PHRED: A GENERATOR FOR NATURAL LANGUAGE INTERFACES[1]

Paul S. Jacobs[2]

Berkeley Artificial Intelligence Research
Division of Computer Science
Department of EECS
University of California
Berkeley, CA, USA

PHRED (PHRasal English Diction is a natural language generator designed for use in a variety of domains. It was constructed to share a knowledge base with PHRAN (PHRasal ANalyzer) as part of a real-time user-friendly interface. The knowledge base consists of *pattern-concept pairs,* i.e., associations between linguistic structures and conceptual templates. Using this knowledge base, PHRED produces appropriate and grammatical natural language output from a conceptual representation.

PHRED and PHRAN are currently used as central components of the user interface to the UNIX Consultant System (UC). This system answers questions and solves problems related to the UNIX[3] operating system. UC passes the conceptual form of its responses, usually either questions or answers to questions, to the PHRED generator, which expresses them in the user's language. Currently the consultant can answer questions and produce its responses in either English or Spanish.

There are a number of practical advantages to PHRED as the generation component of a natural language system. Having a knowledge base shared between analyzer and generator eliminates the redundancy of having separate grammars and lexicons for input and output. It avoids possibly awkward inconsistencies caused by such a separation, and allows for interchangeable interfaces, such as the English and Spanish versions of the UC interface.

The phrasal approach to language processing realized in PHRED has proven helpful in generation as in analysis. PHRED commands the use of idioms, grammatical constructions, and canned phrases without a specialized mechanism or data structure. This is accomplished without restricting the ability of the generator to utilize more general linguistic knowledge.

As the generation component of a natural language interface, PHRED affords extensibility, simplicity, and processing speed. Its design incorporates a cognitive motivation as well. It diverges from the traditional computational approach by focusing on the use of specialized phrasal knowledge. This phrasal approach minimizes the autonomy of the individual word, the bane of some earlier approaches to language processing. The two-stage process used by PHRED to select appropriate linguistic structures also fits well with cognitive theories of language and memory.

1 INTRODUCTION

The PHRED (PHRasal English Diction) system is a language generation module for natural language interfaces. The generator operates from a declarative knowledge base of linguistic knowledge, common to that used by PHRAN (PHRasal ANalyzer; Wilensky and Arens, 1980). PHRED and PHRAN together form an interface for analyzing natural language and producing natural language responses. This interface serves as the linguistic component to the UNIX Consultant system (UC) (Wilensky, Arens, and Chin 1984), a program for responding to inquiries about the UNIX operating system. As the entire UC system operates in several seconds of CPU time, it is an important feature of PHRED that it requires no more than two or three seconds to produce a complete sentence.

The principal knowledge structure used by PHRAN and PHRED is the **pattern-concept pair,** which links a phrasal pattern to a conceptual template. This structure has proven particularly effective in the encoding of specialized linguistic knowledge, i.e., knowledge about

0362-613X/85/040219-242$03.00

particular phrases and their specialized meanings. Part of the theoretical basis of PHRED is the notion that such specialized constructs are an essential component of language use. This idea has among its advocates Chafe (1968), Harris (1968), and Kittredge and Lehrberger (1983), and is behind other generation systems such as Kukich's Ana (1983).

The shared linguistic knowledge base is an unusual feature of PHRED and PHRAN. Computer programs that can effectively communicate in natural language must be capable both of analyzing a range of utterances to derive their meaning or intent, and of producing appropriate and intelligible responses. Historically these two tasks have been treated independently, principally because some of the hard problems in language production differ from those of language analysis. In the MARGIE system, for example, the BABEL generator (Goldman 1975) employed a discrimination net as its principal data structure to facilitate the selection of an appropriate verb and an ATN grammar to apply syntactic constraints, while the ELI analyzer (Riesbeck 1975) in the same system attached routines to individual words to control the interpretations considered during the parsing process.

Throughout the short history of natural language generation systems, programs that produce language have treated generation as a process of *decision making* (McDonald 1980), *choice* (Mann and Matthiessen 1983), or *planning* (Appelt 1982). These systems have employed knowledge structures specifically geared, to varying degrees, to the task of constraining the selection of lexical and grammatical elements. The design of analyzers, on the other hand, focuses on the problem of ambiguity in natural language and makes use of knowledge structures designed to constrain the consideration of alternative interpretations. While the *tasks* of analysis and generation are thus inescapably different, much of the same *knowledge* can be used in performing both tasks.

Even in systems with both analysis and generation components, the knowledge used to derive meaning from language is not used to produce language from meaning. Such systems may be able to use a word or grammatical structure without being able to recognize the same structure, or vice versa, and must duplicate a great deal of information if the generator uses language similar to that understood by the analyzer. Intuitively, it seems that the knowledge used to constrain the interpretation of language can be used to constrain the choice of language. A natural language system with a parsimonious knowledge representation could encompass an interface capable of both analysis and production without excessive duplication, with knowledge *about* language as well as mechanisms for its analysis and generation.

PHRED was conceived as the generation component of such an interface. PHRED, along with its companion analysis program PHRAN, embodies an approach to natural language processing founded on the principle that knowledge about language is in essence declarative, consisting of associations between linguistic and conceptual structures of varying degrees of specificity (Wilensky 1981). Such declarative knowledge should be used for both analysis and generation, and the form of the knowledge should be independent of the particular language. From its initial conception the "English" part of PHRED's acronym was anomalous, as both the analyzer and generator were envisioned as being able to utilize knowledge bases of multiple languages.

The UNIX Consultant, designed to aid novice users of the UNIX operating system by conversing in natural language, is one application of the PHRED generator. The PHRAN analyzer receives the user's input and passes a conceptual interpretation to UC, which then passes its response in conceptual form to PHRED to be converted into natural language output. Below are some examples of questions to UC, with the responses generated by PHRED:

User: How do I delete a file?
UC: **Typing 'rm filename' causes the file filename to be removed from the current directory.**
User: Cómo se borra un archivo?
UC: **Para borrar el archivo de nombre filename del espacio de trabajo, se escribe 'rm filename'.**
User: How do I delete all the files from a directory?
UC: **Typing 'rm *' will remove all files within the current directory from the current directory.**

Typing 'rm -r directoryname' will cause all files and directories recursively inside of directoryname to be deleted from directoryname and removes the directory directoryname from the working directory.

The Spanish example above, a rough paraphrase of the first English question, illustrates the ability of UC to answer questions posed in either language, using the same knowledge about UNIX. The PHRAN/PHRED knowledge base includes the information that *borrar* and *escribe* used with certain objects refer to specific concepts in the UNIX world, the same concepts as *deleting* and *typing*, respectively. Also in the phrasal knowledge base is knowledge that *espacio de trabajo* and **working directory** have a specialized denotation in the UNIX world. Such specialized linguistic knowledge is common in this domain.

While PHRAN and PHRED were originally tested using an English vocabulary used for various stories and news articles, it was a relatively easy task to accommodate linguistic knowledge bases for English and Spanish in order for the same programs to operate in the UC domain. Adding a new vocabulary or language capability to the UC system has required no modification to the program, although the system has not had extensive testing with many languages.

PHRED is implemented in Franz LISP and runs compiled on a VAX 11/780. The English linguistic knowledge base of UC contains about 150 patterns, in addition to knowledge of the morphological characteristics of 30

verbs and 50 nouns commonly used in communicating UNIX information. The compiled program occupies about 100K bytes of memory, of which about 20K is code used also by PHRAN. Output from PHRED in the UC system requires 1-3 seconds of CPU time, roughly a third of the total time used by the system. For sentences of the length typically produced by the generator, the amount of time used is roughly proportional to the length of output. Experiments with larger knowledge bases have suggested that the time used by the generator is not heavily dependent on the size of the knowledge base.

The next section describes the PHRED knowledge base and outlines its role in the generation process. Section 3 covers this process in more detail, and Section 4 traces a complete example of generation using PHRED. Section 5 compares the PHRED approach with other research. Section 6 discusses some current and future research directions.

2 The PHRED Knowledge Base

The knowledge base shared by the phrasal analyzer (PHRAN) and phrasal generator (PHRED) consists of pattern-concept pairs, where the pattern contains a linguistic structure and the concept its internal representation. While this representation may be classified as within the systemic/functional tradition (cf. Halliday 1968, Kay 1979) the implementation of the PHRED knowledge base differs in certain important details. The use of the PC pair in PHRED may be distinguished from some other language production mechanisms (McDonald 1980, Mann and Matthiessen 1983, McKeown 1982) in which grammatical information and conceptual information are separated: The "pattern" component of each PC pair may include conceptual information, and the properties associated with each PC pair may combine linguistic and conceptual attributes. Like the systems described above, however, PHRED uses these properties for indexing and applying each pattern, particularly using information about agreement among constituents of the pattern and relationships between properties of constituents and properties of the entire pattern.

The following is a simple example of a pattern-concept pair, representing some of the knowledge about the use of the verb *remove*:

```
PATTERN:
   <agent> <root = remove> <physob>
      <<word = from> <container>>
CONCEPT:
   (state-change (object ?rem-object)
                 (state-name location)
                 (from (inside-of (object ?cont)))
                 (to (not (concept (inside-of
                                     (object ?cont))))))
PROPERTIES:
   tense = (value 2 tense)
   rem-object = (value 3)
   cont = (value 5)
   forms = (active-s passive-s)
```

Specifications of components of the pattern in angle brackets (< >) include linguistic information (root = remove) or conceptual categories (agent, container) or a combination of linguistic and conceptual specifications. Additional information associated with each PC pair determines the correspondences between elements of the conceptual structure and constituents of the linguistic structure: The special "value" indicator designates the association of a property of the PC pair with a property of one of its constituents, specified by number. Thus "tense = (value 2 tense)" implies that the tense of the pattern is the tense of the second constituent, the verb. "cont = (value 5)" indicates that the token unified with the variable "?cont" in the conceptual template corresponds to the fifth constituent, the object of *from*. The above PC pair can be used by PHRED, depending on the concept being expressed, to produce the sentence *You should remove the files from your directory.* or the infinitive phrase *to remove a file from the top level directory.* The final output is determined by the combination of this PC pair with the input attributes and one or more **ordering patterns,** which embody general linguistic constraints and constraints on surface order.

