CUCS-407-88

Comparison of Surface Language Generators: A Case Study in Choice of Connectives

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 this paper, we identify the characteristics of FUG that we find useful for generation and contrast these with characteristics of the parsing formalisms and with other formalisms that are typically used for generation.

Since we have found FUG a very natural formalism for the kinds of problems we have encountered, our approach is to select a particular generation task and show the advantages FUG provides. We use the task of selecting a connective (e.g., but, however, nonetheless, since, because, etc.) to conjoin two input propositions since, as a subset of the lexical choice problem, it contains a number of challenges peculiar to generation. We contrast an solution using a parsing formalism with the FUG solution, showing the lack of facilities for representing generation tasks.

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. 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 reversable 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.

The general criteria we have used in evaluating language generation systems include:

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 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 interactions between decisions 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. Whether processing is 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 reversable grammar means that syntactic knowledge need not be duplicated for the two tasks. In actuality, 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.

In this paper, we focus on one of these criteria, order of decision making and its impact on grammatical representation. Our claim is that order of decision making in FUG through unification allows for flexibility in representation of difficult generation tasks, such as lexical choice. In contrast, control strategies developed for parsing formalisms lack this flexibilty. In interpreting language, control of processing is driven in part by the input sentence, or word order. 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]).

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 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 representation of constraints 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 a comparisons with other formalisms.

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 that 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 which are to be unified are specified by the CSET attribute 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 in [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.

---

[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

```
;; Voice choice --> operative, middle.
;; Operative =~ active
;; 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
   ((voice operative)
    (verb ((voice-class non-middle)
           (voice active)))
    (alt
      (((process-type actions)
        (subject (^ prot))
        (object (^ goal))
        (iobject (^ benef)))
       ((process-type attributive)
        (subject (^ carrier))
        (object  (^ attribute))
        (iobject none))))))
```

Figure 2: Sample FUG -- Section 2

The grammar for the clause category is divided into three sections. In the first section (Figure 1), syntactic

```
;; 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

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

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. In the second section (Figure 2), the voice-class of the verb is identified and in this alternation, the semantic cases are mapped into the syntactic roles, subject, object, and indirect object. Only the mapping for active is shown. Finally, in the 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. 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 possible semantic clause type (in this grammar, either action or attributive process types), the first step is 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 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.[2] Later on, this choice will be confirmed or rejected by the focus constraint. The result from this unification is shown in Figure 6 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 details of the syntax for the active voice are filled into the grammar, specifying the order of the object and indirect object. The resulting FD is shown in Figure 7 and is linearized as "John gives the blue book to Mary.", following unification of its constituents, **prot, goal**, and **benef**.

3 Choice of Connective: an Example

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. One of the main questions centers about *when* a word is selected in the process of decision making. Is it part of the problem of deciding what to say or how to phrase content? How does it require interaction between the two phases? Although there have been a variety of different implementations and proposals for when lexical choice gets made and how it interacts with other decisions, there does appear to be consensus on at least two points: 1. some sort of interaction will ultimately be required and 2. some flexibility in when lexical choice gets made is likely to be necessary.

Given these questions about how lexical choice fits into the overall process of decision making in generation, one natural question is whether a grammatical formalism can provide the flexibility required in making these

---

[2]In the full grammar, there is an additional voice, the middle voice. It is not shown here as it does not play a role in the example.

[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.

T2 after unification with section 1:

```
((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"))))
```

Figure 5: After unification with Section 1

T2 after unification with section 2:

```
((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)))
```

Figure 6: After unification with Section 2

sorts of choices. In our work on choice of connectives, we have found it desirable to allow selection of a connective at any point along the path from choice of content to final generation of the utterance. In the remainder of this paper, we show how order of decision making in FUG allows flexibility on exactly where in the path of generation the choice gets made. In addition, we show how it allows for the choice of connective to feed back constraints on the content of the next utterance. Before turning to a discussion of constraints on

T2 after unification with (cat clause:)

```
((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 7: After unification of the clause level

connective choice and its representation in FUG, we briefly survey previous work in lexical choice identifying where in the order of decision making the task of lexical choice was positioned.

3.1 Previous 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, as, for example, the verb of the sentence is selected 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 & Mathiessen 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) however, and in that 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.[4] 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]) are similar in that they select whole phrases fairly early on 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 that Kukich works in, the use of 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 [mathiessen 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 Defining Constraints on the Selection of Connectives

Before describing possible ways to implement choice of connectives, we first describe the information involved in the decision to use a particular connective and present an abstract model that can be used as the basis of a selection procedure.

---

[4]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.

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. Connectives are therefore at the junction between deep and surface generation and their study can provide insight into both aspects of generation, and more importantly, into the type of interaction that must exist between them. In order to limit the complexity of the problem of connective selection, we limit our attention to a closed set of connectives and to the relations they can express.

To illustrate the information needed to select a connective, we use examples of the connective ''but'' taken from conversations. Our starting assumption is that connectives in conversation[5] indicate how to interpret and relate a turn to previous or implicit ones, rather than indicating a logical (truth-conditional) connection between the content of these units.

