MS-CIS-82-05

GENERATING NATURAL LANGUAGE TEXT IN RESPONSE TO QUESTIONS ABOUT DATABASE STRUCTURE

Kathleen Rose McKeown

A DISSERTATION

in

Computer and Information Science

Presented to the Graduate Faculties of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

1982

_____

Supervisor of Dissertation

_____

Graduate Group Chairperson

ABSTRACT

GENERATING NATURAL LANGUAGE TEXT
IN RESPONSE
TO QUESTIONS ABOUT DATABASE STRUCTURE

Kathleen Rose McKeown
Supervisor: Aravind K. Joshi

There are two major aspects of computer-based text generation: 1) determining the content and textual shape of what is to be said; and 2) transforming that message into natural language. Emphasis in this research has been on a computational solution to the questions of what to say and how to organize it effectively. A generation method was developed and implemented in a system called TEXT that uses principles of discourse structure, discourse coherency, and relevancy criterion.

The main features of the generation method developed for the TEXT strategic component include 1) selection of relevant information for the answer, 2) the pairing of rhetorical techniques for communication (such as analogy) with discourse purposes (for example, providing definitions) and 3) a focusing mechanism. Rhetorical techniques, which encode aspects of discourse structure, are used to guide the selection of propositions from a relevant knowledge pool. The focusing mechanism aids in the organization of the message by constraining the selection of information to be talked about next to that which ties in with the previous discourse in an appropriate way.

This work on generation has been done within the framework oi natural language interface to a database system. The implemei system generates responses of paragraph length to questions ai database structure. Three classes of questions have been considei questions about information available in the database, requests definitions, and questions about the differences between datal entities.

The main theoretical results of this research have been on effect of discourse structure and focus constraints on the general process. A computational treatment of rhetorical devices has 1 developed which is used to guide the generation process. Previous *x* on focus of attention has been extended for the task of generation provide constraints on what to say next. The use of these interacting mechanisms constitutes a departure from earlier general systems. The approach taken in this research is that the general process should not simply trace the knowledge representation to pro< text. Instead, communicative strategies people are familiar with used to effectively convey information. This means that the *i* information may be described in different ways on different occasioi

ACKNOWLEDGEMENTS

I am deeply indebted to my advisor, Aravind Joshi, for his insightful comments, unending patience, and helpful guidance throughout all stages of this dissertation. This work would not have been possible without his help.

Bonnie Webber also deserves special thanks for her many helpful suggestions, the pointers to relevant papers, and especially, her editorial comments. Thanks also goes to Barbara Grosz for her comments during the writing stages of the dissertation and to the members of the committee for their comments and helpful questions: Norman Badler, Peter Buneman, Ellen Prince, and again, Bonnie Webber.

I would like to thank Kathy McCoy and Steve Bossie for their help in implementing and designing portions of the system: Kathy for her work on the knowledge representation and Steve for his work on the tactical component. I am also indebted to Kathy for her most thorough reading of the dissertation and her role as friend, confident, and sounding board while my office mate. Eric Mays' and Steve Bossie's comments on various drafts of this thesis are also appreciated. Thanks goes to the many other students in the Penn Computer Science department who helped to make my stay here a good one, among them Sitaram Lanka, Steve Platt, Dahlia Benaroya, and the members of the North Corridor.

Finally, thanks goes to my friends outside the department, particularly Missy Staples, Gerry Gendimenico, Joe O'Rourke, and Marylynn Salmon. To my parents, for their understanding and support. And especially, to my fiance, David Cooper, for his patience, encouragement, and understanding throughout all stages of the dissertation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION

In the process of producing discourse, speakers and writers must decide what it is that they want to say and how to present it effectively. They are capable of disregarding information in their large body of knowledge about the world in general which is not specific to the task at hand and they manage to integrate pertinent information into a coherent unit. They determine how to start the discourse, how to order its elements, and how to appropriately close it. These decisions are all part of the process of deciding what to say. The speaker and writer must also determine what words to use and how to group them into sentences.

While researchers in computational linguistics have concentrated on local issues concerning the syntactic and lexical choices involved in transforming a pre-determined message into natural language, problems involving the content and textual shape of the message have been largely ignored. This research emphasizes a computational solution to the problems of deciding what to say and how to organize it effectively. The main contributions of this research, therefore, have been the development and application of principles of discourse structure, discourse coherency, and relevancy criterion to the computer generation of text.

1.1 A Processing Model

The approach taken in this research relies on a model of language production which divides the processing into two stages. The first determines the content and structure of discourse and is termed the "strategic" component, following Thompson [THOMPSON 77]. The second, the "tactical" component, uses a grammar to translate the message into English. The output of the strategic component is an ordered message; all decisions about what to include in the text and when to include it have been made.* The strategic component, furthermore, must be capable of providing information needed by the tactical component to make decisions about lexical and syntactic choice.