In addition to the linguistic patterns and associated conceptual representation, PC pairs contain a set of properties, or attributes, and other information that guides their use. Some of this information, such as "tense = (value 2 tense)" above, is used to determine correspondences between a pattern and its constituents. Other properties are used for indexing purposes. There is also a facility for "escapes", or the ability to call a special procedure from within the declarative knowledge representation. While this facility was often exploited in early versions of PHRAN, it is problematic for knowledge bases shared with PHRED. Procedures called during analysis are seldom useful to the generator or vice versa. Therefore such procedure calls have seldom been used in PHRED, and an attempt has been made to encode all knowledge in a declarative form that can be used by both the generator and the analyzer.

The "pattern" part of the PC pairs is a list of constituents, where each constituent in a pattern is generally described either as a pattern of speech (p-o-s) or as a member of a descriptive category (e.g., person, physical object). Patterns may also be formed by conjunction and disjunction of other patterns and may contain specifications of constraints. For example, the constituent

```
<and root = remove voice = active form = infinitive>
```

is a single-constituent pattern that would generate the infinitive verb *to remove*, while

```
<and p-o-s = noun-phrase> <or person physob>>
```

represents a noun-phrase that refers to a person or physical object.

Patterns are used to represent lexical entries, determiners and particles which refer to nothing, as well as very specific phrases which refer to particular objects. The pattern

<word = the> <word = big> <word = apple>

represents the phrase *the big apple* used to refer to New York City. This phrase can also be produced by the general pattern

<p-o-s = article> <p-o-s = np2>

when used to refer to an apple.

Specialized linguistic constructs are often partially frozen patterns that behave as a particular grammatical unit. The phrase *kick the bucket* behaves as a verb that conjugates but does not passivize. It corresponds to the pattern

<and p-o-s = verb root = kick>
<word = the> <word = bucket>

which functions as an intransitive verb.

Part of the knowledge associated with a pattern-concept pair is the correspondence between the properties of the pattern's constituents and the properties of the entire pattern. Associated with the *kick the bucket* pattern above is the knowledge that the person, number, and tense of the pattern correspond to the person, number and tense of the first constituent, the form of the verb *kick*. In generation, this results in the recursive application of constraints from a pattern to its components: To generate a past-tense verb meaning *died*, the system will operate recursively on the pattern above to generate a past-tense form of *kick*.

Patterns do not necessarily represent a fixed word order. For example, in

<person> <root = tell> <person>
<word = to> <word = get> <word = lost>

the pattern retains its meaning when used in a passive form or infinitive phrase. Such patterns are used in combination with ordering patterns, which control the various ways in which a pattern may be linguistically realized. An example of an ordering pattern that could be used in conjunction with the *get lost* pattern above is the passive infinitive ordering, used to produce, for example, *the man to be told to get lost* or *the file to be removed from the current directory:*

PATTERN:

<and #3 p-o-s = noun-phrase case = objective>
<and #2 p-o-s = verb form = infinitive voice = passive>
<<word = by>
<and #1 p-o-s = noun-phrase case = objective>>
<<#rest>>

PROPERTIES:

p-o-s = inf-phrase
voice = passive
forms = (passive-s)

The "#2" and "#3" within the ordering pattern indicate that the constraints on the second and third constituents of the coordinated pattern are conjoined with the first and second constituents of the ordering pattern, respectively. The "#rest" indicates where additional constituents are generally inserted. This information guides the combination of the ordering pattern with other PC pairs. An extra set of angle brackets is used to mark a constituent that is optional to the pattern, such as the *by* phrase. The "p-o-s = inf-phrase" property specifies that the pattern produces an infinitive phrase, and the "forms = (passive-s)" property restricts the use of this ordering to patterns which have "passive-s" among their forms.

Patterns that have an unspecified word order do not have a "p-o-s" attribute, and thus do not produce a particular pattern of speech independently. These are combined by PHRED with ordering patterns to allow for idioms or expressions which may appear in various forms, such as *bury the hatchet* in *The hatchet was buried at Appomattox*. The same effect could be accomplished without ordering rules by increasing the number of fixed-word-order patterns combinatorially. The use of the ordering patterns, however, has a certain elegance as well as a practical value: it allows the specification of certain specialized constructs as relations among particular constituents, regardless of where the constituents appear in the actual output. In this case, the specialized meaning of *telling someone to get lost* is effectively represented by the relationship between the verb *tell* and its complement *to get lost*. This meaning may be realized in a variety of forms; for example, the combination of the *get lost* pattern with a passive ordering may produce the sentence *John was told by Bill to get lost.*

While there are similarities between the ordering rules used by PHRED and transformational grammar rules, there are some important differences: PHRED assumes no syntactic derivation; rather, the final ordering of a pattern of speech is produced by combining a set of linguistic patterns. Furthermore, there is no strict sequence in which the patterns must be applied: A given ordering pattern may be chosen either before or after a pattern with which it is to be combined. The combina-

tion of ordering patterns is constrained by the interactions among the properties of the patterns, instead of by controlling the order in which they are used. In this way PHRED is more flexible than other systems that handle word order as a final phase of the generation process (cf. Goldman 1975).

The pattern-concept pair representation falls into a class of linguistic representations known as **feature systems,** including lexical functional grammar (Bresnan 1982), functional grammar (Kay 1979) and functional unification grammar (Kay 1984). These systems, which developed in parallel, may be described using a common notation, and vary mostly in the way in which they are typically applied. Pattern-concept pairs have been applied primarily to the problem of representing the specialized linguistic knowledge that seems necessary to use language as a communicative tool. This emphasis causes minor variations to seem important. For example, most unification grammar implementations require that a syntactic category be among the features, or attributes, of every linguistic pattern. The omission of this requirement for pattern-concept pairs facilitates the representation of patterns that have a specialized meaning but do not have rigid surface structures. This is illustrated by the *get lost* pattern and by the specialized knowledge about *borrar* and *escribe* in the UNIX domain.

The next section describes how the knowledge base described here is utilized as part of a real-time generation system.

3 The Generation Process

The production of an utterance in PHRED is a recursive process which can be divided into three phases:

- **Fetching** is the retrieval of pattern-concept pairs from the knowledge base.
- **Restriction** consists of validating a potential pattern-concept pair to confirm that it fulfills a given set of constraints and adding new constraints to the pattern.
- **Interpretation** is the generation of lexical items that match the constraints of the restricted pattern.

The generation algorithm implemented in PHRED is similar to those used in other unification-based systems (cf. McKeown 1982, Appelt 1983). Because of the expectation that PHRED would serve as part of a real-time interface, however, the system was designed to avoid the expensive unification process. Thus the fetching phase of PHRED accomplishes much of the task of checking the constraints of a pattern against the constraints to be satisfied, a function that could be performed by unification. The more time-consuming unification process is applied only after the fetching phase has produced a candidate pattern.

A second important aspect of the PHRED algorithm, also addressed to the problem of avoiding unnecessary computation, is the overall strategy for handling alternative patterns. Once the fetching mechanism has retrieved a pattern, PHRED uses this pattern unless it is found to violate a constraint. This is similar to the strategy implemented in MUMBLE (McDonald 1980), which also avoids comparisons of linguistic structures of comparable validity. Unlike MUMBLE, PHRED does limited backtracking under some circumstances. The backtracking mechanism, however, relies on the fact that the fetching mechanism generally produces *some* useful patterns and that most constraint violations are due to incorrect selections among ordering patterns.

Each of these phases and its role in the generation process will now be discussed in further detail.

3.1 Fetching

While PHRAN and PHRED use the same knowledge structures, the way in which these structures are accessed for the purpose of generation naturally differs from their access by the analyzer. PHRAN must recognize a set of lexemes as possibly corresponding to a pattern and thereby retrieve an appropriate pattern-concept pair from the knowledge base. PHRED, on the other hand, accesses the knowledge base by fetching pattern-concept pairs whose template fits the concept and constraints to be expressed.

Because fetching can be a time-consuming part of the generation process, it is important for the fetching mechanism to operate efficiently, but also to produce only those PC pairs likely to be useful. For this purpose, PHRED uses a hashing scheme designed to produce an ordering, or stream, of candidate patterns with a minimum of computation. Specifically, it performs some quick computation to select a sequence of PC pairs that might be of help in constructing a particular utterance. These pairs are then considered as PHRED continues its work. As the generator uses the first available appropriate utterance rather than evaluate all potential candidates, the ordering of this stream influences the choice process as well as the number of patterns ultimately considered.

The implementation of PHRED permits conceptual attributes to influence the search of linguistic alternatives, but separates this process from other aspects of language planning. High-level text goals are not included in the knowledge structures that influence the fetching process. In this regard, the system within which PHRED operates does not promote the desirable interaction among text planning and structural choices, as suggested by Appelt (1982) and Danlos (1984). Higher-level planning in the UNIX Consultant, for modularity, is performed by a separable planning component.

The role of fetching in PHRED is to provide access to the pattern-concept pairs in the knowledge base. The input to the fetching mechanism is a set of constraints and conceptual attributes. Using this input as a guide, the fetching mechanism chooses PC pairs that serve as building blocks for the language produced. The pattern components of these PC pairs may include general

patterns and ordering patterns as well as specialized phrases and lexical choices.

In producing the phrase *the file filename to be removed from the current directory,* the fetching stage is given the following input:

```
p-o-s = inf-phrase
voice = passive
concept =  (state-change (object file1)
                         (state-name location)
                         (from (inside-of (object current-directory)))
                         (to (not (concept (inside-of (object current-directory))))))
```

From this input, the fetching mechanism must retrieve the *remove* pattern shown earlier as well as the ordering patterns necessary to produce a passive infinitive phrase.