The role of certain connectives - often called *clue words* - as discourse structure markers has been acknowledged in previous work (see [Reichman 85, Cohen 87, Grosz & Sidner 86, Hirschberg & Litman 87]). The idea underlying this type of work is that connectives delimit discourse segments and indicate the position of the connected segments in a tree-like structure. We go further in this direction: connectives not only mark segment boundaries in discourse, they also identify and determine important semantico-pragmatic features of the units they join, that indirectly constrain their surface linguistic realization. For example, it is not sufficient to notice that 'but' marks a discourse segment boundary; we also need to determine what features of the connected units are affected by the presence of a 'but', and what role the complex sentence containing 'but' can play in a larger discouse.

In this section we present a set of such features, which we call an *interpretative format*, that can be used to choose a connective and generate an appropriate surface form. The set of features are drawn from the semantic theory of Ducrot [Ducrot 83, List 84] and the pragmatic theory of Roulet [Roulet *et al* 85]. We focus here only on a subset of interpretative formats, the set of argumentative features.

3.2.1 Interpretative Formats

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

The classic definition of 'but' indicates that the complex *'p but q'* expresses an opposition between *p* and *q* as illustrated in (1). A formalization of this definition in logical terms would state the equivalence:

$$p\ but\ q \equiv (p \wedge q) \wedge opposition(p,q)$$

But consider now the following example:

(2): **A:** It's beautiful
B: but it's expensive.

Whatever representation is chosen to specify the semantics of the predicate *opposition*, it seems unlikely one would maintain that the logical representation of A and B in (2) can be in opposition. For it is well accepted in our society that beauty deserves a high price, and we can reasonably accept the statement:

---

[5]We believe that this is a general characteristic of connectives in any type of discourse, but it is easier to detect in conversations.

$beautiful(x) \rightarrow expensive(x)$

This implication would tend to show that the logical representation of A and B are more in 'agreement' than in 'opposition'.

Similarly, [Quirk *et al* 72] propose that *'p but q'* expresses the surprise of a locutor that *q* is true in view of *p*. Here again, aside from the fuzziness of the notion 'surprise', it is difficult to believe that B can be surprised by the expensiveness of an object given its beauty.

Although we do want to 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'. The description of 'but' given in [Ducrot *et al* 80] 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. An interpretation of *p but q* requires the instantiation of what Ducrot calls the **argumentative variables** *c1* and *c2*.

In our example, such conclusions could be:

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

The opposition between A and B is **indirect** and requires the identification of **implicit conclusions.** 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' in uses similar to (2) as: 'but' indicates an opposition between the AO of the units it connects.

It remains to define 'opposition' in this context. If we consider the conclusions aimed at by utterances as formula of a first order language[7], we can define the logical predicate *oppose* between single formula first as simply:

$oppose(c1,c2) \equiv (c1 \rightarrow \neg c2) \equiv \neg(c1 \wedge c2)$

and as a predicate between sets of formula:

$oppose(AO1,AO2) \equiv (AO1 \cup AO2)$ *is inconsistent*

But we also must define the relation between a proposition P and its conclusions (AO). One approach would be to state that a proposition is an argument for a conclusion if an implication can be established linking the proposition to the conclusion. There are several problems with this approach. First, complex sentences introduce contradictions using this reasoning mechanism. For in a sentence *'P but Q'*, *P* is an argument for a conclusion *c1*, while *Q* is an argument against it. Nonetheless, the total sentence serves to support the

---

[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','almost', 'only' 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

conclusion drawn from *Q*, and no contradiction is left unresolved. Another problem with using implication is that the description of words like 'even' or 'only' requires defining the relation 'a1 is a stronger argument than a2 for the conclusion c'. Thus we need some way to rank arguments so that comparisons can be made.[8] To provide this ranking, [Anscombre & Ducrot 83] introduced the notion of **argumentative scale**. 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 & 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 morelless P, the morelless Q"*, P and Q being arbitrary formula along a scale, or as just constraints on the ordering of the formula *P* and *Q* and their negations. Using these tools, *Q* can be determined to provide a better argument than *P* if it falls higher on the scale.

In example (2), the topoi supporting the interpretation we have proposed would be:

(T1) The more an object is beautiful, the more a person wants to buy it.
(T2) The more an object is expensive, the less a person wants to buy it.

A connective thus expresses a relation between a set of semantico-pragmatic features of the units it conjoins and we represent these features in a structure we can an *interpretative format*. The preceding section indicated the need for the three features shown below. While we illustrated the need for these features through examples using the connective ''but'', they are also needed for other connectives.

- The *argumentative orientation* of an utterance is the set of conclusions that it can support. Since a proposition can support a conclusion if they both appear in an argumentative scale, we can represent the AO as the set of argumentative scales activated by the sentence and the propositions located on these scales: P_i **as in** σ where P_i is a proposition derived from the propositional content of the utterance and σ is an *argumentative scale*. When an utterance is not used as an argument, this feature is just empty, that is, the set of conclusions is not constrained.
- The *discourse law* is a list of *topoi* or argumentative scales, explaining the derivation of the argumentative orientation in the given situation. We represent topoi as pairs of scales with a sign indicating their polarity: (+/- σ1, +/- σ2).
- The *propositional content* of a turn is the notion used in the theory of speech acts ([Searle & Vanderveken 85]).