1.2 The Strategic Component

The strategic component embodies both semantic and structural processes. Semantic processes are necessary for determining relevancy: of all that could be said, the component must be capable of selecting that information that is relevant to the current discourse goal. The strategic component must also be capable of determining what rhetorical strategies are appropriate in the given discourse situation. Communicative techniques must be selected and integrated to form the

*Although processing in this research was based on a division of the two stages such that the results of the strategic component were completely determined and then passed to the tactical component, the tenets of this research would not be affected by a control structure which allowed for backtracking between the tactical and strategic component such as Appelt suggests [APPELT 81]. See Chapter 6 for further discussion of this issue.

text. In this process, semantics is also important since the text generated must be a coherent unit.

Although some of the decisions that must be made are basically semantic in nature, while others are structural, the mechanisms that handle these decisions need not be wholly semantic or structural. In fact, the claim is made here that each of these decisions is determined by an interaction between structural and semantic processes. The rhetorical strategies used in the text will affect its content and the information that is determined to be relevant will influence the organization of the text.

1.3 Generation Of Text

Issues of discourse structure and discourse coherency are important since this research concerns the generation of text and not simply the generation of single sentences. Since emphasis in this work is on problems specific to the generation of multi-sentential strings, a study of discourse structure and its relation to the process of generating natural language was required.

Generation of text differs from generation of single sentences within dialogue in that a text is more or less a linguistically complete structure. A textual unit in and of itself constitutes a description or explanation that has a meaningful interpretation. This is in contrast to a dialogue sentence which may only be comprehensible in the context of the preceding discourse.

Considerations of context are also important for the generation of text however• Generation of a single sentence within a text must take into account the preceding and succeeding text. Even if the overall organization of the text provides an appropriate framework for the single sentence, it must nonetheless be semantically linked in some way to the preceding and succeeding sentences if the resulting text is to be coherent• If the text is generated within an interactive environment, the preceding discourse may also affect its generation.

1#* Written Versus Spoken Text

An assumption was made that the generation of text done in this research would more closely resemble written than spoken text. This was done partially to avoid accounting for some of the phenomena which normally occur in speech, such as self-correction, incomplete or ungrammatical sentences, informal styles or phrases (e.g. "yeah ...lf, "well11), interruption, and circularity. It also means that an investigation of the process of planning text is important, since writers typically spend more time planning the organization and content of what is to be said than do speakers. For practical reasons, the use of written text is more appropriate since reading transcribed spoken text would not be easy for the given application (see Section 1.6).

1 -> Generation Versus Understanding

While a great deal of time has been devoted to the problems involved in computer interpretation of natural language, interest in the problems of generating it is just beginning to gain momentum (see Chapter 6 for a discussion of research in generation). Although some investigation has been done on the development of grammars that can be used for both parsing and generation (e.g. [KAY 79]), there are some important distinctions that should be made about the processes required for each task.

Understanding natural language requires examination of the evidence provided by a particular text in order to determine the meaning and intentions of the speaker who produced it. Evidence may consist of the words selected by the speaker which provide clues as to the content or the phrase and sentence structures he uses which not only transmit meaning but also set up context and indicate what the speaker is focusing on. Interpretation of language necessitates determining from among a limited set of options known to be available to the speaker, that option that the speaker took.

While interpretation involves specification of how a speaker's options are limited at any given point (for example, by writing grammars, by establishing constraints on focus of attention, etc.), it does not require a formulation of reasons for selecting between those options. In generation of natural language, however, this is exactly what is required. A generator must be able to examine all

possibilities for expression and decide which is the best for the given situation. Where research on interpretation may describe limitations on options in order to more efficiently determine the option taken, research in generation must specify the reasons for selecting one option over another in varying situations. This includes reasons for such high-level decisions as whether to include one piece of information before another as well as such low-level decisions as when the passive is more appropriate than the active construction. This research, therefore, differs from research on interpretation in that it must examine decision mechanisms for selecting between options.

1.6 Application

In order to test principles about natural language generation, an application was selected that could provide a motivation for speaking and a manageable domain. A system was developed, therefore, within the framework of a natural language interface to a database system that addressed the specific problem of generating answers to questions about database structure.

To date, natural language database systems have concentrated on answering factual questions, providing answers in the form of lists or tables of objects in the database.* These questions query the existence or identity of restricted classes of objects in the database. To answer such questions, the database is searched for objects which meet the given restrictions.

To ask such questions, the user must already know what information is stored in the database and how it is structured. (Note that even if the user already knows what type of information is available, its structure in the database does not always correspond to some "natural" conception.) If the user is not aware of the nature of information stored and its structure, he can neither request the system to supply this information (since systems don't possess this capability) nor properly phrase questions about the database contents.

The TEXT system was developed to generate responses to such meta-level questions. Three classes of questions have been considered: questions about information available in the database, requests for definitions, and questions about the differences between database entities. In this context, input questions provide the initial motivation for speaking.

Although the specific application of answering questions about database structure was used primarily for testing principles about text generation, it is a feature that many users would like. Several experiments ([MALHOTRA 75], [TENNANT 79]) have shown that users often ask questions to familiarize themselves with the database structure before proceeding to make requests about the database contents.

