encounters plan units in the surface structure.

Realization is responsible for all choices that have to be made during language generation. It selects an appropriate word or phrase to "realize" a plan unit. There are two types of realization classes: domain dependent classes, which are essentially dictionary entries that list possible word choices for plan elements, and linguistic classes, which identify transformation families for a particular syntactic constituent. For example, McDonald defines a linguistic realization class for transitive verbs that can be nominalized that identifies seven syntactic choices for realizing any verb of this class (e.g., active, passive, gerundive with subject, gerundive passive with subject, etc.). Thus, grammatical decisions are made by two components: by realization classes, which make syntactic choice, and by procedures, which enforce syntactic constraints and are invoked by tree labels encountered by PSE during traversal.

In generating from the example input (Figure 19), MUMBLE's first step would be to attach the first plan unit (**main-event**) to the node dominating the first sentence in the surface structure (shown in Figure 20). PSE would then be invoked to traverse the tree. Almost immediately Realization would be called to realize the main-event. Realization can be done incrementally; its first decision may be to select the verb and construct a new tree under the **S** node, that includes the verb and its syntactic choice and assigns the remaining plan arguments to syntactic roles (see Figure 21). Realization would also denote which of the new nodes were active attachment points (in this case, there would be three, the subject head, the object head, the recipient head and the next clause). PSE continues and when it encounters the attachment point for subject would re-invoke Attachment, only to discover that there were no new plan units that could be incorporated as modifiers or qualifiers of the subject. Realization would be invoked to realize the subject as "John" and this word would be output. Similarly, the verb would next be conjugated and output. On encountering the object, Attachment would be invoked again and this time the plan unit specifying attributes of book would be folded into the surface structure. This new subtree would be traversed and the phrase "the blue book" produced. The recipient label might be responsible for adding the "to" preposition and finally, the phrase "to Mary" would be produced completing the sentence.

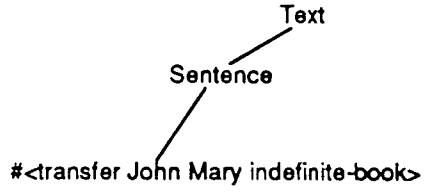


Figure 20: Surface Tree after 1st Attachment

As can be seen in this example, order of decision making in MUMBLE has the effect of separating out the representation of different kinds of constraints. Constraints dictating how syntactic structure is built are

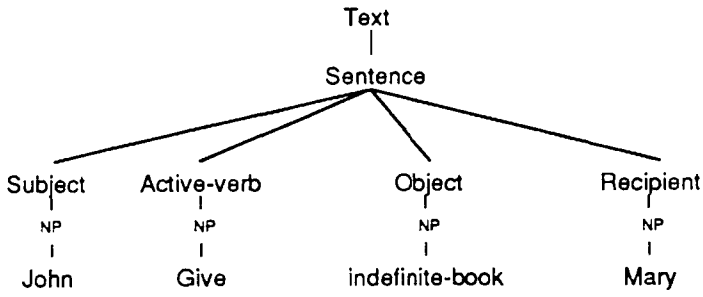


Figure 21: Surface Tree after 1st Realization

invoked during PSE and are represented as procedures invoked by node labels. Attachment also plays a role in building syntactic structure and represents constraints on how smaller trees can combine to form larger ones. Constraints on ordering of constituents are represented in linguistic realization classes. They are represented somewhat more declaratively in the class entries, but they still invoke procedures to carry out construction of the tree representing ordering. The mapping from semantic input to syntactic roles is done in the domain dependent realization classes. Thus, like FUG, MUMBLE supports a separate representation of different kinds of constraints. Unlike FUG, however, each type of constraint is represented differently (e.g., some in procedures, some in realization classes). Furthermore, the separation of constraints is fixed by the flow of control. In FUG however, one could decide to add additional classes of constraints in yet another layer of grammar that is unified with results from earlier layers.

MUMBLE's indelibility constraint is the main significant difference. Decisions get made in a fixed order and this precludes the possibility for constraints to be bidirectional. As we understand, this means that MUMBLE would not be able to handle the interaction between selection of connective and the generation of the embedded clause. We can only speculate about how connectives might be produced, since there is little published material about the generation of complex sentences (but see [McDonald 86]). We suspect that there would be separate plan units for the propositional content of the two clauses to be conjoined. Since MUMBLE proceeds left to right through the message to generate the first of the clauses (we suspect that the ordering of the clauses would be governed by the message), Attachment would add the second clause to the tree only after the first clause is generated.[13] As a result, while decisions made in the production of the clause can influence the connective, the selection of a connective can not influence the clause.