Other features that we have found play a role in connective selection are shown below. Of these, we will make further use of *functional status* in this paper, but we will not discuss the remaining features. We simply list them here for the sake of completeness. Functional status 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 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

[8]Note that the problem of contradictions could be dealt with by partitioning sets of propositions into belief spaces. But defining the relation 'better argument' without the notion of argumentative scale would require a definition of 'proximity' between logical propositions. We do not want to build this notion of distance into the theory, and prefer to describe the ordering of propositions as an activity of the locutors. Note also that by this definition of the argumentative relation, arguments are not logical propositions but discourse segments. This decision is compatible with the fact that the same logical content can be used for or against the same conclusion, depending on the context.

Q'' Q is the directive act, in ''Although P, Q'' Q is the directive act, and in ''P because Q'', P is the directive act. The full interpretative format for a unit is shown in Figure 8.

We call such a structure an *interpretative format*. The argumentative features have been presented above, other features are described in more details in [Elhadad & McKeown 88].

- The *utterer* of a turn is an abstract entity related to but distinct from the locutor. The two first features, U and LU, account for an important aspect of language use (*polyphony* as presented in [Ducrot 83]) but are not presented in this paper.
- The *illocutionary force* is the final speech act derived from the sentence. It includes any derivation from a surface speech act to an indirect one. For example, for an utterance of "can you close the door?" the value of this feature will often be a type of imperative.
- The *theme* is a set of discursive objects. Discursive objects can be any of the objects, properties or relations denoted by the propositional content, or a property derived from the argumentative, illocutionary or functional specifications of an utterance or a discourse segment. We currently represent the theme as a flat set, but plan on having a hierarchy of themes representing the focusing structure of the utterance.
- The *thematization procedure* indicates how the theme has been promoted to the status of theme. We have identified four types of procedures so far:
  - Propositional content: this is the 'normal' situation, where the theme is chosen from the content of the sentence as in example (1).
  - Argumentative derivation: (cf example (2)). The theme common to the two conjoints of the 'but' is argumentatively derived from the propositional content.
  - Illocutionary thematization: the speech act realized in an utterance becomes the theme of the connection.
  - Re-interpretation: One of the interpretative specifications of a previous utterance becomes the theme of the connection (cf [Fox 87, Roulet *et al* 85] for a presentation).

```
I(s) = [    utterer (U)
            link between locutor and utterer (LU)
            propositional content (PC)
            illocutionary force (IF)
            theme (Th)
            thematization procedure (ThP)
            argumentative orientation (AO)
            discourse law supporting the interpretation (DL)
            functional status (FS)]
```

Figure 8: Interpretative Format

3.2.2 An Abstract Model of Connective Selection

We represent a connective as a *relation between interpretative formats*. Therefore, in order to produce a connective, a generator is provided with a set of interpretative formats as input. If these formats 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.

As an example, consider the description of 'but'. It is a set of constraints on two interpretative formats P and Q. It specifies that:

- the two utterers of P and Q must be distinct:[9] There is a contradiction between what is said in P and what is said in Q. "But" allows to resolution of this contradiction by indicating that the locutor attributes the saying of P and of Q to two different instances (for example, other locutors) that we call utterers, and expresses his/her support for the one expressing Q.
- the intersection of the themes of P and Q must be non-empty. $(Th(P) \cap Th(Q) \neq \emptyset)$: P and Q must be "about the same thing" or related in some way. If it is not the case, the conjunction is odd as in "John is short but the TV costs $300."
- Any kind of thematization procedure can be used for both P and Q: this is very specific to "but" and explains why it is so frequently used. This means that "but" can express a relation between different aspects of the units it conjoins. We have seen examples here of relations between either propositional content or argumentative orientation, but the relation can also be between the illocutionary force of the two units of between reinterpretations of previous units. Most other connectives don't allow illocutionary thematization and reinterpretation.
- 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 than the one mentioned in Q. $(P_i \text{ as in } \sigma) \in AO(P) \text{ and } (Q_j \text{ as in } \sigma) \in AO(Q) \text{ and } (P_i <_\sigma Q_j)$
  If it is not the case, it becomes 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, like in "John is hungry but he is short." If there is a scale in common, but Q has a lesser degree than Q then the preference of the locutor is difficult to directly understand, like in "John is starving but Mary is hungry."
- The topoi used for P and Q must have their right-hand sides of different polarities: if $DL(P) = (..., +\sigma)$ then $DL(Q) = (..., -\sigma)$ 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 the "therefore" links on the argumentative features of P but not of Q.