---

*Note that in some systems, the list (especially in cases where it consists of only one object) may be embedded in a sentence, or a table may be introduced by a sentence which has been generated by the system [GRISHMAN 79]. In a few systems (e.g. [MALHOTRA 75], [CODD 78]), a one or two sentence reply about the information in the database may be generated, but this reply is usually stored as a whole in the knowledge structure.

Malhotra's experiment involved a simulated management decision support system in which users typed in their questions at a terminal. These questions were intercepted by a person familiar with the system, who rephrased the questions using syntax acceptable to the system. When questions were asked which the system could not answer, the interceptor would either answer the question himself or construct a series of questions necessary to answer the one asked. Subjects were given a problem to solve which required using information stored in the database. Transcripts of the user sessions indicate that people often begin by asking questions to familiarize themselves with the material available before asking questions particular to the given problem. Typical of the questions asked are the following:

> What kind of data do you have?
> What do you know about unit cost?
> What is the difference between material cost and production cost?
> What is production cost?

Tennant's experiments were done on two natural language database systems: the PLANES system, which accesses a large database containing information about Naval aircraft, and Automatic Advisor, which accesses a smaller database containing course information. University students were asked to solve database problems after reading introductory information about the database. Tennant found that systems tended to be lacking in conceptual coverage. Like Malhotra, he found that users often asked questions which were not interpretable as database queries. These included questions about the database (e.g. "What do you know?[11])

and questions about vocabulary (e.g. "What is a buser?").

Responding to questions such as these requires more than a simp search of the database. These types of questions do not provide cle restrictions on what information is sufficient to answer them as specific questions about the database content. In fact, it is uncle that there is a single correct way to answer them. Since answers such questions about the structure of the database will usually requi more than a single sentence, the application provides an appropria testbed for generation principles. The system will be required determine how to select the appropriate information to be included the answer and how to organize it into a multi-sentential text.

Implementation of the TEXT system for natural language generati used a portion of the Office of Naval Research (ONR) database. T portion used contains information about vehicles and destructi devices. Some examples of questions that can be asked of the syst include:

> What is a frigate?
> What do you know about submarines?
> What is the difference between a whisky and a kitty hawk?

Examples of questions from this domain will be used throughout t thesis.

The kind of generation of which the system is capable is illustrated by the response it generates to question (A) below. Other responses will be shown throughout the dissertation.

A) What kind of data do you have?

> All entities in the ONR database have DB attributes REMARKS. There are 2 types of entities in the ONR database: destructive devices and vehicles. The vehicle has DB attributes that provide information on SPEED-INDICES and TRAVEL-MEANS. The destructive device has DB attributes that provide information on LETHAL-INDICES.

The type of response generated by the TEXT system could be used not only for specific questions about the database structure, but also as supportive explanations for yes/no questions* or as explanations for structural presumption failures [MAYS 80].** As an example, consider the question "What is the draft and displacement of the whisky?". A plausible response is given in (B) below. This is very similar to some of the responses currently generated by the TEXT system.

B) The database contains no information on DRAFT and DISPLACEMENT for the whisky. Ships have DB attributes DRAFT

---

*Kaplan [KAPLAN 79] also discusses the use of supportive explanations for yes/no questions. Kaplan's system, however, will only supply explanations when an extensional failure of a presumption of the question occurs. Explanations would be supplied here when a negative response occurred as a result of the database structure.

**The system developed is not capable of detecting an intensional failure. Assuming that such a failure has been found, the system could be extended to generate a response that explains the failure.

and DISPLACEMENT. The whisky is an underwater submarine with a PROPULSION-TYPE of DIESEL and a FLAG of RDOR. The submarine's underwater capabilities are provided by the DB attributes under DEPTH (for example, OPERATING_DEPTH) and MAXIMUM_SUBMERGED_SPEED. Other DB attributes of the submarine include OFFICIAL_NAME, FUEL (FUEL_TYPE and FUEL_CAPACITY), and PROPULSION_TYPE.

A system for generating textual responses to questions requirin descriptions or explanations could be useful in other application area in addition to the database query system. Computer assiste instruction systems [COLLINS 74] and expert systems [GROSZ 77] ar examples of areas where the provision of descriptions and explanation would be useful. The methods for generation developed for the TEX system are not specific to the database application and could b adapted for systems where generation of descriptions of stati information is required.

The database query system was chosen as an application because need for a facility that could provide information about databas structure had been demonstrated and thus, the application is practical one. Because of the nature of information represented in th knowledge base (entities and relations between entities), the answer that can be given are to a certain extent limited, making the proble more manageable. Because of the types of questions considered however, the generation required is complex enough to thoroughly tes the principles developed in this research.