The design of the hashing scheme that accomplishes this retrieval is based on the following reasoning: The input to the fetching mechanism may be described at least in part by a set of conceptual and linguistic properties, as may the pattern-concept pairs in the data base. The process of restriction, described in section 3.2, relies heavily on matching these two sets of properties. This process may therefore be expedited by computing an address in memory that "points" to PC pairs with a particular set of attributes. Since there are combinatorially many such sets, there must be (1) a large number of addresses, or "buckets", and (2) an effective means of selecting which sets of "important" properties to use in computing each address.

The selection of "important" properties is determined as follows: All conceptual attributes, including those included within the concept part of the input, are considered important, and the linguistic attributes used for each p-o-s type are specified in the knowledge base. The fetching mechanism first searches buckets found through large sets of attributes, then buckets that correspond to smaller sets of attributes. The idea of this process is to consider first the PC pairs that most closely fit the input to the fetching mechanism. Since a hash into an empty bucket takes very little time, there is no great loss of time efficiency in using a fairly large number of hashes. Although the access to a PC pair through multiple buckets requires some additional space, this space is negligible compared to the size of the knowledge structures themselves.

The fetching component of PHRED, like the other parts of the system, is geared towards simplicity and uniformity. In spite of some of the differences among, for example, the selection of a verb, the choice of a referring expression and the selection of an article that agrees with its head noun, the same method is used for fetching in all three cases. The same hashing scheme is employed also to retrieve ordering patterns from the knowledge base. Such orderings can be effectively retrieved from their attributes in the same manner that any other PC pair is fetched. Thus, while the nature of the knowledge contained in the attributes of a lexical structure is arguably different from the knowledge within an ordering PC pair, these different types of knowledge may be accessed through the same mechanism. The principle behind this uniformity is that the level of specificity of the knowledge required to realize particular concepts and constraint cannot be predetermined; thus general and specific knowledge should be accessed in the same fashion.

The main loop of PHRED passes to the fetching component the set of constraints a PC pair must satisfy. Typically, if there is a specific phrase, structural formula or other pattern that directly satisfies these constraints, it will appear in the stream before more general patterns. A pattern of unfixed word order will generally appear in the stream before an ordering pattern, because the ordering patterns tend to have few or no conceptual attributes. Most often, the unfixed pattern is chosen based on the concept passed to the fetching mechanism, while the ordering pattern is chosen to select an ordering that produces the appropriate pattern-of-speech. The manner in which these patterns are combined is discussed in section 3.2. The fetching mechanism is repeatedly called to return patterns from the stream until all possible constraints are satisfied. For example, to produce the phrase ... *not to remove the file,* a negative ordering, infinitive, and *remove* pattern must all be fetched before the phrase can be restricted.

The construction of hash keys based on successively smaller sets of attributes assures that the PC pairs whose concept most closely matches the input concept will be considered first. The fetching mechanism produces a stream of pattern-concept pairs which are returned one at a time as they are requested by the generator. The rest of the program is insulated from the retrieval process. This way, some of the hashing computation can be postponed until it is required.

In the case of the *remove* example given above, the PC pair is indexed according to a combination of the semantic attributes "state-change", "location", "inside-of", and "not-inside-of". This combination is used at the time the PC pair is read in to determine which buckets should include the PC pair. The indexing mechanism ignores variables (e.g., "?actor"). During generation, a bucket indicating this PC pair will be found, based on the same semantic attributes. Some empty buckets, based on different combinations of attributes, will be searched also. A bucket including the passive infinitive ordering pattern is found by using the p-o-s and voice attributes.

Buckets that correspond to more complete sets of attributes are searched first. For example, if the "delete" pattern were constrained to be used only for the deletion of files, it would be retrieved before the *remove* PC pair because the bucket identified by the conjunction of the "file" attribute of file1 with the other semantic attributes of the concept is searched first.

A simple pattern, such as the word *the,* does not really have a concept associated with it, and thus is indexed according to sets of its linguistic attributes only: A search for a definite article would find a bucket based on the properties "p-o-s = article" and "ref = def" and would thereby yield the pattern for "the".

The fetching component of PHRED constitutes about 10K bytes of object code, one tenth of the total program. A profile of PHRED shows that more than half of the CPU time consumed by the generator is spent in the fetching process. Earlier versions of the program, which did no ordering of candidate patterns in the fetching phases, spent less time fetching but more time overall.

When the fetching mechanism retrieves a pattern which has the appropriate "p-o-s" attribute, control is passed to the restriction phase. This phase is considered below.

3.2 RESTRICTION

Each time a candidate pattern is returned from the stream by the fetching mechanism, it is passed to the restriction phase, along with any other unfixed-order patterns which have been retrieved. The restriction mechanism creates an instance of the pattern, adding new constraints to the pattern constituents while verifying that the PC pair meets the constraints given. There are three main aspects of this process:

- **unification** of the variables within the PC pair's conceptual template and its associated properties with the target concept and properties,
- **elaboration** of the pattern constituents to include properties from corresponding properties in the pattern indicated by the "value" marker, and
- **combination** of the properties of constituents among the pattern and ordering patterns.

The following is an example of an instance of the *remove* PC pair given earlier, after restriction:

```
PATTERN:
   <and object  concept = file1  p-o-s = noun-phrase  case = objective>
   <and  root = remove  form = infinitive  voice = passive>
   <<word = by>
    <and  agent  concept = user1  case = objective>>
   <<word = from>
    <and  container  concept = directory1  case = objective>>
CONCEPT:
   (state-change  (object file1)
                  (state-name location)
                  (from (inside-of (object directory1)))
                  (to (not (concept (inside-of (object directory1))))))
PROPERTIES:
   p-o-s = inf-phrase
   tense = (value 2 tense)
   rem-object = file1
   cont = directory1
   forms = (passive-s)
```

This PC pair is the product of applying the restriction process twice in succession, once to the passive infinitive ordering and once to the *remove* pattern. Unification has occurred to bind the variables "?cont" and "?rem-object". Elaboration has added the tokens bound to these variables to the individual constituents. Combination of the *remove* pattern with the passive infinitive ordering has produced a pattern whose constituents are specified by the conglomeration of constraints of the PC pairs used.

Any of these three aspects of the restriction phase may result in failure. In the above example, unification would fail in an attempt to bind the multiple occurrences of "?cont" to different tokens, or if some variable binding violated an input constraint. Elaboration results in failure if a property to be added to a constituent does not fit the other properties. For example, if "directory1" in the example is not a container, the pattern would be judged inappropriate. Combination could likewise result in failure if the constraints from the ordering rule were incompatible with those from the *remove* pattern, for example, if it had no passive form.

Properties marked by "value" in the PC pair are treated as variables and unified along with the other properties. If these variables remain unbound throughout the restriction process, however, the pattern retains the property with its "value" marker. This is necessary for future stages of the production process to obtain the property on demand. For example, a noun-phrase pattern in Spanish, where there is gender agreement between the subject of a passive infinitive phrase and the past participle, maintains the "gender = (value 2)" property to

reflect that the gender of the NP is the gender of its NP2 This property is not determined until the head noun is chosen, after which it can be retrieved through the NP if necessary.

Restriction uses about 60% of the code of the generator and most of the CPU time not consumed by fetching. The bulk of this time is spent doing repeated unification when a large number of patterns are required. Because the nature of the knowledge structures in the system seems to require such unification, the fetching mechanism, as described in section 3.1, is designed to prevent the consideration of patterns which might lead to failure during unification.

The next step in the generation process, after restriction, is to go through each constituent of the restricted pattern and invoke the generation process on the individual constituents, if necessary. This phase is described in the next section.

3.3 INTERPRETATION

The third major phase in PHRED is interpretation, the application of constraints to a restricted pattern to produce a surface structure suitable for output.

The process of interpreting a given constituent may have three possible results:

1) the successful completion of an element of surface structure,
2) the recursive application of the fetch-restrict-interpret sequence on the given constituent, or
3) failure, if the generator is unable to produce a specified pattern of speech.

The first result occurs when the pattern provides a complete specification of a word or words for output, such as *the big apple,* which is specified by the pattern

<word = the> <word = big> <word = apple>

The second case occurs if a constituent contains a more general set of constraints, for example,

<and p-o-s = verb root = remove tense = past>

which requires another recurrence of the fetch-restrict-interpret sequence.

In the third result, where no output produces the desired pattern of speech subject to the constraints given by the uninterpreted pattern, the system must back up to select an alternate pattern. To be efficient, the system must utilize as much as possible the patterns already selected. If the constituent that fails in the interpretation phase is optional to the pattern to which it belongs, it is deleted. Otherwise, failure results in backing up to the level where the failed pattern was fetched, getting another pattern from the stream, and attempting restriction of the new pattern. Most often this new pattern will be an ordering rule, and most of the failed pattern will be used in the restriction of the ordering pattern. A simple case of this is where the generator fails to produce a pattern of speech for the subject of a sentence and instead generates a passive sentence. In this case the restricted version of the PC pair as it was before the combination with the active ordering pattern is backed up on a stack so that the passive ordering can be tried.

Failure during interpretation is rare, and generally results from an insufficiency of the knowledge base in producing a reference. While a better model of the generation process might allow for the anticipation of such failures, such anticipation would in general require decisions considerably more complex than those made by PHRED. This complexity would be underutilized in light of the infrequency with which back-up is necessary. Although the back-up algorithm employed in these failures is time-consuming, it increases the likelihood that some successful utterance will be produced.

The agreement of constituents within a pattern is assured during the interpretation phase. A constituent that must agree with another has a form such as the following:

<and p-o-s = verb root = remove tense = past
number = (matches 1) person = (matches 1)>

This specifies a past tense form of *remove* that matches its subject in person and number. Interpretation results in the substitution of properties from the matched constituent to produce, for example,

<and p-o-s = verb root = remove tense = past
number = singular person = third>

In English there are only limited forms of agreement. There are few examples where it passes from right to left, such as in subject-aux inversion where the verb agrees with a subject that follows it. In other languages agreement within a pattern may be much more complex. In the Spanish example

Juan les habló a sus amigos

('John spoke to his friends'), the indirect pronoun *les,* which precedes the verb, agrees with the indirect object, which follows the verb.