3.3 Requirements on a Formalism specific to the selection of Connectives

This abstract model of connective selection places certain requirements on a formalism that will be used for the implementation. In particular, we first show the kind of flexibility in order of decision making that is required. This in turn places restrictions on input specification and representation of constraints. It must be possible to partially specify the input to the surface generator and to represent the constraints in such a way that they can be used bidirectionally - as tests for valid values, and as generators of valid values.

---

[9] But the locutors can be the same. This device allows to account for the possibility of a same speaker expressing different points of view.

3.3.1 Order of decision making: Interaction between connective selection and other aspects of generation

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

An example of internal interaction is 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 DL) are constrained. As a consequence, the verbs and adjectives chosen in the connected clauses are also constrained.

For example, consider the case where the unit P has PC = *take(john,mary,book)* (John takes a book from Mary), with the AO containing $\neg$*honest(john)* (John is not honest); the unit Q has PC = *honest(john)*. If the AO is not indicated by anything else, P must be generated as 'John stole a book from Mary'. If, however, the AO can be inferred from the use of a particular connective, it may be possible to use verbs other than "steal". For example, P can be realized as 'John took a book from Mary' as in 'John took a book from Mary but he is an honest person'. In this sentence, the inference that the taking of the book could be considered an illicit act 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 based on 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 'steal' must be used, 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. We have a propogation of constraints from connective down to the clauses it connects and as well, we have a propogation of constraints from decisions made in the clause back up to choice of connective.

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 can hire him.". In contrast, P is directive in "P although Q" and this explains the awkwardness of the sequence "John took a book from Mary although he is an honest person. We can hire him." 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.

3.3.2 Effects on the input: partial specification

Input to the surface generator is a set of interpretative formats which have been filled by the deep generator. Since interaction between the surface and deep generator cannot be uni-directional, the surface generator must be able to send back the formats to the deep planner with new constraints added. This exchange of information implies that the surface generator be able to fill some values of the formats, and behave correctly when some values are not specified in the input.

The argumentative orientation of a format can be unknown, or only partially known in certain situations. Similarly, the model does not require the polyphonic structure of all input formats to be specified. The implementation must therefore allow partial specification of the input.

3.3.3 Effects on the representation of constraints: generate or test

Connectives are represented as relations between formats, or as a set of constraints holding between formats. Since the formats being tested against the constraints are only partially specified, the selection procedure must be able to behave correctly when testing a constraint on a non-specified value.

For example, when testing that the topoi of A and B have right hand sides of opposite polarity, it can happen that the topos of B is not known yet. In this case, it is desirable to add the value of the topos of B based on the constraint that the polarity of its right hand side be the opposite of A's. A more complex example is to add the constraint that $Th(A) \cap Th(B) \neq \varnothing$ when $Th(B)$ is not specified.

This bidirectionality is the main constraint on the representation of connectives as relations. It must also be possible to express these relations in the implementation: the language used should be expressive enough to support concepts such as sets, intersection, order. And of course, the formalism should allow organization of the various constraints of the model in a consistent and readable way: we want to avoid duplication of constraints and to group similar constraints in the same region.

3.4 Implementation

3.4.1 Overview of a FUG for connective selection

We have implemented a FUG that selects connectives from among 'but', 'because', 'since', 'although', 'however' and 'nonetheless'. In this paper, we present a simplified version of this FUG, restricted to 'because', 'since', 'but' and '**although**'. It expects as input an FD of category **discourse-segment**, for which the grammar is shown in **figure** 9. A discourse segment, following [Roulet *et al* 85] 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.

Utterances are represented as interpretative formats, characterized by the nine features presented in paragraph 3.2.1. Note that discourse segments are also characterized by the nine features of the interpretative formats, as

```
;; ================================================
;; 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)))))))))
 (Th ((cat list)))
 (IF (^ directive IF))
 (PC (^ directive PC))
 (U  (^ directive U))
 (AO (^ directive AO)))
```

Figure 9: The CATegory Discourse-segment

these formats are designed to capture the properties of single utterances as well as complex segments. Consistent with Roulet's theory, discourse segments inherit some of their properties (*e.g.*, illocutionary force, argumentative orientation) from their directive act.

The grammar for discourse segments contains a complex alt describing all possible orderings of the propositions, which include DCS, SCD, CSD and CDS.[10] where D is the directive unit, S the subordinate one, and C an optional connective. Whether a connective can be used freely in 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 10). 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 same alt, the **feature connective** is introduced and specified as category **connective**. It has the sub-features P and Q **which linked** to either **subordinate** or **directive**. For the sake of simplicity, in the rest of this presentation we will **assume** that both P and Q are single utterances. The clause section presented in section 2 is used for the generation of each simple clause.

The category **connective**, partially shown in Figures 11 and 12, expects two utterances P and Q as features, and describes the relation that holds between them when the complex PcQ can be realized. It is at the heart of

[10]The order CDS does not seem to be existing in English.