[13]The statement "..attachment is interleaved with Phrase Structure Execution so that most earlier units will have been realized and their text spoken before the last one [plan unit] is positioned." [McDonald 86] leads us to believe that this is the case. However, a second possible interpretation arises from the statement "With the realization of the first unit, the attachment possibilities for the second can be considered." If "realization" here refers to the Realization phase and not to the full traversal of the surface structure, it is possible that all attachments are considered immediately after a tree structure replaces a plan unit, but before PSE continues. This seems unlikely as this would mean attachment of all plan units would be possible before PSE continued and anything was produced. Since this is what McDonald wants to avoid it seems unlikely.

4.4 Systemic Formalisms

There have been two recent major implementations of systemic grammar for generation, NIGEL [Mann 83] and Patten's system [Patten 88], each of which uses a different control strategy for processing. The grammar is represented as an interconnected set of systems, where the lowest levels represent the most delicate decisions (e.g., lexical choice) and the highest level systems the least delicate (e.g., clause type).

NIGEL is part of the PENMAN system, which in addition to the surface grammar, includes a lexicon, an input specification language (called SPL), and the most general part of a knowledge base (called the "upper model"). NIGEL as a surface grammar expects an extremely rich input, represented as a set of features. Within the PENMAN system, the SPL interpreter allows the user to enter specifications in a much simpler way and to only partially specify the features that must be expressed. Figure 22 shows a possible SPL specification that might be used as input to generate our example "John gives a blue book to Mary." All the features that are not specified in the SPL input are given a default value. SPL and defaulting are described in [Penman 88].

```
((GIVE1 / GIVE                ;; GIVE1 is an instance of GIVE in the domain model
   :actor JOHN1
   :destination MARY1
   :object BOOK1
   :tense PRESENT
   :speechact ASSERTION)
 (JOHN1 / PERSON
   :name John)
 (MARY1 / PERSON
   :name Mary)
 (BOOK1 / BOOK
   :determiner A
   :relations  ((C1 / COLORING
                 :domain BOOK1
                 :range BLUE)))
 (BLUE / COLOR))
```

Figure 22: Input to NIGEL for "John gives a blue book to Mary"

Note that in the SPL input, the values of the slots refer to entities in a knowledge base (the domain model). When needed, NIGEL will query this knowledge base to make a decision through a mechanism using special functions called *inquiry* and *choosers*.

In NIGEL a sentence is produced by "traversing" the grammar systems, starting with the least delicate. In each system, a choice is made by invoking the *chooser* function associated with the system, which in turn invokes one or more primitive *inquiry operators*. These functions will query the domain model of the system for information needed to make the choice. Depending on the results of the choice and the selected system, different systems will be invoked next in the overall process of producing the sentence. Features can be preselected, however. In preselection, a "leaf"[14] feature is input which means that the path from higher level

[14]All quoted terms in descriptions of NIGEL are our own terms and may not correspond to the terminology used by the NIGEL group.

systems to that lower level feature can be avoided in processing.