In all cases PHRED can ensure proper agreement if some order of interpreting the constituents allows the correct application of constraints. The surface order of the constituents is the default order for their interpretation, but interpretation of a constituent where necessary is done only after that of constituents with which it must agree. In English, nouns within noun phrases are interpreted before their attached determiners, because the determiner must sometime agree in number with the head noun. In more inflected languages verbs must generally be produced last.

Anaphora are handled specially during interpretation. In the case of constituents for which PHRED has already produced references, the generator applies a set of heuristics that will remove the constituent entirely if it is not necessary to the utterance, pronominalize, or regenerate the entire constituent. The principal heuristics are

1) If the anaphoric constituent is optional, remove it from the current pattern, and
2) pronominalize other anaphoric constituents wherever possible.

There are of course many cases in which an alternative reference would be preferable, but the method used by PHRED is generally effective in producing coherent references. The heuristics lead, for example, to the production of *Mary was told by John that he wanted the book to be given to him* rather than *Mary was told by John that John wanted the book to be given to John by Mary.* It is apparent that these heuristics would break down in the generation of longer texts, a task for which neither PHRED nor the PHRAN/PHRED knowledge base was designed.

The interpretation mechanism occupies about 20% of the code of the generator, and requires a small amount of time relative to the rest of the program.

This discussion has described the overall design of PHRED and presented some details of its implementation. The next section traces an example of the generation process and discusses the role of each of the three phases considered here.

4 A Detailed Example

Below is a trace of PHRED while generating the sentence, *Typing "rm filename" causes the file filename to be removed from the current directory.* This is a fairly simple example, but demonstrates well the process used by PHRED to produce an output. At each step in the trace, the generator prints out which phase it is going through, and what the input to that phase is. Ellipses (...) are used to indicate information that has been omitted because it reduplicates other material. As earlier in the text, symbols preceded by a question mark indicate variables, such as "?actor". Symbols surrounded by asterisks, e.g., "*user*", are tokens that have special processing implications in the UNIX Consultant. Other special tokens are indicated by atoms followed by numerals, such as "file1".

The input to the generator is the concept which the UNIX consultant has chosen to express, in response to a question about removing files in UNIX. The concept represents UC's knowledge that using the 'rm' command is an established plan (here "planfor") for deleting a file (here "file1"):

```
***FETCHING***

concept =
   (planfor
      (result
         (state-change  (object file1)
                        (state-name location)
                        (from (inside-of (object current-directory)))
                        (to (not (concept (inside-of (object current-directory)))))))
      (method
         (mtrans     (actor *user*)
                     (object
                           (command (name rm)
                        (args (filename))))
                     (from *user*)
                     (to *UNIX*))))
```

There are a number of patterns that could potentially be used to express the concept that an action is a plan for something. Two of the possible constructs in the PHRED knowledge base are an imperative, e.g., *Use 'rm' to delete a file*, and a future or present tense declarative, e.g., *'Rm' will delete a file.* In this case, PHRED selects another pattern with the verb *cause*. The stream of candidate patterns includes first the constructs found in a bucket reached through the "planfor" concept, followed by other sentence-level PC pairs. In examples such as this one, where PHRED's fetching mechanism reaches several constructs through the same bucket, the generator selects a random order in the stream for the alternatives. For this example, therefore, a random selection ultimately determines the form of the output. After the selection is made, the restriction process is applied to the first pattern.

```
***RESTRICTING***

PATTERN:
   <p-o-s = act-phrase>
   <and  p-o-s = verb  root = cause>
   <and  p-o-s = inf-phrase  voice = passive>
CONCEPT:
   (planfor (result ?result) (method ?method))
PROPERTIES:
   method = (value 1)
   result = (value 3)
   p-o-s = sentence
   form = (declarative active)
   tense = (value 2)
```

The restriction process here results in the addition of the appropriate conceptual components to the constituents of the restricted pattern. The conceptual content of the first and third constituents, which will produce a gerund phrase and passive infinitive phrase, respectively, have been added. This results from the unification of the variables "method" and "result" in the list of properties above and the elaboration of the constituents specified by the terms "(value 1)" and "(value 3)" attached to these variables. Combination with an active sentence pattern adds the subject-verb agreement, and the restricted pattern enters the interpretation phase:

```
***INTERPRETING***

PATTERN:
   <and p-o-s = act-phrase
      concept =
                  (mtrans (actor *user*)
                     (object
                        (command (name rm) (args (filename))))
                     (from *user*)
                     (to *UNIX*))>
   <and  p-o-s = verb  root = cause
      person = (matches 1)  number = (matches 1)>
   <and p-o-s = inf-phrase
      voice = passive
      concept =  (state-change (object file1)
                     (state-name location)
                     (from (inside-of (object current-directory)))
                     (to (not (concept (inside-of (object current-directory))))))>
   PROPERTIES:
   tense = value 2
   form = (declarative active)
   p-o-s = sentence
   result = value 3
   method = value 1
   concept = ...
   method = ...
   result = ...
```

At this point the generator has successfully applied the input concept to restrict the surface structure chosen, and recursively interprets this structure, starting with the gerund phrase:

```
***INTERPRETING***

PATTERN:
   <and p-o-s = act-phrase
      concept =  (mtrans (actor *user*)
                     (object
                        (command (name rm) (args (filename))))
                     (from *user*)
                     (to *UNIX*))>
```

Interpreting a simple constituent results in a reinvocation of the fetch-restrict-interpret sequence on that constituent:

FETCHING

```
concept =  (mtrans (actor *user*)
                   (object
                      (command (name rm) (args (filename))))
                   (from *user*)
                   (to *UNIX*)))

p-o-s = act-phrase
```

Since there is no pattern that directly generates a gerund phrase (here "p-o-s = act-phrase") with the given concept, the fetch above yields an ordering pattern which can be used for combination with other patterns to produce the final phrase. Thus another fetch is performed before any restriction is done, this time without the "p-o-s" attribute.

FETCHING

```
concept = (mtrans ... )
```

PHRED searches for a way of expressing the "mtrans", or communicative transfer, of the 'rm' command to the operating system. The hashing mechanism gives preference to the terms for technical transmission of commands, because the concepts associated with these terms match the input concept more closely, but a problematic pattern still results:

RESTRICTING

PATTERN:

```
<person> <root do> <command>
```

CONCEPT:

```
(mtrans (actor ?actor)
   (object ?command)
   (from ?actor)
   (to *UNIX*))
```

PROPERTIES:

```
command = (command (name ?name) (args nil))
name = (value 3 command-name)
```

This pattern fails during unification because it requires that the command not have arguments, something which the fetching mechanism failed to detect because the bucket that includes the pattern is found by considering less specific attributes. This failure is illustrative of a class of examples where PHRED's hashing mechanism, in short-cutting the complexity of unification, picks the wrong pattern. With the gerund ordering pattern still being saved, the fetching mechanism is called again for another candidate. The pattern returned here by the fetching mechanism is the next one in the stream after the failed "do" pattern. This new pattern, with the verb *type,* is then passed through restriction:

RESTRICTING

PATTERN:

```
<person> <root = type> <command-spec>
```

CONCEPT:

```
(mtrans (actor ?actor)
   (object ?command)
   (from ?actor)
   (to *UNIX*))
```

PROPERTIES:

```
command = (value 3)
```

Unification of the variables in the above PC-pair with those in the input concept is followed by elaboration of the constituents and combination with the gerund ordering pattern. This yields the following result:

RESTRICTING

PATTERN:

```
<and  root = type  form = progressive>
<and  command-spec
   concept = (command (name rm) (args (filename)))>
```

PROPERTIES:

```
p-o-s = act-phrase
```

The combination of the "type" pattern with the gerund ordering satisfies the necessary constraints, producing a two-constituent pattern which then proceeds to the interpretation phase:

INTERPRETING

PATTERN:
<and p-o-s = verb root = type form = progressive>
<and command-spec
concept = (command (name rm) (args (filename)))>

PHRED recursively invokes the interpretation procedure on each of the two constituents, starting with the progressive verb:

INTERPRETING

PATTERN:
<and p-o-s = verb root = type form = progressive>

FETCHING

p-o-s = verb form = progressive root = type

This fetch uses a hash on the root and form of the verb given to retrieve the progressive form *typing*, whose properties unify trivially with the given constraints:

RESTRICTING

PATTERN:
<word = typing>

PROPERTIES:
form = progressive p-o-s = verb root = type

INTERPRETING

PATTERN:
<word = typing>

PROPERTIES:
root = type p-o-s = verb form = progressive

The word *typing* and its properties are now completely specified, so no further restriction is needed. The next constituent in the gerund phrase, the noun phrase that describes the command 'rm', is thus passed to the interpretation mechanism:

INTERPRETING

PATTERN:
<p-o-s = noun-phrase command-spec
concept = (command (name rm) (args (filename)))>

As usual, interpretation first results in a fetch:

FETCHING

concept = (command (name rm) (args (filename)))
p-o-s = noun-phrase

The pattern selected for the command is a specific formula for expressing commands to UNIX, the command name following by its arguments, in quotes:

RESTRICTING

PATTERN:
<word = |`| > <command> <p-o-s = args> <word = |'| >

PROPERTIES:
p-o-s = noun-phrase
concept = (command (name ?command) (args ?args))
args = (value 3 name)
command = (value 2 command-name))

The formula for producing 'rm filename' is straightforward, and results in very little additional work by the generator:

INTERPRETING

PATTERN:
<word = |`| >
<and command command-name = rm>
<and p-o-s = args name = filename>
<word = |'| >

PROPERTIES:
p-o-s = noun-phrase
...