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

Figure 10: Grammar for CONJunctions

our selection procedure. The beginning of the grammar contains the features common to all the connectives described. These state that the themes must intersect (the notation TEST is presented in the next section) and that a single conjunction be used for the connective, as expressed by the **(pattern (c))** expression.

```
;; ==========================================================
;; 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 11: The Connective FUG -- Section 1

The remainder of the connective grammar has been broken into four alts, two of which are shown here. Each alt corresponds to a class of features from the interpretative formats. Functional Status (FS) forms one class and argumentative orientation (AO) and topos, or the discourse law (DL) forms the class, argumentation. The use of alternatives encodes the fact that there is no natural priority in the model between the different features included in a format. 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 is an illustration of the internal flexible order of decision mentioned in page 15.

In the first alt (figure 11), we handle the constraints generated by the functional status. This alt will generate most of the constraints on the ordering of the complex clause. Remember that the features FS of P and Q are filled in the discourse-segment section of the grammar. This section interacts directly with the ordering expressed by the patterns given in the category discourse-segment. Conversely, if constraints on the ordering are generated by the deep planner, they will be introduced in the procedure of connective selection by the bias of the FS feature.

Figure 12 presents the part of the grammar handling argumentative features. For this alt, the AO feature represents a structure *(C_i as in σ)* where C_i is a logical clause and σ is an argumentative scale. The constraint on 'but' that the AO of P must have a lesser degree than the AO of Q is not represented in this fragment. We represent opposition between two AOs using topoi: two AOs are opposed if their respective scales appear in a topos with opposite signs *(+σ1, -σ2)*. All the active topoi are stored in the grammar, in the Topos category. Note that the deep generator may create new topoi dynamically and include them in the grammar. If the topoi are given as input, they do not have to be part of the topoi known to the grammar; if they are not specified, the surface generator will try to find topoi satisfying all the constraints of the relation between the clauses. We have here again an example of bidirectional interaction between the deep and surface generators.

A sample input to the grammar is shown in figure 13. Note that this input contains an argumentative constraint that we assume comes from the deep generator. This constraint states that the clause realizing this proposition must argument for the conclusion that John is dishonest.

The rest of this input FD is mainly a description of the content to be described in conceptual form. Note that the propositional content part (pc) does not include a lexical specification for the verb, but only specifies that the concept to be expressed is "Transfer". Part of the task of the grammar is to choose a verb that will express this concept.

Figure 14 shows a fragment of the lexicon that helps performing the mapping concept to verb. The fragment shows that the verbs "take" and "steal" both can express the concept of "Transfer of possession", but "steal" adds an argumentative instruction to the clause: when a locutor uses the verb "steal", he positions the actor of the clause on the scale of dishonesty[11].

Figure 15 shows how the argumentative constraint given in the input can be satisfied by the grammar. The first

[11]This example is taken from [Raccah 87]

```
;; Second alt: Argumentation
;; AO has 2 (main) features: conclusion and scale.
;; DL has 4 (main) features: sign and scale/left and right.
;; For all but 'but', DL(P) must be none.
(alt
  (((P ((DL none)))
    (alt
      (((Q ((DL ((scale-left  (^ ^ ^ Q AO scale))
                 (sign-right -)
                 (scale-right (^ ^ ^ P AO scale))))))
        (c ((lex "although"))))
       ((Q ((DL ((scale-left  (^ ^ ^ Q AO scale))
                 (sign-right +)
                 (scale-right (^ ^ ^ P AO scale))))))
        (alt (((c ((lex "because"))))
              ((c ((lex "since")))))))))
   ((P ((DL ((scale-left  (^ ^ ^ P AO scale))
             (scale-right (^ ^ ^ Q DL scale-right))))))
    (Q ((DL ((scale-left  (^ ^ ^ Q AO scale))))))
    ;; sign-right of DL(P) and DL(Q) must be opposed
    (alt
      (((P ((DL ((sign-right +)))))
        (Q ((DL ((sign-right -))))))
       ((P ((DL ((sign-right -)))))
        (Q ((DL ((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 12: The Connective FUG -- Section 2

```
C1 =
        ((cat discourse-segment)
         (directive ((th ~(John Mary Book Transfer))
                     (if ((force assert)))
                     (ao ((scale dishonest)
                          (conclusion
                           ((process-type attributive)
                            (carrier === John)
                            (attribute === dishonest)))))
                     (pc ((cat clause)
                          (process-type action)
                          (concept Transfer)
                          (actor ((lex John) (np-type proper)))
                          (benef ((lex Mary) (np-type proper)))
                          (medium ((lex book) (np-type common))))))))
```

Figure 13: Sample input with argumentative constraint

fragment, taken form the grammar for verbs, shows how the argumentative instruction found in the lexicon, if there is one, is sent up to the clause. The feature `justified` in the AO description is then set to the value `yes` indicating that some device from the grammar accounts for the value of the AO in the clause. The second

Lexicon =

```
(alt (((concept Move) ...)
      ...
      ((concept Transfer)
       (lex "take"))
      ((concept Transfer)
       (lex "steal")
       (AO ((conclusion
             ((process-type attributive)
              (carrier (^ ^ ^ pc prot))
              (attribute dishonest)))
            (scale dishonest))))
     ...))
```

Figure 14: A fragment from the lexicon

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 15: Fragments from the grammars for verbs and clauses