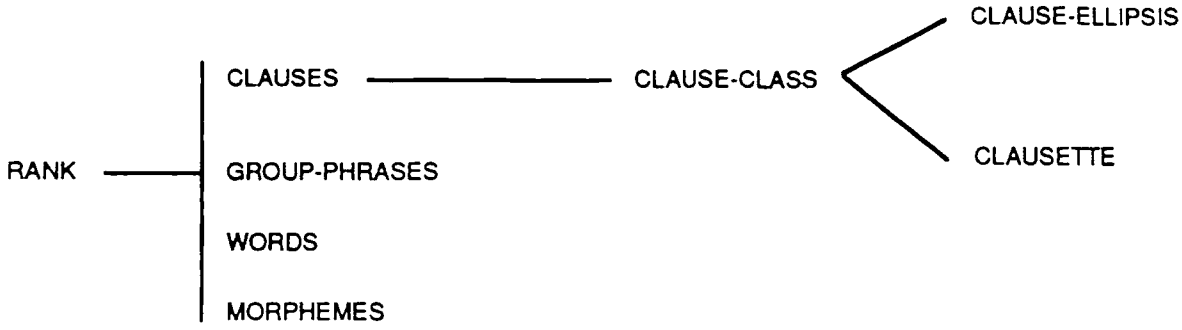


Figure 23: Top level systems in NIGEL

For example, Figure 23 shows the top level systems of the NIGEL grammar. When going through the root system, the grammar will decide the rank of the expression to generate. To make this decision, NIGEL will determine whether the input includes a speech-act. As our example does contain a speech-act, the next system, CLAUSECLASS is entered. Next, NIGEL needs to determine whether the speech-act has a propositional parameter (that it, if it is applied to a proposition). The answer to this query will determine whether the generated clause will be a full clause or an exclamation or a greeting (which correspond to speech-acts without propositional content). Since our input includes a propositional content, the next system, CLAUSEELLIPSIS, is then selected. NIGEL needs to decide whether the clause is an answer to a question, in which case it can use ellipsis, or not, in which case the clause must be fully expanded. Flow of control continues in the same fashion through all the systems of the grammar, from the most general to the most specific (in what systemists call order of "delicacy").

The selection of connectives in NIGEL follows Halliday's description of clause complexes. Note that the decision to produce a single clause or a clause complex (that is, several clauses, connected in some way) is one of the first decisions made in NIGEL. Figure 24 shows the top level systems controlling connective choice.

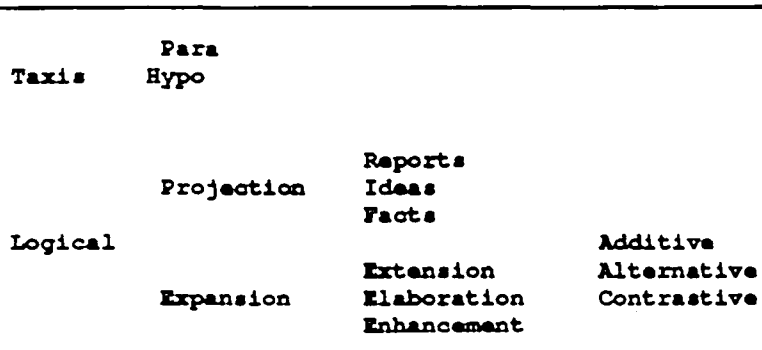


Figure 24: Top level systems describing clause connections

In order to arrange two clauses to produce a single clause complex, NIGEL determines the type of connections to express in the complex along two dimensions: taxis and logical. The decisions concerning taxis and logical can be made in any order, and are mostly independent of each other.

Taxis determines the syntactic status of each clause within the complex. In a *paratactic* relation, the two clauses have the same syntactic status (this roughly corresponds to the traditional notion of co-ordination); in an *hypotactic* relation, one clause is presented as a modifier of the other, and thus has a lower status (this corresponds to subordination).

The logical system attempts to describe the types of semantic relations between clauses that can be expressed in language. Once the logical and taxis dimensions are determined, NIGEL can choose a connective compatible with the decisions taken. For each combination of features on the taxis and logical systems there correspond one connective. For example, "but" corresponds to the features `paratactic` and `expansion:extension:contrastive`.

The default order of decision making for choosing a connective is therefore: (1) determine the type of connection between the clauses, (2) choose the connective, (3) realize each clause. Note that when the type of connection is determined, all relevant features in each clause get pre-selected. For example, if a hypotactic relation is chosen, the feature `dependent` will be selected in the `dependence` system - thus forcing the subordinated clause to be realized in a certain manner. Therefore, the flow of control in NIGEL is usually top-down, with the choice of the connective constraining the realization of each clause, both to the right and to the left of the connective.

In summary, as implemented in NIGEL, decision making is primarily top-down and is not inherently bidirectional. A systemic grammar gives priority to the functional status of language. Since systems are organized around the paradigmatic axis, order of decision making is very different from either of the parsing formalisms discussed so far. For example, top-level decisions will send constraints down to lower level decisions and order of decision making is not governed by left-to-right construction of syntactic constituents. We note that NIGEL is not a finished product, but is continually under development, and order of decision making as currently implemented is not a theoretical claim of its developers.