INTERPRETING

PATTERN:
<word = |'|>

INTERPRETING

PATTERN:
<command command-name = rm>

FETCHING

command-name = rm

RESTRICTING

PATTERN:
<word = rm>

INTERPRETING

PATTERN:
<word = rm>

INTERPRETING

PATTERN:
<and p-o-s = args name = (filename)>

INTERPRETING

PATTERN:
<word = |'|>

Having completed the clause *Typing 'rm filename'*, the generator now returns to the highest level of the surface structure to finish the sentence. The next constituent in this surface structure is the conjugated form of the verb *cause:*

INTERPRETING

PATTERN:
<p-o-s = verb root = cause person = (matches 1) number = (matches 1)>

The interpretation mechanism finds the person and number of the first constituent of the surface structure. Since this is a singular gerund phrase, it has the third person and singular properties. These are then used in fetching the appropriate verb form:

FETCHING

p-o-s = verb root = cause form = basic person = third number = singular

---

As with *typing*, hashing results in the retrieval of the correct verb, and restriction is a simple process:

RESTRICTING

PATTERN:
<word = causes>

PROPERTIES:
tense = present root = cause person = third number = singular form = basic

INTERPRETING

PATTERN:
<word = causes>

PROPERTIES:
...

Having completed the specification of the verb *causes*, PHRED continues its depth-first interpretation with the third and final top-level constituent, the infinitive phrase:

```
***INTERPRETING***

PATTERN:
   <and  p-o-s = inf-phrase  voice = passive
     concept =
                  (state-change  (object file1)
                                 (state-name location)
                                 (from (inside-of (object current-directory)))
                                 (to (not (concept (inside-of (object current-directory))))))>

***FETCHING***

concept = (state-change ... )
p-o-s = inf-phrase
voice = passive

***FETCHING***

...
```

The first fetch in this case again brings the ordering pattern, the second the "remove" pattern. The restriction process is applied first to the "remove" pattern:

```
***RESTRICTING***

PATTERN:
   <person> <root = remove> <physob> <<word = from> <container>>

CONCEPT:
   (state-change (object ?rem-object))
      (state-name location)
      (from (inside-of (object ?container)))
      (to (not (concept (inside-of (object ?container))))))

PROPERTIES:
   rem-object = (value 3)
   ...
```

At this point, the generator is producing an expression for the passive infinitive phrase following the verb *causes*. After unification and elaboration of the pattern above, the pattern is then combined with the ordering pattern for the passive infinitive phrase, a somewhat more specialized pattern than is necessary for the construction of such phrases. The restriction process results in the determination of the final ordering of the constituents, and another round of restriction:

```
***RESTRICTING***

PATTERN:
   <and physob  concept = file1>
   <and  p-o-s = verb  root = be  form = infinitive>
   <and  p-o-s = verb  root = remove  form = perfective>
   <<word = from>
    <and container  concept = current-directory>>

PROPERTIES:
   subject = ?inf-phrase-subject
   voice = passive
   object = ?inf-phrase-object
   p-o-s = inf-phrase
   form = (passive)
```

Having completed the restriction of the infinitive, PHRED passes control to the interpretation mechanism, which then proceeds to generate each part of the infinitive phrase pattern:

```
***INTERPRETING***

PATTERN:
   <and  physob  concept = file1>
   <and  p-o-s = verb  root = be  form = infinitive>
   <and  p-o-s = verb  root = remove  form = perfective>
   <<word = from>
    <and container  concept = current-directory>>

PROPERTIES:
   p-o-s = inf-phrase
   concept = ...
   ...
```

As the interpretation starts with the first constituent of the infinitive phrase, PHRED now must produce a reference to the specified file. To do this, it expands the token "file1" to get the necessary information from its attributes.

INTERPRETING

PATTERN:
<and physob concept = (file (name (filename)))

PROPERTIES:
concept = ...
p-o-s = noun-phrase

FETCHING

p-o-s = noun-phrase
concept = ...

PHRED uses a structural formula, directly associated by a PC pair with the concept of a file, to refer to the hypothetical file:

RESTRICTING

PATTERN:
<word = the> <word = file> <name>

CONCEPT:
(file (name (filename)))

PROPERTIES:
ref = def
p-o-s = noun-phrase
person = third
number = singular
name = (value 3 name)

This pattern is the default reference for files, which is superseded when more information about a given file must be conveyed. The noun phrase now reaches the interpretation phase, resulting in the simple verification that its constituents are complete:

INTERPRETING

PATTERN:
<word = the> <word = file> <and p-o-s = args name = (filename)>

PROPERTIES:
concept = ... p-o-s = noun-phrase ref = def
...

Having completed the reference, the system now continues with the infinitive phrase. The second constituent of the infinitive phrase is the infinitive of the verb *be:*

INTERPRETING

PATTERN:
<and p-o-s = verb root = be form = infinitive>

As with the other verbs, fetching yields the appropriate form:

FETCHING

p-o-s = verb root = be form = infinitive

RESTRICTING

PATTERN:
<word = to> <word = be>

PROPERTIES:
p-o-s = verb root = be form = infinitive voice = active

INTERPRETING

PATTERN:
<word = to> <word = be>

PROPERTIES:
...

The third constituent of the passive infinitive phrase is the past participle of the verb *remove*, which is interpreted next. This process similarly results in the completed verb form:

INTERPRETING

PATTERN:
<p-o-s = verb root = remove form = perfective>

FETCHING

p-o-s = verb
root = remove
form = perfective
...

INTERPRETING

PATTERN:
<removed>

PROPERTIES:
...

The final constituent of the infinitive phrase and of the sentence is the optional prepositional phrase specifying from where the file is being deleted. The extra angle brackets in the pattern below indicate to the interpretation mechanism that if it fails to produce a reference or if the reference in the prepositional phrase is anaphoric, the entire constituent may be omitted:

INTERPRETING

PATTERN:
<<word = from> <and container concept = current-directory>>

The first constituent of the prepositional phrase, the word *from,* is already complete:

INTERPRETING

PATTERN:
<word = from>

The second constituent, the referent for the **current-directory,** is interpreted next:

INTERPRETING

PATTERN:
<and p-o-s = noun-phrase container concept = current-directory>

FETCHING

p-o-s = noun-phrase concept = current-directory ref = def

Unlike the previous noun phrase, there is no specific structural formula for referring to the current directory. PHRED thus uses a general noun phrase pattern:

RESTRICTING

PATTERN:
<and p-o-s = article consonance = (matches 2) number = (matches 2)>
<and p-o-s = noun number = singular>

PROPERTIES:
p-o-s = noun-phrase
person = third
number = singular
...
concept = (value 2)
number = (value 2 number)
person = (value 2 person)
...

"Consonance" here is the property used to handle the distinction between *a* and *an*, which depends on the choice of noun. "Hard" consonance is used for nouns or adjectives beginning with a consonant sound, and "soft" for those beginning with a vowel sound. For definite articles, the property is not used.

Elaboration of the pattern above results in a two-constituent pattern to be interpreted, the second constituent of which must refer to the current-directory concept.

INTERPRETING

PATTERN:
<and p-o-s = article consonance = (matches 2)
number = (matches 2) ref = def>
<and p-o-s = noun concept = current-directory
number = singular>

PROPERTIES:
Properties:

While there is no special noun phrase for referring to the current-directory con- cept, there are special noun constructs. PHRED selects randomly between two ways of referring to this concept, current directory and working directory.

INTERPRETING

PATTERN:
<and p-o-s = noun concept = current-directory number = singular>

FETCHING

p-o-s = noun number = singular person = third concept = current-directory

---

The reference selected for the directory is the compound noun current directory. This is interpreted before the article within the noun phrase, since articles are produced after head nouns to ensure agreement:

RESTRICTING

PATTERN:
<word = current> <word = directory>

PROPERTIES:
concept = ... consonance = hard person = third number = singular p-o-s = noun

INTERPRETING

PATTERN:
<word = current> <word = directory>

PROPERTIES:
Properties:

The interpretation mechanism judges the noun compound to be completed, and the final determiner is then interpreted:

INTERPRETING

PATTERN:
<and p-o-s = article ref = def number = singular consonance = hard>
...

INTERPRETING

PATTERN:
<word = the>

PROPERTIES:
...

After the final part of the surface structure is complete, a walk through the surface structure tree is used to produce the final output:

Typing 'rm filename' causes the file filename to be removed from the current directory.

---

5 Comparison with Other Research

PHRED differs in design from most other natural language generation systems because of its conception as a generator to accompany PHRAN as part of a language interface. The application of specialized phrasal knowledge seems to be an effective means of satisfying the demands on a generator in a domain such as that of the UNIX Consultant. The use of a declarative knowledge base shared between analyzer and generator has helped to make the system practical and easily extensible. PHRED's simplicity and the speed with which it applies this knowledge have made it well-suited for use in real-time natural language interfaces.

Primarily for historical reasons, most research in computational linguistics has focused on rules governing syntax. In language analysis, it is often practical to design systems whose principal function is to apply and

test such rules by determining the grammaticality of the input. Such systems generally use compositional rules, if any, for determining the semantic content of the input. The task of language generation, however, is inextricably tied to the *appropriateness* of the linguistic output as well as to its grammaticality. Because of this, work in generation focuses not on the representation of core syntactic rules but on the means by which a *choice* is made among syntactic and lexical constructs. Compositional rules generally fail to constrain this choice adequately. For this reason systems which are designed for language generation have often employed either special choice systems of the type found in systemic grammar (Halliday 1968), or have had pattern-based grammars of the type found in PHRAN/PHRED and in unification grammar (Kay 1984), which require a sophisticated mechanism for dealing with the interaction of the patterns. Thus PHRAN/PHRED is the first interface in a natural language-based artificial intelligence system to use an entirely common representation and knowledge base for linguistic knowledge employed in both analysis and production.

The declarative pattern-concept pair representation, its theory, and its role in PHRED, are considered in the discussion that follows.

5.1 THE PC PAIR

The pattern-concept pair representation differs on the surface from traditional grammars because the grammar is embedded implicitly in the knowledge structures. These knowledge structures often require the combination of a number of patterns to produce an utterance. In this way the representation is comparable to unification grammar, which contains patterns associated with functional descriptions. The restriction process described in this paper is similar to the unification procedure in TELEGRAM (Appelt 1983), which employs a unification grammar.