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 Transfer is first mapped to the verb "take". But then, no linguistic device realizes the argumentative constraint given in input, and the feature `justified` remains unbound. After checking for the `any` constraint shown in figure 15, this first unification fails. The unmarked verb "take" cannot be used in this context. The unifier then backtracks and tries the verb "steal" to express the concept Transfer. The argumentative constraint of the input is then satisfied by the fragment of the verb grammar shown in figure 15, and the feature `justified` is set to `yes`. The grammar eventually produces the sentence "John stole a book from Mary".

```
C2 =
        ((cat discourse-segment)
         (subordinate
          ((directive ((th ~(John Mary Book Transfer))
                       (if ((force assert)))
                       (ao ((scale dishonest)
                            (conclusion
                             ((process-type attributive)
                              (carrier === John)
                              (attribute === dishonest)))))
                       (pc ((cat clause)
                            (process-type action)
                            (concept Transfer)
                            (actor ((lex John) (np-type proper)))
                            (benef ((lex Mary) (np-type proper)))
                            (medium ((lex book) (np-type common)))))))))
         (directive ((th ~(John Honest))
                     (if ((force assert)))
                     (pc ((cat clause)
                          (process-type attributive)
                          (carrier ((lex John) (np-type proper)))
                          (attribute === honest)))))))
```

Figure 16: Sample input for connective

Figure 16 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 John "tranfered" Mary's book, and that this "tranfer" can be an argument for John's dishonesty, but he states his stronger belief that John is honest in the directive move. The whole move integrates the "bad" argument and resolves the contradiction.

The unification of C2 leads to the generation of ''John took a book from Mary, but he is honest.'' Note that the verb "take" is chosen this time. The first step of the unification will go through the discourse-segment category of figure 9. The first decision will be made in the alt containing the patterns, between the order DcS or ScD. This decision will be later confirmed or rejected by the Functional Status part of the connective category. We follow the path of unification that chooses to use the second branch of this first alt, that is the order ScD is chosen. The second decision concerns the position of the conjunction: the alt allows the ordering ScD or cDS. We follow the first branch, choosing ScD.

The first interesting step is the unification of the connective feature of C2. The unifier goes through the four alts dealing with the constraints on theme, functional status, polyphony and then 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 C2.

Argumentative features added to C2:

```
((directive
   ((AO ((conclusion ((process-type attributive)
                      (polarity negative)
                      (carrier === John)
                      (attribute === dishonest)))
         (scale dishonesty)))
    (DL ((scale-left honesty)
         (sign-left +)
         (scale-right dishonesty)
         (sign-right -)))))
 (subordinate
   ((AO ((conclusion ((process-type attributive)
                      (polarity positive)
                      (carrier === John)
                      (attribute === dishonest)))
         (scale dishonesty)))
    (DL ((scale-left Tranfer)
         (sign-left +)
         (scale-right dishonesty)
         (sign-right +))))))
```

Figure 17: Argumentative features added to C2

Assuming that the topoi *(+honesty, -dishonesty)*[12] and *(+Tranfer, +dishonesty)*[13] are in the grammar, the argumentative features in figure 17 will be added to C2. 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 "take" is chosen, and the choice needs not be reconsidered, since the input constraint is already satisfied. The sentence eventually produced is ''John took a book from Mary but he is honest.''

This example demonstrates how a 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 what is the interaction. Similarly, if the deep planner had any constraints on lexical choice for example, 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.

[12]The scales honesty and dishonesty are two distinct scales [Anscombre & Ducrot 83]

[13]Actually, the left side of this topos need not be Transfer. Since the AO is given in the input as a constraint generated by the deep generator, the grammar does not have to find a complete topos justifying the AO, but it will still instantiate correctly the right hand side.

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;patten88] and MUMBLE [McDonald 86].