The mechanism to enforce the bidirectional influence we have mentioned in FUG, where a choice in the clause constrains the choice of the connective, is apparently not implemented in NIGEL. However, Patten's implementation of systemic grammar allows for successive back and forth sweeps through the grammar to deduce all possible choices given a set of preselected features. This control strategy seems to capture the bidirectional constraints for which we have argued in the FUG. Lower level decisions can influence higher level decisions and vice versa. In other respects, decision making is similar to NIGEL as it is guided by functional aspects and not syntactic structure.

5 Conclusions

The strong points of the FUG formalism we have identified for connective selection are the partial specification of the input and flexible order of decision making, both internal and external, that a FUG naturally implements. The FUG formalism also allows organization of the grammar along the different types of constraints involved. The localization of constraints of the same type in separate regions permits the grammar writer to identify the effect of constraints in an efficient and readable manner. This organization is not enforced by the formalism,

but can be used as a guideline to lay out FUGs in a readable way. A side benefit from the type of organization we advocate is that it is easy to detect and remove duplication of constraints across similar cases.

5.1 Problems with FUG: Representation and Use of complex constraints

FUG does have problems in representing certain types of constraints. The types of constraints we want to express when implementing the connective selection procedure are: equality of one feature with a constant (e.g., P must have a directive status), equality of two features (e.g., P and Q must have the same utterer), limiting the possible values of a feature (e.g., the thematization of P can be propositional, illocutionary or reinterpretation), and the negation of the previous types. We also need to express set relations: test the intersection of two sets (e.g, Th(P) and Th(Q) are not disjoint), membership (e.g., (P_i as in σ) is a member of AO(P)). Other types of constraints can occur (e.g., the right-hand side of the topoi are of opposite signs).

The FUG formalism directly encodes constraints of the first types: equality with a constant is expressed as *(attribute constant)*, equality between two features is expressed as *(attribute1 <path to attribute2>)*. To limit the possible values of a feature, an alternation can be used: *(attribute (alt (val1 ... valn)))*. Negation is not part of the basic unification formalism, but has been added to many unifiers ([Shieber 86, Karttunen 84]). It can also be simulated using the special value *none*. More complex constraints can be expressed as composition of the previous types. For example, to express the constraint on the signs of the topoi, the following expression can be used:

```
(alt (((dl-p ((sign-right +)))
       (dl-q ((sign-right -))))
      ((dl-p ((sign-right -)))
       (dl-q ((sign-right +))))))
```

Constraints on sets are more problematic. It is difficult to express anything about sets using the standard FUG formalism. FDs allow values to be either atomic or FDs. It is possible to write grammars to deal with sets or lists; we actually have written such a grammar to compute the *append* of two lists, test for membership, or compute the intersection of two lists. Such grammars are, however, terribly inefficient and not very readable. It is more productive to acknowledge the limitation of the formalism, and to add facilities to express more complex constraints [Elhadad 89].

We have addressed these problems by introducing a special attribute into the formalism, **test**, that has a special unification behavior. This adds to the existing special attributes **pattern, cset,** and **alt**. The value of a **test** attribute can be an arbitrary predicate represented by a Lisp expression, containing, if necessary, references (paths) to other features in the FD being unified. The unification behavior of a **test** feature is to ensure that the predicate is true in the FD resulting from the unification. In practice, the Lisp expression is evaluated at the end of the unification and when it fails, the unifier backtracks.[15] At a more abstract level, a **test** feature enforces a complex constraint on an FD. For example, the following grammar fragment insures that the themes of P and Q are non disjoint:

```
((cat connective)
 ...
 (Test (FD-intersection @(^ ^ P Th) @(^ ^ Q Th))))[16]
```

[15] A more efficient strategy to choose the moment when the constraint must be evaluated is being implemented.

[16] The symbol '@' indicates that the following expression is a path referring to a value in the FD.

Partial specifications on values: The **test** feature allows us to achieve an acceptable level of performance for testing complex constraints. Unfortunately, it also has a major drawback: the regular unification algorithm does not make a distinction between testing a constraint and adding a constraint. When testing against a non specified value, the unifier can just add the constraint. The **test** feature, in contrast, does not indicate how the constraint it enforces should be added. In other words, **test** is not bidirectional.