1.7 Methods And Main Contributions

The main features of the generation method developed for the TEXT strategic component include 1) selection of relevant information for the answer, 2) the pairing of rhetorical techniques for communication (such as analogy) with discourse purposes (for example, providing definitions) and 3) a focusing mechanism. Rhetorical techniques, which encode aspects of discourse structure, are used to guide the selection of propositions from a relevant knowledge pool: a subset of the knowledge base which serves as the source for all information which can be included in the text. The focusing mechanism helps maintain discourse coherency. It aids in the organization of the message by constraining the selection of information to be talked about next to that which ties in with the previous discourse in an appropriate way. These processes, which operate in a cooperative fashion to produce the textual message, are described in more detail after setting out the framework of the system.

The relevant knowledge pool is constructed by semantic processes after receiving the input question. It contains information determined by the system to be relevant to the given question. Use of a relevant knowledge pool provides a limit on the information that needs to be considered when constructing an answer to a given question, thus increasing the efficiency of the program while at the same time providing a model of a speaker's narrowing of attention when answering a question.

Rhetorical techniques are the means which a speaker has available for description. In the TEXT system, these techniques have been encoded as schemas which represent patterns of discourse structure. Use of schemas reflects the fact that people have preconceived ideas about how to provide different kinds of descriptions. The choice of a particular schema to use for an answer is affected by a characterization of the information available and by the discourse purpose of the current answer. The schema is effectively a plan for the text and is used to guide the generation process.

Focusing constraints are used to ensure that the generated text is coherent. Since text is about something, what is said at any given point must be appropriately related to what has already been said. The focusing mechanism tracks focus of attention as the text is created and eliminates options for what to say next that violate its knowledge about valid shifts in focusing. The focus constraints monitor the use of the schemas in the TEXT system.

The main theoretical emphases of this dissertation have been on the effect of discourse structure and focus constraints on the generation process. This has involved a formulation of discourse structure that is commonly used in naturally occurring texts as well as an analysis of the ways in which focus of attention can and does shift throughout a text. This work presents a computational model of rhetorical devices that can be used for generation, an approach that has not previously been taken. It also illustrates how focus of attention can be used for the generation process through the

development of an ordering on focus constraints used for interpretation of discourse.

These formulations have been embodied in the semantic and structural processes of the strategic component. That these processes interact with each other results in a greater variety of possible texts. A single plan for generation used in different situations doesn't always produce the same text because of the focus constraints. Similarly, although the same information may be produced by semantic processes for answering two different questions, the answers generated may be different since schemas are associated with the discourse purpose of the answer.

1.8 System Overview

To answer an incoming question about database structure, TEXT first selects a set of possible schemas to be used for the answer. The schemas encode those rhetorical techniques associated with the discourse purpose of the current answer (for example, providing definitions). On the basis of the input question, semantic processes produce a pool of relevant knowledge. For questions requiring a comparison, this involves an assessment of the conceptual closeness of the two items. The type of information available in this pool is used to select a single schema from the set of possible schemas. This marks the beginning of interaction between the structural and semantic processes in the system; here semantics influences the structure selected for the answer.

The answer is constructed by "filling" the schema: propositions are selected from the relevant knowledge pool which match the rhetorical techniques in the schema. Each rhetorical technique has associated semantics that indicate which types of propositions in the knowledge base it matches. (Note that semantics associated with the rhetorical techniques are particular for the database query domain, but are not dependent upon the database domain.) A focusing mechanism monitors the matching process; where there are choices for what to say next (i.e. - where the rhetorical technique matches several propositions in the knowledge pool), the focusing mechanism selects that proposition which ties in most closely with the previous discourse. When the proposition has been selected, focus information about the proposition is recorded.

When the schema has been filled, the system passes the constructed, ordered message to the tactical component. The tactical component uses a functional grammar, based on a formalism defined by Kay [KAY 79], to translate the message into English. The grammar was designed so that it can use the focus information provided in the message to select an appropriate syntactic construction. A simple overview of the text generation process is shown in Figure 1.1.