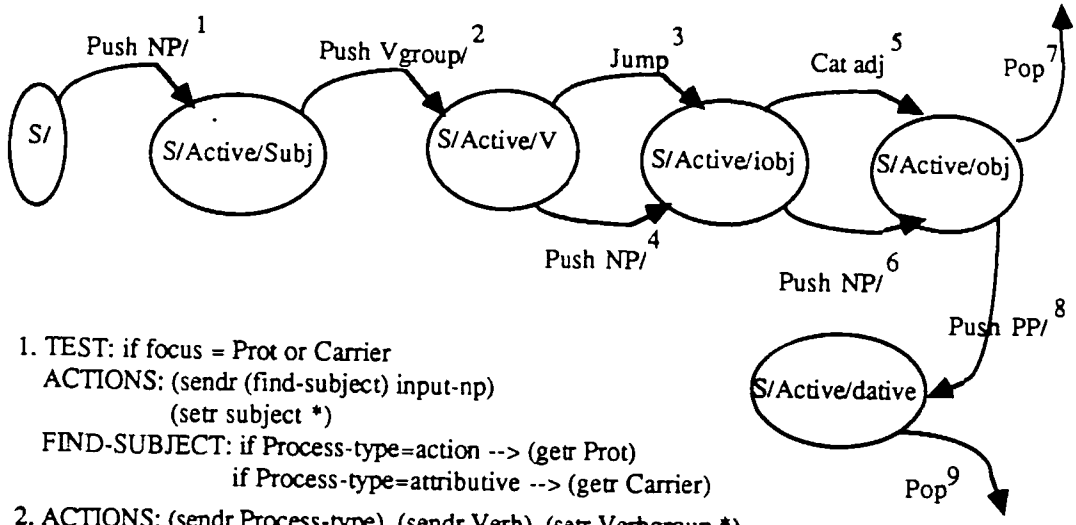
4.1 Using an ATN for Generation

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 should string 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 decision 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 18: 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 18 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.[14] Assignment of semantic roles to syntactic ones is done by complex actions on the arcs.

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 by the test. This arc would

[14]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.

produce the NP for the subject, "John" and place it in the subject register. Arc 2 would be taken next which produces the verb "gives". At node S/Active-V, the system must decide whether to generate the indirect object first (if there is one) or to generate the object first. Note that this is a decision about ordering of syntactic constituents that gets determined after other ordering decisions (e.g., to use the active form) and after all decisions about other embedded constituents (e.g., subject and verb group) have been made. In this case, arc 3 will be traversed since there is a **dative prep** in the input. Finally, at node **S/Active-Iobject**, the system must determine whether to produce an adjective or NP as direct object. This decision is based on semantic type of the clause (i.e., an adjective can only be produced if **process-type** is **attributive**) in addition to whether the concept can be realized as an adjective.

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 smantic roles to syntactic roles, building of syntactic structure, and ordering of roles all occur simultaneously. This is but one way of translating the FUG. There are a variety of other possibilities. Separate networks could be constructed for each of the process types in both active and passive forms. For example, in the network for process type action, active form, the first **push np/** arc would send the **prot** register down to the np network and would set the subject register to the results of the subnetwork. Another possible translation could use the buildq actions on the pop arcs totally for determining ordering and allow arcs to occur in any order in the network. Such a translation might have the production of the verb group occurring before the production of the subject simply by changing arcs 5 and 6 in Figure 18. While this would be possible, the resulting grammar could certainly not be used for interpretation as well, and would be quite different from the normal use of ATNs. Yet another possible translation would be to follow the FUG even more exactly, creating 3 stages in the ATN, where the first two stages resulted only 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 translation would only use **test** and **jump** arcs and would also not correspond to the normal use of ATNs.

4.1.2 Characterization of Differences

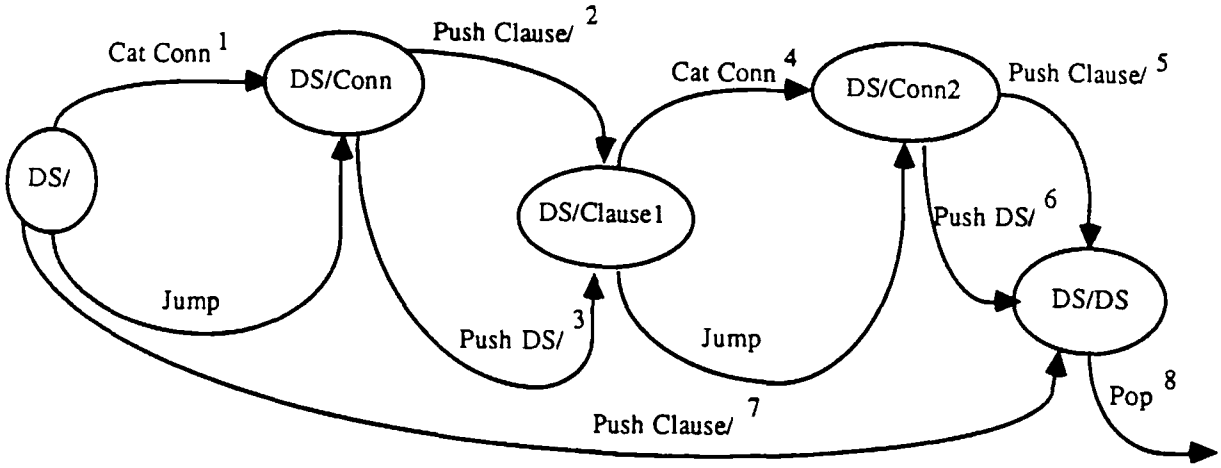
In the ATN version of the FUG that we presented here, the ordering of constituents in the resulting sentence is conflated with the assignment of syntactic structure to constituents while in FUG these tasks were represented in two separate sections of the grammar. The necessity to do some sort of mixing of the different sorts of information sorted out in the three stages of the FUG as well as the depth-first traversal algorithm of the ATN results in these primary differences in order of decision making and grammar representation:

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. This 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.

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 19: Top level Network for Complex clause

Figure 19 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 need therefore to 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. What that means is basically that the procedure of selection 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 as only testers or also 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.