5.2 Summary

We argue in this paper that the FUG formalism is a natural choice for the task of text generation. As the scope of text generation extends to include more decisions, of different nature, using different information, the problem of order of decision making becomes more acute. We have distinguished two aspects of this problem: internal - how decisions interact within the surface realization component - and external - how decisions in the surface realization component interact with its environment. Because language imposes arbitrarily complex constraints on any decision, these interactions can be quite complex. They cannot be handled by a module that is not be aware of the linguistic intricacies. Since constraints cannot always be strictly ordered, it is natural to let the linguistic component deal with interactions in the most flexible way.

We have illustrated how FUGs allow for that flexibility by examining the task of connective selection. FUGs allow for flexible internal order of decision making, and provide tools for organizing the grammar without duplicating constraints, and allow a clear grouping of similar constraints to increase readability. They allow for flexible external order of decision because unification is bidirectional, and the constraints expressed in the grammar can both generate and test values.

References

[Anscombre 73] Anscombre, J.C.
Meme le roi de France est sage. Un essai de description semantique.
Communications (20):40-82, 1973.

[Anscombre & Ducrot 83]
Anscombre, J.C. & Ducrot, O.
Philosophie et langage: L'argumentation dans la langue.
Pierre Mardaga, Bruxelles, 1983.

[Appelt 85] Appelt, D.E.
Planning English Sentences.
Cambridge University Press, Cambridge, England, 1985.

[Brachman 79] Brachman, R.
On the epsitemological status of semantic networks.
Associative networks: representation and use of knowledge by computers.
Academic Press, NY, 1979.

[Danlos 87] Danlos, L.
The Linguistic Basis of Text Generation.
Cambridge University Press, Cambridge, England, 1987.

[Danlos 88] Danlos, L.
Interaction of Decisions in Text Generation: Some Pronominalization Issues.
1988.

[Danlos and Namer 88]
Danlos, L. and F. Namer.
Morphology and Cross Dependencies in the Synthesis of Personal Pronouns in Romance Languages.
In *Proceedings of COLING-88.* Budapest, 1988.

[Derr and McKeown 84]
Derr, M. and K. McKeown.
Using focus to generate complex and simple sentences.
In *Coling84.* COLING, Stanford, California, July, 1984.

[Ducrot 83] Ducrot, O.
Le sens commun: Le dire et le dit.
Les editions de Minuit, Paris, 1983.

[Elhadad 88] Elhadad, M.
The FUF Functional Unifier: User's manual.
Technical Report, Columbia University, June, 1988.

[Elhadad 89] Elhadad, M.
Extended Functional Unification ProGrammars.
Technical Report, Columbia University, New York, NY, 1989.

[Elhadad & McKeown 88]
Elhadad, M. and McKeown, K.R.
What do you need to produce a 'but'.
Technical Report CUCS-334-88, Columbia University, January, 1988.

[Elhadad & McKeown 89]
Elhadad, M. and McKeown, K.R.
A Procedure for the Selection of Connectives in Text Generation.
Technical Report, Columbia University, New York, NY, 1989.

[Fillmore, Kay & O'Connor 87]
Fillmore, C.J., Paul Kay and M.C. O'Connor.
Regularity and idiomacity in grammatical constructions: The case of let alone.
Technical Report 48, Berkeley: UCB Cognitive Science Program, 1987.

[Finin 84] Finin T.
Documentation for the Bottom-Up Parser (BUP).
1984.

[Fraser 71] Fraser, Bruce.
An analysis of *even* in English.
Studies in linguistics semantics.
Holt, Rinehart and Winston, New York, 1971, pages 151-178.

[Goldman 75] Goldman, N.M.
Conceptual generation.
Conceptual Information Processing.
North Holland, Amsterdam, 1975.

[Halliday 85] Halliday, M.A.K.
An Introduction to Functional Grammar.
Edward Arnold, London, 1985.

[Horn 69] Horn, L.
A Presuppositional Analysis of *Only* and *Even.*
In *Papers from the Fifth Regional Meeting*, pages 98-107. Chicago Linguistics Society, 1969.

[Hovy 86] Hovy, E.H.
Integrating text planning and production in generation.
In *Proceedings of the 9th IJCAI*, pages 848-851. IJCAI, 1986.

[Iordanskaja et al 88]
Iordanskaja L., Kittredge R. and Polguere A.
Lexical Selection and Paraphrase in a Meaning-text Generation Model.
Presented at the Workshop on Natural Language Generation.
Forthcoming, 1988.

[Jacobs 85] Jacobs, P. S.
A knowledge-based approach to language production.
PhD thesis, Univ. of California, Berkeley, 1985.