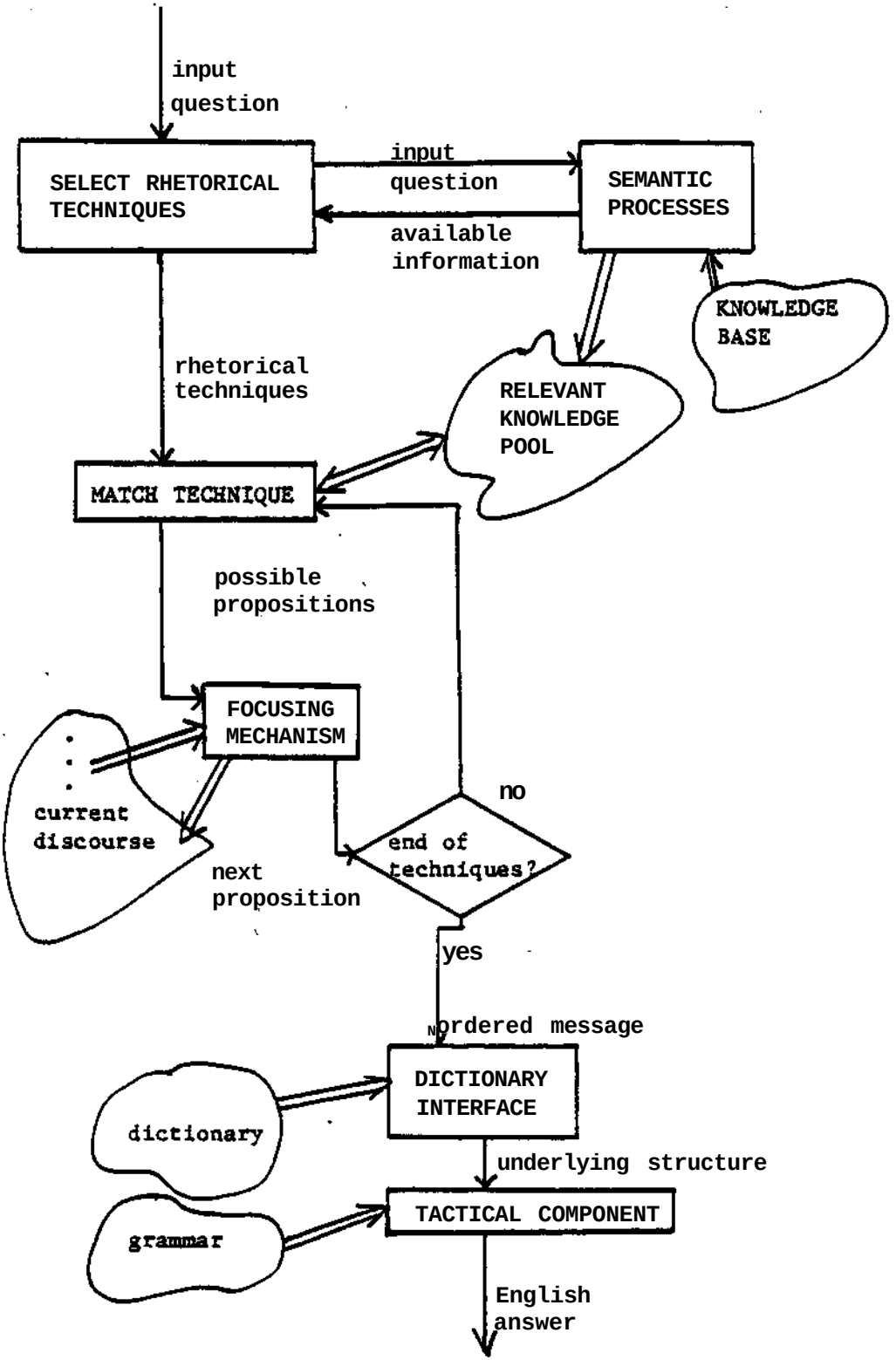


FIGURE 1.1 System Overview

1.9 Other Issues And Limitations

In order to develop a system that can generate text in response to questions about database structure, problem areas outside the realm of text generation per se had to be considered. These include the knowledge representation, which contains the information to be described, interpretation of the user's question, user modelling, and the tactical component for producing actual English (although a generation issue, it was not the major emphasis of the research done here).

A knowledge representation was implemented which draws heavily on features used in other database models. It is based on the Chen entity-relationship model [CHEN 76] and also includes a generalization hierarchy on entities, a hierarchy on attributes, and distinguishing characteristics of entities in the generalization hierarchy. The representation and accessing functions were implemented by McCoy [MCCOY 82].

A tactical component was also implemented in order to illustrate that the methods used for planning text are successful. The major modules of this component were designed and partially implemented by Bossie [BOSSIE 82].

No facility for interpreting a user's questions is provided in the TEXT system implementation since this work is on the generation of language and not interpretation. Questions must be phrased using a simple functional notation which corresponds to the types of questions

that can be asked. It is assumed that a component could be built to perform this type of task and that the decisions it must make would not affect the performance of the generation system. The TEXT system provides a canned explanation of this notation when it is invoked.

An extensive user model was not implemented as part of the TEXT system. The system assumes a static casual and naive user and gears its reponses to a level appropriate for this characterization. Although not implemented, some analysis was done on the effect of the previous discourse on the generation of responses.

This work, delimited as it is, represents an important contribution to the field of natural language generation. By limiting the scope of the project, this research could focus on issues concerning the content and organization of the generated text. These two problems have not, for the most part, been addressed in the past and they represent areas about which little is known. In order to handle them appropriately, a comprehensive treatment of discourse structure and focusing constraints and their relation to the generation of natural language was necessary.

1.10 Guide To Remaining Chapters

A discussion of discourse structure, its effect on generation, and the implementation of the schemas is provided in Chapter 2. The focus constraints, both as they affect discourse coherency and as they restrict attention to relevant information, are discussed in Chapter 3.