One difference between PHRED's knowledge structures and those in unification grammar is that conceptual attributes of the PC pairs, as well as functional attributes, or properties, are used to constrain a pattern. Unification grammar, like most feature systems, generally fosters the separation of conceptual and functional components. Another distinction is that, in unification grammar, the syntactic category is given special status; in pattern-concept pairs it is treated as an attribute, and does not necessarily have to be specified for every pattern. This is important for patterns that can be used in conjunction with many different orderings to produce a variety of syntactic structures.

A general difference between the PC pair and other representations lies in the level of specificity of the patterns. The PC pair makes it easy to encode specialized phrases and constructs to be used by the generator. It allows the generator to apply the same mechanisms to both general and specific constructions, and to choose PC pairs based on their conceptual attributes. This is, naturally, a distinction based on how the pattern-concept pairs are used rather than on their basic structure. The same result might well be achieved within the basic framework of lexical functional grammar or unification grammar.

Semantic grammar (Burton 1976) is another representation scheme which, like that of PHRED, facilitates the use of semantic attributes in language processing. There are versions of such grammars that allow for varying degrees of interaction between syntax, semantics, and pragmatics. PHRED differs from true semantic grammars primarily in that it facilitates the interaction of the more general patterns with the more specialized. Semantic grammars are often too constrained to be adapted to a new domain. Many of the knowledge structures in PHRED, by comparison, are general enough so that much of the linguistic knowledge used within the UNIX domain existed in the PHRAN/PHRED knowledge base before UC was even conceived.

The pattern-concept pair representation has developed in parallel with research on idiomatic and specialized use of language, done primarily by cognitive linguists. Similar ideas may be found in a variety of grammatical theories emphasizing the study of levels of linguistic and conceptual knowledge and the relations between them (cf. Lockwood 1972, Makkai 1972). The concept of units of meaning linked to lexical units is described, for example, by Pike (1962) and Lamb (1973).

Much of the work on specialized language questions the cognitive validity of traditional generative theories of grammar. Chafe (1968) identifies certain idioms, such as *by and large* and *all of a sudden,* which would be ungrammatical were they not given special status as idiomatic constructions. Other expressions, such as *kick the bucket,* are grammatical, but have a meaning that is not determined by any compositional relationship among their components. Chafe argues that these idiomatic constructs sufficiently pervade everyday language to warrant an approach to language that handles these constructs not as special cases or exceptions but as an integral part of a language.

Becker (1975) presents the idea of the phrasal lexicon as a means of handling canned and idiomatic phrases. Becker identifies in particular a range of phrases which are grammatical and even comprehensible via compositional rules, yet which suggest specialized contextual knowledge. The expression *It only hurts when I laugh* can theoretically be handled using traditional theories of grammar, but treating it as such would be ignoring an important component of the expression's meaning. The existence of such expressions, which involve either partially or entirely specialized knowledge, has generally been treated as of minor importance in computational theories of language. However, a cognitively realistic representation must take into account the role of both general syntactic knowledge and specialized knowledge about particular phrases.

While these arguments are directed at developing cognitively valid theories of linguistic representation, the handling of idiomatic constructs and of specialized phrasal knowledge has a substantial influence on the robustness and efficiency of a system. If specialized linguistic knowledge is indeed as pervasive as Chafe argues, a system that deals only with "core" grammatical and productive constructs will handle but a small portion of a language. A generator working within such a system would be severely limited in the range of utterances it could produce and in its ability to produce an output appropriate to a given context. On the other hand, failing to take advantage of linguistic generalizations can introduce redundancy and possibly inefficiency into the knowledge base. Robust and efficient language processing therefore demands a representation that takes advantage of both specialized idiomatic and general syntactic knowledge. Experience with the UNIX Consultant has suggested that the interaction of specialized and general linguistic knowledge is important for a natural language interface. This interaction is accomplished in PHRED by allowing the generator to combine ordering patterns with patterns used to relate linguistic constructs to their particular meanings.

Fillmore (1979) gives arguments for the idea of the **structural formula,** a phrase or construction that cannot be described strictly as the composition of its components but may still have a certain degree of structural freedom. Fillmore presents "<Time unit> in and <Time unit> out" as an example of such a formula, manifest in expressions such as *day in and day out* and *week in and week out.* More recently, Fillmore and others extend this idea to a theory of **grammatical constructions** (cf. Fillmore, Kay, and O'Connor 1984; Lakoff 1984), focusing on expressions that exhibit certain regularities and obey some grammatical constraints but whose behavior cannot be determined by "core" grammar. Examples of such expressions are *let alone* as in *He didn't make first lieutenant, let alone general,* and the deictic *there,* as in *There goes Harry, shootin.g his mouth off again* Fillmore, Kay, and O'Connor point out the difference between attempting to develop a minimal base of knowledge from which a linguistic competence can be *computed,* and attempting to develop a knowledge base that represents how human linguistic knowledge is in fact *stored.*

As an example of this distinction, consider the division drawn by Fillmore, Kay, and O'Connor between idioms of **decoding,** such as *kick the bucket,* and *spill the beans,* and idioms of **encoding** only, such as *answer the door,* and *wide awake.* All of these are grammatical idioms; that is, they have a syntactic structure and word order compatible with core grammatical constructs. The idioms of decoding, however, require specialized knowledge both for the comprehension of their meaning and their appropriate use. The idioms of encoding could possibly be comprehended using knowledge about their components only, but specialized knowledge is required to predict their use. Whether this specialized knowledge is to be stored in a given representational model therefore depends on what problem the model is addressing: competence, comprehension, or production. We have thus distinguished three potential classes of linguistic knowledge:

1) the knowledge required to determine the *membership* of a given phrase or sentence in a language,
2) the knowledge necessary to determine the *meaning* of a phrase, and
3) the knowledge that determines appropriate *use* of the phrase.

Computational linguistics has emphasized the first class, and thus many systems have attempted to define the second and third knowledge classes by adding auxiliary knowledge to a grammar for a linguistic competence. The PHRAN/PHRED pattern-concept pair representation, on the other hand, attempts to *subsume* the three classes into a single framework. Since the goal of PHRAN and PHRED is proficient analysis and use of language, the distinction between grammatical and extragrammatical idioms becomes of minor importance. It seems counterintuitive to treat phrases such as *all of a sudden* as of a different nature from *kick the bucket* simply because the former is extragrammatical. Further, the emphasis on the ability to compute a linguistic competence using a small set of rules is diminished. If specialized knowledge about a given phrase is required for its appropriate use, there is no reason why this knowledge cannot also be used for its syntactic analysis, even if, in a system that performs analysis alone, such knowledge would be redundant.

Consider the phrase *answer the door.* A pure syntactic analyzer would require no special knowledge to recognize the construct as a valid verb phrase. It is possible as well that the meaning of the phrase could be determined based on the structure of the verb phrase and its constituents. However, in order for PHRED to give the phrase its deserved distinction from *respond to the door* or other less appropriate utterances, special knowledge, that *answer the door* means to open a door in response to a knock or doorbell, is required. Since this knowledge is encoded into the common knowledge base, it may also be used by PHRAN to determine the meaning of the phrase.

The development of a knowledge base for the purposes of both language analysis and language production therefore changes the nature of the linguistic knowledge base and its use. Information that is redundant when considered from a formal linguistic standpoint may be important for a particular aspect of language processing. Such specialized knowledge may then be used by other components of the system. Thus the emphasis in the PHRAN/PHRED representation is on the *storage* of such redundant information rather than on its *computation.*

Specialized knowledge about phrases and constructions is an integral part of the knowledge base and is used preferentially to general knowledge which

requires more computation, both for analysis and production.

Of course, fundamental differences between analysis and generation still exist in PHRAN and PHRED. While the two programs have a shared knowledge base, they have entirely independent methods of accessing and applying their linguistic knowledge. PHRAN accesses patterns by recognizing sequences of constituents; PHRED must select a pattern based on the concept it is to express and the constraints which the pattern must satisfy. The PHRED approach to language generation is committed to the representation of linguistic knowledge in a declarative form which can be shared by the analyzer. The knowledge structures used by the generator are the same as those used by the analyzer, but the **process** that makes use of this knowledge to produce an utterance still reflects the basic choice problem.

The appropriateness of natural language output seems enhanced by the pattern-concept pair representation. Much of the knowledge used to produce language, particularly in specialized domains, is specialized knowledge. A natural language program that treats grammatical constructions and canned or idiomatic phrases independently of "core" grammar requires special rules and procedures to make use of such phrases. In PHRED specialized constructs are selected and produced using the same mechanism as the more productive constructs, facilitating the interaction of linguistic knowledge of varying levels of generality. In this way a wider range of appropriate utterances may be produced from a given conceptual form.

This discussion has focussed on the general representational aspects of PHRED. The next section concentrates on the details which relate specifically to other generation systems.

5.2. PHRED AND OTHER GENERATION SYSTEMS

PHRED differs from other generation systems primarily in the way it applies its knowledge to the generation task. Many language generation systems used in conjunction with large programs separate the linguistic knowledge base and lexicon from the conceptual knowledge base of the system (McDonald 1980, Mann and Matthiessen 1983, McKeown 1982). This has a variety of advantages, particularly the ability to develop and modify one module without affecting another. It also has the disadvantage of inhibiting the use of conceptual information by the generator, or of requiring redundant representation of such information, unless the modules are specifically designed to utilize common knowledge. In PHRED, linguistic knowledge, e.g., pattern concept pairs, is maintained separately from world knowledge, e.g., knowledge about the UNIX domain, to permit such advantages as the interchangeability of English and Spanish knowledge bases in UC. However, the generator may access the conceptual knowledge base of the system and such knowledge may interact with the syntactic knowledge. For example, the verbs *remove* and *delete* are synonymous when used to refer to actions on files, but *delete* may not generally be used with physical objects. PHRED restricts the use of *delete* during elaboration by examining the semantic nature of its object. If the object is not a file, the use of *delete* to refer to the action of removing it is prohibited.

Certain other complete natural language systems, like PHRAN/PHRED, exploit knowledge shared between analyzer and generator. The HAM-ANS question-answerer (Wahlster et al. 1983, Busemann 1984) makes use of a shared lexicon. The VIE-LANG system (Steinacker and Buchberger 1983) shares a "syntactico-semantic" lexicon, but the generator accesses this lexicon using a discrimination net with specialized choice knowledge.