[Karttunen 84] Karttunen, L.
Features and Values.
In *Coling84*, pages 28-33. COLING, Stanford, California, July, 1984.

[Kay 79] Kay, M.
Functional Grammar.
In *Proceedings of the 5th meeting of the Berkeley Linguistics Society.* Berkeley Linguistics Society, 1979.

[Kay 87] Kay, Paul.
Even.
July, 1987.
University of California, Berkeley.

[Kukich 83] Kukich, K.
Design of a knowldege based text generator.
In *Proceedings of the 21st ACL Conference*. ACL, 1983.

[Mann 83] Mann, W.C.
An overview of the Nigel Text Generation Grammar.
Technical Report ISI/RR-83-113, USC/ISI, April, 1983.

[Mann & Matthiessen 83]
Mann, W.C. and Matthiessen, C.
Nigel: a Systemic Grammar for Text Generation.
Technical Report ISI/RR-83-105, USC/ISI, 1983.

[Matthiessen 81] Matthiessen, C.M.I.M.
A grammar and a lexicon for a text production system.
In *Proceedings of the 19th Conference of the ACL*, pages 49-53. Association for Computational Linguistics, 1981.

[McDonald 86] McDonald, D.D. and Pustejovsky, J.D.
Description-directed natural language generation.
In *Proceedings of the 9th IJCAI*, pages 799-805. IJCAI, 1986.

[McDonald 88] McDonald, D.D.
On the place of words in the generation process.
Presented at the Workshop on Natural Language Generation.
Forthcoming, 1988.

[McDonald and Pustejovsky 87]
McDonald, D.D. and Pustejovsky, J.D.
TAGs as a Grammatical Formalism for Generation.
In *Proceedings of the European ACL*, pages 94-103. ACL, 1987.

[McKeown 85] McKeown, K.R.
Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.
Cambridge University Press, Cambridge, England, 1985.

[McKeown & Paris 87]
McKeown, K.R. and Paris, C.L.
Functional Unification Grammar Revisited.
In *Proceedings of the ACL conference*, pages 97-103. ACL, July, 1987.

[Nolke 83] Nolke Henning.
Etudes Romanes de l'Universite de Copenhague: Les adverbes paradigmatisants: Fonction et analyse.
Universite de Copenhague, 1983.
Revue Romane, special issue 23.

[Paris 87] Paris, C.L.
The Use of Explicit User models in Text Generation: Tailoring to a User's level of expertise.
PhD thesis, Columbia University, 1987.

[Patten 88] Patten, T.
A systemic bridge.
Presented at the Workshop on Natural Language Generation.
Forthcoming, 1988.

[Penman 88] The Penmane Natural Language Generation Group.
The Penman User Guide
USC-ISI, 4676 Admiralty Way, Marina Del Rey, CA 90292, 1988.
Draft.

[Pereira & Warren 80]
Pereira, F.C.N. and Warren D.H.D.
Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks.
Artificial Intelligence 13:231-278, 1980.

[Quirk *et al* 72] Quirk, R. *et al.*
A Grammar of Contemporary English.
Longman, 1972.

[Raccah 87] Raccah, P.Y.
Modelling argumentation and modelling with argumentation.
Argumentation , 1987.

[Roulet *et al* 85] Roulet, E. *et al.*
L'articulation du discours en francais contemporain.
Berne, Lang, 1985.

[Rubinoff 88] Rubinoff, R.
A Cooperative Model of Strategy and Tactics in Generation.
Presented at the Workshop on Natural Language Generation.
Forthcoming, 1988.

[Sadock 81] Sadock, J.M.
Almost.
Radical Pragmatics.
Academic Press, New York, 1981, pages 257-271.

[Shieber 86] Shieber, S.
CSLI Lecture Notes. Volume 4: *An introduction to Unification-Based Approaches to Grammar.*
University of Chicago Press, Chicago, Il, 1986.

[Sinclair & Coulthard 75]
Sinclair and Coulthard.
Towards an Analysis of Discourse.
Oxford University Press, Oxford, England, 1975.

[Winograd 83] Winograd, T.
Language as a Cognitive Process.
Addison-Wesley, Reading, Ma., 1983.

[Woods 70] Woods, W.A.
Transition network grammars for natural language analysis.
Communications of the ACM 13:591-606, 1970.