The implementation of the TEXT system is described in Chapter 4, although implementation that was thoroughly discussed in Chapters 2 and 3 is not reiterated. The knowledge base used, the method used to determine relevancy, the dictionary and the tactical component are all described in Chapter 4. It closes with a discussion of practical considerations* Chapter 5 gives an analysis of how the previous discourse could be used to improve the quality of the responses generated. A comparison of this work to other research in natural language generation is provided in Chapter 6 and the final chapter presents some conclusions. Appendix B provides examples of the TEXT system in operation.

2.0 DISCOURSE STRUCTURE

The approach towards text generation adopted in this dissertation is based on two fundamental hypotheses about the production of text: 1) that how information is stored in memory and how a person describes that information need not be the same and 2) that people have preconceived notions about the ways in which descriptions can be achieved.

I assume that information is not described in exactly the same way it is organized in memory. Rather, such descriptions reflect one or more principles of text organization.* It is not uncommon for a person to repeat himself and talk about the same thing on different occasions. Rarely, however, will he repeat himself exactly. He may describe aspects of the subject which he omitted on first telling or he may, on the other hand, describe things from a different perspective, giving the text a new emphasis. Chafe [CHAFE 79] has performed a series of experiments which he claims support the notion that the speaker decides as he is talking what material should go into

* I make no claims about the general nature of stored knowledge in this dissertation. For the purposes of text generation, any representation of knowledge could have been used; its structure is replaced, in any case, by the structure of the produced text. In practice, however, a particular representation for the given application had to be selected. Questions about how a representation can restrict the generation process, either in terms of content or ease of inferencing, are discussed in Chapter 4.3.

a sentence. These experiments show that the distribution of semantic constituents among sentences often varies significantly from one version of a narrative to another.

The second hypothesis central to this dissertation is that people have preconceived ideas about the means with which particular communicative tasks can be achieved as well as about the ways in which these means can be integrated to form a text. In other words, people generally follow standard patterns of discourse structure. For example, they commonly begin a narrative by describing the setting (the scene, the characters, or the time-frame).

In the TEXT system, these types of standard patterns of discourse structure have been exploited through the use of schemas. A schema is a representation of a standard pattern of discourse structure which efficiently encodes the set of communicative techniques that a speaker can use for a particular discourse purpose. It defines a particular organizing principle* for text and is used to structure the information that will be included in the answer. It is used to guide the generation process, controlling decisions about what to say first and how to end a text. This mechanism embodies a computational treatment of rhetorical devices, which have not previously been formalized in such a way.

*It should be noted that the organizing principles developed are not extensible to new and different organizing principles. To develop new principles of organization, the same kind of analysis that was needed to develop the principles adopted here would be necessary.

2.1 Rhetorical Predicates

Rhetorical predicates are the means which a speaker has for describing information. They characterize the different types of predicating acts he may use and delineate the structural relation between propositions in a text. Some examples are "analogy" (the making of an analogy), "constituency" (description of sub-parts or sub-types), and "attributive" (providing detail about an entity or event). Linguistic discussion of such predicates (e.g. [GRIMES 75], [SHEPHERD 26]) seems to indicate that some combinations are preferable to others. The following sections give the linguistic background of rhetorical predicates.

2.1.1 Linguistic Background -

The notion of the means available to a speaker or writer goes back to Aristotle, who describes the means which a speaker can use for persuasive argument in The Rhetoric. He distinguished between enthymemes (or syllogisms) and examples, where syllogisms are argument types and examples provide evidence for different arguments.

Shepherd, an early 20th century grammarian, categorized sentences by their function [SHEPHERD 26] in order to illustrate to the beginning writer how to construct paragraphs. Some of the functions he identified include: topic, general illustration, particular illustration, comparison, amplification, contrasting sentences, and

conclusions. Although Shepherd enumerated many of the "do's" and "don'ts" of writing, he said nothing about combining sentence functions to form paragraphs. He merely cited examples of prose that he considered well-done and identified the function of each sentence in the examples.

In more recent years, Grimes describes rhetorical predicates as explicit organizing relations used in discourse [GRIMES 75]. Grimes distinguishes three functions that predicates can serve in discourse:

1. supporting or supplementary (which add detail, explain, or substantiate what has come before. The three examples of predicates given above fall into this category.)
2. setting (which locate an object or event in space or time)
3. identification (which establish or maintain reference to an object)

Grimes claims that the predicates are recursive and can be used to identify the organization of text at any level (i.e. proposition, sentence, paragraph, or longer sequence of text), but does not show how.

2.1.2 Ordering Communicative Techniques -

Although the use of rhetorical predicates in text as structuring devices has been considered, most researchers have not discussed the ways in which they may be combined to form larger units of text. Both Grimes and Shepherd imply this use however. Grimes claims that the predicates are recursive, and Shepherd cites examples of well-written prose, identifying the predicates used.