A notable difference in implementation between PHRED and other generators is in the fetching mechanism. The division of the choice problem into an initial biasing and an evaluation component allows PHRED to bias its construction of utterances using a specialized hashing scheme. This has proven a boon for both simplicity and efficiency, as some of the rules which govern choice are carried out by a simple hashing process and thus fewer patterns reach the restriction phase. The basic choice mechanism as implemented in PHRED therefore encompasses two different phenomena, which may be viewed as **predisposition** and **selection.**

Predisposition is the process by which access to a knowledge base is influenced by various factors – such as the context, the concept to be expressed, or specific constraints on the desired output – to influence the order or priority in which elements of the knowledge base are considered. Selection is the evaluation of an element from the knowledge base. Intuitively, predisposition is the underlying access process that influences the likelihood of considering a particular word or phrase; selection is the judgement process which determines whether the word or phrase is appropriate. This resembles the notion of "register" in the systemic tradition (cf. Halliday 1978), but the biasing is not limited to situational influences.

There are three motivations for a design that provides for both a predisposition and a selection phase of the choice process. First, a system that employs as its principal choice mechanism, for example, a discrimination net such as Goldman's (Goldman 1975) or a unification scheme such as McKeown's (McKeown 1982) may apply its choice algorithm to many unlikely candidates, sometimes causing inefficiency. For example, the system might consider the verbs *smoke* and *inhale* every time it chooses the verb *breathe*. A fast indexing mechanism that quickly selects candidates trims the time spent evaluating inappropriate choices.

The second motivating force lies in the distinction between utterances that are technically correct in expressing a given concept and those that are generally appropriate to a given context. *John inhaled air* is techni-

cally correct but generally inappropriate in place of *John breathed.* This type of distinction can be embedded in a choice mechanism by attempting to axiomatize the rules that determine appropriateness, or it can be embedded in a predisposition mechanism which happens to order the choices according to the context. Predisposition thus provides a means for biasing choice without blurring the distinction between correctness and appropriateness.

The third motivation is cognitive validity. The predisposition-selection distinction fits the intuition that people have when they hear an unusual sentence: *It's okay but I wouldn't say it.* In the example of *breathe* and *inhale air* both utterances may fit the input conceptualization, but fluent speakers tend to choose the former. Fluent speakers also bias their predisposition mechanisms according to the nature and formality of the context. Pawley and Syder (1980) find that one of the differences between native and non-native speakers of a language is that non-native speakers take a long time to develop the predisposition component necessary for fluency. Chafe (1984) has pointed out some of the influential factors in the variations between spoken and written, or informal and formal, language. While some of this work is still in its early stages, the evidence strongly suggests a contextual biasing component distinct from the selection or evaluation phase of production.

The goal behind the PHRED indexing scheme is to incorporate as much of the choice problem as possible into the fetching, or predisposition, phase. Some language generators (Goldman 1975, McDonald 1980) use indexing tools that model choice as a multistage evaluation or decision-making process. The division of this process in PHRED into an "automatic" biasing component and a judgment component has some practical advantages. The hashing algorithm which drives the fetching mechanism orders the stream of patterns retrieved before any of them is actually evaluated, and thus the more time-consuming restriction process is spared having to apply heuristics to make certain choices. For example, a general heuristic used by a number of language generators can be expressed as "Choose the most specific pattern which matches the input constraints". In PHRED, this heuristic is realized by the hashing mechanism, which orders candidate patterns in terms of the number of buckets that yield them. In this way the sentence *John asked Bill to leave* is generally produced without considering the alternative *John informed Bill that he wanted him to leave.*

Appelt (1982) has presented language generation as the multi-level process of planning utterances to satisfy multiple goals. A division in this multi-stage process can be made between the task domain and the linguistic domain, i.e., between the system level and the interface level. PHRED operates at the interface level. User input to the UNIX consultant system is first analyzed by PHRAN, producing a conceptual knowledge structure which motivates the system's response (Wilensky, Arens, and Chin 1984). The planning component of the system exists entirely within the task domain of UC. Independent of the language being used, the UC planner makes the choice of illocutionary act, speech act, and the message to be conveyed. PHRED expresses the message in natural language.

While the ability to handle complex problems in language planning, such as the generation of references requiring knowledge about the hearer's knowledge, might be desirable even at the PHRED level, it is difficult to perform such planning within a real-time system. It is both counter-intuitive and inefficient to treat language production as primarily a reasoning process involving complex inference mechanisms. In fact, the need for such reasoning in language production seems rare. Thus the UC system draws a convenient, if arbitrary, division between the choices of responses and speech acts made by the UC planner and the lexical and structural choices made by PHRED.

Other systems such as Penman (Mann 1983), and TEXT (McKeown 1982) attack the problem of generating coherent multisentential text. This involves the influence of linguistic rules governing reference and focus on the process of deciding what to say. PHRED is not well equipped for this problem. While PHRED produces multisentential text when UC passes it successive concepts to express, it has no knowledge of coherence. Nor is there substantial communication between the PHRED level of production and the higher levels of language planning. Such communication, as described by Appelt (1982), would allow the generator to subsume multiple UC goals. In PHRED and UC much of the process of producing utterances is not considered as planning per se, but as the application of prestored knowledge about how language is used. The distinction between this prestored knowledge and general planning is analogous to the difference between compiled and interpreted code in programs. More research is required on how knowledge is compiled and on how the use of prestored knowledge about patterns of speech can be used in conjunction with general knowledge about planning.

This discussion has described some of the advantages of the PHRED approach to language generation, as well as some of the areas not really addressed in PHRED. The next section considers some of the promising ways in which the research described here can be extended.

6 Future Directions

PHRED is a successful implementation of a real-time generation system covering a range of linguistic phenomena, and has served also to open up new ground for further work. This work involves aspects of language processing not directly involved in PHRED as well as problems with the PHRED approach and implementation.

6.1 STRUCTURED ASSOCIATIONS

Much of the work on specialized language discussed earlier, as well as research on metaphor by Lakoff and others (Lakoff 1977, Lakoff and Johnson 1980) has suggested that there exist a range of underlying **motivations**[4] for many idioms and grammatical constructions, knowledge of which can help govern the use of language. For example, PHRED in its current form has the knowledge that the phrase *kick the bucket* does not passivize but *bury the hatchet* does, without any attempt to represent the motivation for the latter phrase. Knowing that *bury the hatchet* is motivated, i.e., that *bury* refers to terminating and *hatchet* to war, helps to explain the grammatical properties of the phrase. Ross (1981) has suggested that in many cases the variety of forms in which idioms of this type can appear depends on the ability of the noun component of the idiom to function independently as a noun. Passivization, however, seems subject to a more specific constraint; that is, the ability of the noun component of the idiom to *refer*.

To take advantage of this knowledge, a representation of the *bury the hatchet* idiom must encode the information not only that the expression refers to making peace, but that the *hatchet* part of the idiom refers to war or to the tools of war.

As another example where motivation might be useful, PHRED now generates *John took a punch from Mary* and *Mary gave John a punch* without representing the common metaphorical derivation of the two sentences. For example, PHRED might have a pattern

<person> <root = give> <person> <striking-action>

to produce the sentence *Ali gave Frazier a punch.* This is thus specialized knowledge about *giving* and a potential object. There might also be a pattern

<person> <root = take> <striking-action> <<word = from> <person>>

used to produce *Frazier took a punch from Ali.* Similar patterns might exist for *getting a punch* and *receiving a punch.* Treating these patterns independently seems cognitively unrealistic, because motivated phrases are in general easier to use and remember, and inefficient, since a more general representation of the *striking as transfer* metaphor might eliminate the need for some of the specialized knowledge about each of the patterns. While knowing the motivation does not obviate entirely the need for specialized knowledge, it can lead to a more parsimonious encoding of the specialized knowledge.

A potential improvement to the PHRAN/PHRED representation is the treatment of knowledge used to associate language and meaning as **structured associations.**[5] The structured association is an explicit relation between two knowledge structures that also associates their corresponding "components". These components may be **aspectuals,** or attributes, of the two structures or other arbitrarily related structures. A structured association may be used to relate the concept of a striking action to the concept of a transfer, with the patient of the action corresponding to the recipient of the transfer and the actor of the striking action corresponding to the source of the transfer. A structured association might also relate linguistic structures to associated concepts. The *bury the hatchet* expression may be related to a concept by a structured association, with the *hatchet* part corresponding to the *war* part of the concept and *bury* corresponding to the action of terminating the war. Metaphors and pattern-concept pairs alike may thus be represented as types of structured associations (cf. Jacobs 1985).

The structured association derives from the idea of a "view" (cf. Moore and Newell 1973, Bobrow and Winograd 1977, Wilensky 1984, Jacobs and Rau 1984), but is more general. The term **view** is used principally to describe relationships used to understand analogous concepts, while the structured association relates arbitrary knowledge structures. Also, the structured association is not a primitive relation, as structured associations themselves are a conceptual hierarchy.

Gentner's **structure-mapping** theory (Gentner 1983) addresses problems in understanding analogy that are comparable to some of the metaphorical issues discussed above. Gentner focuses on the process by which structure-mappings are synthesized rather than on the explicit representation of associations that may be used for such mappings.

Incorporating structured associations into a hierarchical knowledge base could further facilitate the interaction of general and specialized linguistic knowledge. Thus PHRED, and PHRAN as well, could gain efficiency in representation from the generalizations which apply without losing the advantages of having specialized patterns.

6.2 CONTEXT AND MEMORY MODELS

Another major area for future work is in the development of models of memory that help account for the role of context in language processing. A kind of spreading activation model (Arens 1982) was used in UC to help resolve references and to activate particular goals, plans and speech acts. The idea behind an activation-based model is that subtle changes in context can influence language processing without requiring the addition of large amounts of conceptual information to all of the linguistic knowledge structures.