If the deep planner decides to use a certain lexical item, e.g., the verb 'to steal', the decision will be represented by filling the register verb in the input. In the FUG implementation, the Utterance category derives all the constraints from this selection (e.g., the argumentative constraint on the actor of the verb). These constraints can then be used to choose the appropriate connective. In the ATN implementation, these constraints will not be used unless the connective procedure checks all the possible sources of such constraints, e.g., the code of the procedure must include a test of the value of the verb register.

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 derives of course from our desire to leave the 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 assign 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 (e.g., 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 20. 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).

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

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

Figure 20: Sample DCG Rule

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 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 use 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).

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 a *chooser* function associated with the system, which in turn invokes one or more primitive *inquiry operators*. These functions will query other modules 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"[15] feature is input which means that the path from higher level systems to that lower level feature can be avoided in processing.

Thus in NIGEL, decision making, as it is implemented, is primarily top-down and is not inherently bidirectional. However, since systems are organized around the paradigmatic axis (i.e., a systemic grammar gives priority to the functional status of language and choosers typically query semantic and pragmatic aspects, while their results instantiate syntactic features), order of decision making is very different from either of the parsing formalisms discussed. 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.

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

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 that we have argued for 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.

4.4 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. When a tree structure has been added, it is also traversed, again replacing plan units when they are found. The second main difference is McDonald's commitment to a linear algorithm. All decisions are indelible. This is in direct contrast to the unification algorithm of FUG. While a linear algorithm is ultimately preferable to a nondeterministic algorithm, it is unclear at this point how McDonald would encode the bidirectional constraints and complex interaction between constraints that we have argued are needed.

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 layout 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. Organizing a FUG is not an easy task in general however, and, unfortunately, can lead to major inefficiencies.

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 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), limit 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 want 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 extended to many unifiers ([Shieber 86, Kartunnen 84]). 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 +))))))
```

The 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. The representation of sets is not easy to determine. The model can be changed slightly, and all references to sets changed to lists or FDs. This is, however, a limitation of the formalism.

In addition, there is nothing in the formalism to handle numbers and arithmetic operations. It is of course 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 88].

We have introduced into the formalism a special attribute, **test**, that has a special unification behavior adding 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 insure 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.[16] 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))))
```
[17]

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 arbitrary complex

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

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

constraints between any decision, these interactions can be quite complex, and cannot be handled by a module that would not be aware of the purely linguistic intricacies. It is therefore natural to let the linguistic component deal with the interactions, in the most flexible way.

We have shown that FUGs allow for that flexibility by looking at the task of connective selection. FUGs allow for flexible internal order of decision, and provide tools to organize the grammar without duplication of constraints, and with a clear grouping of similar constraints. 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 & 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.

[Cohen 87] Cohen, R.
Analysing the structure of argumentative discourse.
Computational Linguistics, 13(1-2):11-24, 1987.

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

[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.

[Ducrot *et al* 80] Ducrot, O. *et al.*
Le sens commun: Les mots du discours.
Les editions de Minuit, Paris, 1980.

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

[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.

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

[Fox 87] Fox, B.A.
Interactional Reconstruction in Real-time Language Processing.
Cognitive Science, 11:365-387, 1987.

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

[Grosz & Sidner 86]
Grosz, B. and Sidner, C.
Attentions, intentions, and the structure of discourse.
Computational Linguistics, 12(3):175-204, 1986.

[Hirschberg & Litman 87]
Hirschberg, J. and Litman, D.
Now let's talk about Now: identifying clue phrases intonationally.
In *Proceedings of the 25th Conference of the ACL*, pages 163-171. Association for Computational Linguistics, 1987.

[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.
In *Presented at the Workshop on Natural Language Generation.* 1988.

[Kartunnen 84] Kartunnen, 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 Berkely Linguistics Society.* Berkeley Linguistics Society, 1979.

[kukich 83] kukich, k.
design of a knowldege based text generator.
In *proceedings of the 21st acl conference.* acl, 1983.

[List 84] List, K.L.
Coherence and Cohesion: contextualization of Oswald Ducrot's general theory of linguistic semantics.
PhD thesis, University of Michigan, 1984.

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

[Mann & Mathiessen 83]
mann, w.c. and mathiessen, c.
nigel: a systemic grammar for text generation.
Technical Report isi/rr-83-105, usc/isi, 1983.

[mathiessen 81] mathiessen, 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.
In *Presented at the Workshop on Natural Language Generation*. 1988.

[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.

[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.
In *Presented at the Workshop on Natural Language Generation*. 1988.

[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.

[Reichman 85] Reichman, R.
Getting computers to talk like you and me: discourse context, focus and semantics (an ATN model).
MIT press, Cambridge, Ma, 1985.

[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.
In *Presented at the Workshop on Natural Language Generation*. 1988.

[Searle & Vanderveken 85]
Searle, J. and Vanderveken, D.
Foundations of Illocutionary Logic.
Cambridge University Press, Cambridge, 1985.

[Shieber 86] Shieber, S.
CSLI Lecture Notes: An introduction to Unification-Based Formalisms.
CSLI, 1986.

[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.