My own examination of texts and transcripts has shown that not only are certain combinations of rhetorical techniques more likely than others, certain ones are more appropriate in some discourse situations than others. For example, I found that identification of objects was frequently achieved by employing some combination of the following means: (1) identification of an item as a member of some generic class, (2) description of an object's function, attributes, and constituency (either physical or class), (3) analogies made to familiar objects, and (4) examples. These techniques were rarely used in random order; for instance it was common to identify an item as a member of some generic class before providing examples.

For this analysis of rhetorical predicates, a variety of texts were examined - ten different authors, in varying styles, from very literate written to transcribed spoken texts form the basis of the study. Short samples of expository writing were used since this seemed most relevant to the system being developed. This also avoided problems involved in narrative writing (e.g. - scene, temporal

description, personality). The data were drawn from the following texts: Working (the introduction plus two transcriptions) [TERKEL 72], Dictionary of Weapons and Military Terms [QUICK 73], Encyclopedia Americana [ENCYCLOPEDIA 76], The Hamlyn Pocket Dictionary of Wines [PATERSON 80], The Poorperson's Guide to Great Cheap Wines [NELSON 77], "The American Style of Warfare and Military Balance" [LUTTWAK 79], Future Facts [ROSEN 76], "Toxicants occurring naturally in spices and flavors" [HALL 73], transcripts of mother-child dialogues*, [SHIPLEY 80], transcripts of user interaction with database systems [MALHOTRA 75] and "Tactical Nuclear Weapons" [MARTIN 73].

Each proposition in the texts** was classified as one of the set of predicates shown in Figure 2.1. A proposition is a simple predicating act and can surface linguistically as either a sentence, sentence fragment, or a clause. A proposition was classified as a single predicate taken from any of the three groups shown. In a few cases it was difficult to classify a proposition definitively as a single predicate. In such cases, the ambiguous proposition was assigned several predicates. The first group of predicates was taken from [GRIMES 75]. The second group of predicates was taken from

*These are transcripts of taped dialogues between mothers and their children where the mothers were asked to show their child pictures of familiar and unfamiliar objects and discuss them. Some mothers described the picture in great detail, while others provided minimal comments. The dialogues were taped by Liz Shipley and her colleagues for psychological experiments.

**Only a sampling of paragraphs was used from each text.

[SHEPHERD 26]. Some of these are somewhat similar to those proposed by Grimes, but provide a viewpoint different enough to be useful. For example, "conclusion" names a predicate which draws a conclusion from the previous discourse, while Grimes' "inference" identifies a specific fact deduced from previous facts.

The final group of predicates are some that I found necessary to add during the analysis of texts. "Identification" identifies an entity as belonging to a specific class (the opposite of Grimes' "constituency"). The predicate may be followed by attributes or functions which further identify the entity. "Positing" simply introduces an entity into the text (e.g. - "Just think of Marcus Welby" [TERKEL 72], "Movies set up these glamorized occupations"). Further discussion of the entity was only provided in succeeding sentences and not in the positing proposition. "Renaming" provides alternative names for an entity (e.g. - "Also known as the 'Red Baron' ...").

FIGURE 2.1

Rhetorical Predicates and Examples

Each predicate is followed by an example English sentence* In some cases, a preceding sentence was needed to provide a context in which to give the example. In such cases, the example illustrating the predicate is underlined.

1. Attributive
   Mary has a pink coat.

2. Equivalent
   Wines described as 'great' are fine wines from an especially good village.

3. Specification (Specification of general fact)
   Mary is quite heavy. She weighs 200 pounds.

4. Explanation (reasoning behind an inference drawn)
   So people form a low self-image of themselves, because their lives can never match the way Americans live on the screen.

5. Evidence (evidence for a given fact)
   The audience recognized the difference. They started laughing right from the very first frames of that film.

6. Analogy
   You make it in exactly the same way as red-wine sangria, except that you use any of your inexpensive white wines instead of one of your inexpensive reds.

7. Representative (item representative of a set)
   What does a giraffe have that's special? ... a long neck.

8. Constituency (presentation of sub-parts or sub-classes)
   This is an octopus... There is his eye, these are his legs, and he has these suction cups.

9. Covariance (antecedent, consequent statement)
   If John went to the movies, then he can tell us what happened.

10. Alternatives
    We can visit the Empire State Building or call it a day.

11. Cause-effect
    The addition of spirit during the period of fermentation arrests the fermentation development ...

12. Adversative
    It was a case of sink or swim.

13. Inference
    So people form a low self-image of themselves.

[GRIMES 75]

Shepherd's predicates are illustrated by providing an example paragraph from his text in which each sentence is classified as one of his predicates.