A spreading-activation model has the potential of being especially useful in the predisposition, or fetching phase, of generation. Information about objects and events that have been explicitly referred to or activated in the current context, as well as about the topic of conversation and the participants in the conversation, can influence the language considered. There are, however, three major practical difficulties with using spreading activation as a means of controlling the effect of context on language production. First, the spreading activation

model is a parallel one which tends to produce slow, awkward simulations. Second, the encoding of knowledge into a network suitable for such a memory model must involve either a complex method of acquiring the knowledge from data or a contrived set of associative strengths based on introspection. Finally, while spreading activation is often effective in describing subconscious effects such as associative priming, it is difficult to account for the interaction of such effects with conscious or planned behavior. Most likely, a memory model will prove useful as a means of modeling the predisposition process and will simplify, but not replace, language planning and language selection.

7 Conclusion

PHRED is a practical language generator for use in natural language interfaces. The phrasal approach to language processing allows the generator to serve as an effective communicative tool within specialized domains without sacrificing the ability to adapt the system to new functions. The simple and efficient design of the program, particularly the process by which PHRED avoids expensive unification, allows it to serve as part of a real-time user interface. The use of a knowledge base shared with the PHRAN analyzer makes it easy to adapt the interface to a variety of domains in which understanding and production of fairly robust language is required.

In addition to its value as a useful language processing mechanism, PHRED has paved for the way for better models of language generation and linguistic representation. The PHRED approach supports a view of generation as a knowledge-intensive process in which the knowledge structures that relate language to meaning play a key role. The way in which these knowledge structures are accessed and applied emerges as the central issue in this model. The construction of robust, efficient and extensible natural language interfaces demands continued work at refining the means by which this "knowledge about language" is captured.

References

Appelt, D. 1982 Planning Natural Language Utterances to Satisfy Multiple Goals. SRI International AI Center Technical Note 259.

Appelt, D. 1983 Telegram: A Grammar Formalism for Language Planning. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics,* Cambridge, Massachusetts.

Arens, Y. 1982 The context model: language and understanding in context. In *Proceedings of the Fourth Annual Conference of the Cognitive Science Society,* Ann Arbor, Michigan.

Becker, J.D. 1975 The Phrasal Lexicon. In Schank, R. and Webber, B.L., Eds., *Theoretical Issues in Natural Language Processing.* (publisher?): Cambridge, Massachusetts.

Bobrow, D. and Winograd, T. 1977 An Overview of KRL, a Knowledge Representation Language. *Cognitive Science* 1(1).

Bresnan, J., Ed. 1983 *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge.

Burton, R. 1976 Semantic Grammar: an Engineering Technique for Constructing Natural Language Understanding Systems. Bolt Beranek and Newman Report No. 3453.

Busemann, S. 1984 Topicalization and Pronominalization. Extending a Natural Language Generation System. In *Proceedings of the Sixth European Conference on Artificial Intelligence,* Pisa, Italy.

Chafe, W.L. 1968 Idiomaticity as an Anomaly in the Chomskyan Paradigm. *Foundations of Language* 6(1).

Chafe, W. L. 1984 Integration and Involvement in Speaking, Writing, and Oral Literature. In Tannen, D., Ed., *Oral and Written Language.* Ablex, Norwood, New Jersey.

Danlos, L. 1984 Conceptual and Linguistic Decisions in Generation. In *Proceedings of the Tenth International Conference on Computational Linguistics,* Stanford, California.

Fillmore, C.J. 1968 The Case for Case. In Bach, E. and Harms, R., Eds., *Universals in Linguistic Theory.* Holt, Rinehart and Winston, New York.

Fillmore, C.J. 1979 Innocence: a Second Idealization for Linguistics. In *Proceedings of the Fifth Berkeley Linguistics Symposium,* Berkeley, California.

Fillmore, C.J.; Kay, P.; and O' Connor, M.C. 1984 Regularity and Idiomaticity in Grammar: The Case of Let Alone. University of California, Cognitive Science Working Paper.

Gentner, D. 1983 Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science* 7:155-170.

Goldman, N. 1975 Conceptual Generation. In Schank, R.C., Ed., *Conceptual Information Processing.* American Elsevier Publishing Company, Inc., New York.

Halliday, M.A.K. 1968 Notes on Transitivity and Theme in English. *Journal of Linguistics* 4.

Halliday, M.A.K. 1978 *Language as Social Semiotic.* University Park Press, Baltimore.

Harris, Z. 1968 *Mathematical Structures of Language.* John Wiley and Sons, New York.

Hudson, R. 1976 *Arguments for a Non-Transformational Grammar.* University of Chicago Press, Chicago.

Jacobs, P. 1983 Generation in a Natural Language Interface. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, Germany.

Jacobs, P. and Rau, L. 1984 Ace: Associating Language with Meaning. In *Proceedings of the Sixth European Conference on Artificial Intelligence,* Pisa, Italy.

Jacobs, P. 1985 A Knowledge-Based Approach to Language Production. University of California at Berkeley, Computer Science Division Report #UCB/CSD 86/254.

Kay, M. 1979 Functional Grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistic Society.*

Kay, M. 1984 Functional Unification Grammar: A formalism for Machine Translation. In *Proceedings of the Tenth International Conference on Computational Linguistics,* Stanford, California.

Kempen, G. and Hoenkamp, E. 1982 An Incremental Procedural Grammar for Sentence Formulation. University of Nijmegen (the Netherlands) Department of Psychology, Internal Report 82-FU-14.

Kittredge, R. and Lehrberger, J. 1983 *Sublanguages: Studies of Language in Restricted Domains.* Walter DeGruyter, New York.

Lakoff, G. 1977 Linguistic Gestalts. In *Proceedings of the Thirteenth Regional Meeting of the Chicago Linguistics Society.*

Lakoff, G. and Johnson, D. 1980 *Metaphors We Live By.* University of Chicago Press, Chicago.

Lakoff, G. 1984 There-constructions: A Case Study in Grammatical Construction Theory. University of California, Linguistics Working Paper.

Lamb, S. The Crooked Path of Progress in Cognitive Linguistics. In Makkai, A. and Lockwood, D., Eds., *Readings in Stratificational Linguistics.* University of Alabama Press, University, Alabama.

Lockwood, D. 1972 *Introduction to Stratificational Linguistics.* Harcourt Brace and Jovanovich, New York.

Makkai, A. 1972 *Idiom Structure in English.* Mouton, The Hague.

Makkai, A., Ed. 1975 *A Dictionary of American Idioms.* Barron's Educational Series, New York.

Mann, W. 1983 An Overview of the Penman Text Generation System. In *Proceedings of the National Conference on Artificial Intelligence,* Washington, D. C.

Mann, W. and Matthiessen, C. 1983 Nigel: A Systemic Grammar for Text Generation. University of Southern California, ISI Technical Report #ISI/RR-83-105.

McDonald, D.D. 1980 Language Production as a Process of Decision-making Under Constraints. Ph.D. dissertation, MIT.

McKeown, K. 1982 Generating Natural Language Text in Response to Questions about Database Structure. Ph.D. thesis, University of Pennsylvania.

Moore, J. and Newell, A. 1974 How can MERLIN Understand? In Gregg, L., Ed., *Knowledge and Cognition.* Erlbaum Associates, Inc.

Pawley, A. and Syder, F.H. 1980 Two Puzzles for Linguistic Theory: Nativelike Selection and Nativelike Fluency. Unpublished manuscript.

Pike, K. 1962 Dimensions of Grammatical Constructions. In Brand, R., Ed., *Kenneth L. Pike: Selected Writings.* Mouton, The Hague.

Riesbeck, C. 1975 Conceptual Analysis. In Schank, R.C., Ed., *Conceptual Information Processing.* American Elsevier Publishing Company, Inc., New York.

Rosch, E. 1977 Human Categorization. In Warren, N., Ed., *Studies in Cross-Cultural Psychology (Vol. I).* London, Academic Press.

Ross, John Robert 1973 Nouniness. In Fujimura, Osamu, Ed., *Three Dimensions of Linguistic Theory.* TECCorporation, Tokyo.

Ross, John Robert 1981 Nominal Decay. Unpublished manuscript.

Schank, R.C., Ed. 1975 *Conceptual Information Processing.* American Elsevier Publishing Company, Inc., New York.

Steinacker, I. and Buchberger, E. 1983 Relating Syntax and Semantics: The Syntactico-semantic Lexicon of the System VIE-LANG. In *Proceedings of the First European Meeting of the ACL,* Pisa, Italy.

Wahlster, W.; Marburger, H.; Jameson, A.; and Busemann, S. 1983 Overanswering Yes-No Questions: Extended Responses in a Natural Language Interface to a Vision System. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, W. Germany.

Wilensky, R. and Arens, Y. 1980 PHRAN – A Knowledge-Based Approach to Natural Language Analysis. University of California at Berkeley, Electronics Research Laboratory Memorandum #UCB/ERL M80/34.

Wilensky, R. 1981 A Knowledge-Based Approach to Natural Language Processing: A Progress Report. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* Vancouver, British Columbia.

Wilensky, R. 1984 KODIAK – A Knowledge Representation Language. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society,* Boulder, Colorado.

Wilensky, R.; Arens, Y.; and Chin, D. 1984 Talking to UNIX in English: An Overview of UC. *Communications of the Association for Computing Machinery* 27(6).

Notes

1. This research was sponsored in part by the Office of Naval Research under contract 00014-80-C-0732, the National Science Foundation under grants IST-8007045 and IST-8208602, and the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3041, Monitored by the Naval Electronic Systems Command under contract N00039-82-C-0235.

   I am grateful to Robert Wilensky for his guidance and for his important comments on numerous drafts of this paper, and to Lisa Rau for many helpful suggestions.
2. Author's current address: Knowledge-Based Systems Branch, General Electric Corporate Research and Development, Schenectady, NY 12301.
3. UNIX is a trademark of AT&T Bell Laboratories.
4. The term motivation, as employed here, is due to Charles Fillmore and George Lakoff, personal communication.
5. The term structured association and the use of structured associations in language processing were suggested by Robert Wilensky, personal communication.