Comparison	Topic
General illustration	Particular illustration
Amplification	Contrasting
Conclusion	

"What, then, are the proper encouragements of genius? (topic) I answer, subsistence and respect, for these are rewards congenial to nature. (amplification) Every animal has an aliment suited to its constitution. (general illustration) The heavy ox seeks nourishment from earth; the light chameleon has been supposed to exist on air. (particular illustration) A sparer diet than even this satisfies the man of true genius, for he makes a luxurious banquet upon empty applause. (comparison) It is this alone which has inspired all that ever was truly great and noble among us. It is as Cicero finely calls it, the echo of virtue. (amplification) Avarice is the pain of inferiour natures; money the pay of the common herd. (contrasting sentences) The author who draws his quill merely to take a purse no more deserves success than he who presents a pistol. (conclusion) "

[SHEPHERD 26]

Additional Predicates needed for the analysis

1. Identification
   ELTVILLE (Germany) An important wine village of the Rheingau region.

2. Renaming
   Also known as the Red Baron.

3. Positing
   Just think of Marcus Welby.

FIGURE 2.1 Rhetorical Predicates and Examples

2.2 Analysis Results

My analysis has shown that, with slight variations, similar patterns of predicate usage occur across the various expository texts. These patterns have been represented as schemas. Schemas are recursive descriptions and may be embedded in other schemas to form paragraphs. In addition, in the texts a paragraph was sometimes introduced by the positing predicate. Allowing for schema embedding and positing initial sequences, each paragraph that was examined (a total of 56) could be described by one of the schemas developed.*

The schemas are shown in Figures 2.2 - 2.5. "{}" indicate optionality, "/" indicates alternatives, "+" indicates that the item may appear 1-n times, and "*" indicates that the item is optional and may appear 0-n times. Each schema is followed by a sample paragraph

---

*Note that in order to make such an analysis, the function of each proposition had to be determined and a predicate assigned to it. Since there are no hard and fast rules for predicate assignment, the analysis is subjective and could have had somewhat different results if done by someone else.

taken from the data and a classification of the propositions contained in the paragraph. ";" is used to represent classification of ambiguous propositions in the paragraph. These were translated into the schemas as alternatives.

The attributive schema (Figure 2.2) can be used to illustrate a particular point about a concept or object. The sample paragraph, taken from the Introduction to Working, attributes the topic (working and violence) to the book, amplifies on that ("spiritual as well as physical") in proposition 2), and in the third sentence, provides a series of illustrations. The fourth selects out one instance as representative of the problem and the fifth amplifies on that instance.

The identification schema (Figure 2.3) is used to identify entities or events. The characteristic techniques it uses to do so include identification, particular illustration, evidence, analogy, renaming, and various descriptive predicating acts. It should be noted that the identification schema was only found in texts whose primary function was to provide definitions (i.e. - dictionaries and encyclopedias). Moreover, the schema represents the types of definitions provided in the particular examples analyzed but does not dictate what every definition must look like. For example, some definitions may be provided by describing process information associated with the term. The other texts simply did not have occasion to provide definitions.

Attributive Schema

Attributive
{Amplification; restriction}
Particular illustration*
{Representative}
{Question; problem
Answer} /
{Comparison;contrast
Adversative}
Amplification/Explanation/Inference/
Comparison

Example

" 1) This book, being about work, is, by its very nature, about violence - 2) to the spirit as well as to the body. 3) It is about ulcers as well as accidents, about shouting matches as well as fistfights, about nervous breakdowns as well as kicking the dog around. 4) It is, above all (or beneath all), about daily humiliations. 5) To survive the day is triumph enough for the walking wounded among the great many of us."

[TERKEL 72]

Example Classification

1. Attributive
2. Amplification
3. Particular illustration
4. Representative
5. Amplification; explanation

FIGURE 2.2 The Attributive Schema

Identification Schema

Identification (class & attribute/function)
{Analogy/Constituency/Attributive/Renaming}*
Particular-illustration/Evidence+
{Amplification/Analogy/Attributive}
{Particular illustration/Evidence}

Example

"Eltville (Germany) 1) An important wine village of the Rheinga region. 2) The vineyards make wines that are emphatically of th Rheingau style, 3) with a considerable weight for a white wine 4) Taubenberg, Sonnenberg and Langenstuck are among vineyards of note. [PATERSON 80]

Example Classification

1. Identification (class & attribute)
2. Attributive
3. Amplification
4. Particular illustration

FIGURE 2.3 The Identification Schema

The constituency schema (Figure 2.4) describes an entity or event in terms of its sub-parts or sub-types. After identifying its sub-types, the focus can either switch to each of its sub-types in turn (following the depth-identification or depth attributive path) or can continue focusing on the entity itself, describing either its attributes (attributive path) or its functions (cause-effect path). The schema may end by optionally returning to discussion of the original by using the amplification, explanation, attributive, or analogy predicate.

In the sample paragraph, taken from the American Encyclopedia, part of the entry under torpedo includes a description of its classification. In the section title and first sentence, the two types of torpedoes are introduced. First the steam-propelled model is identified by citing facts about it and then the electric-powered model is compared against it, with the significant difference cited.

Constituency Schema

Constituency
Cause-effect*/Attributive*/
{ Depth-identification/Depth-attributive
{Particular-illustration/evidence}
{Comparison;analogy} } +
{Amplification/Explanation/Attributive/
Analogy}

Example

Steam and electric torpedoes. 1) Modern torpedoes are of 2 general ypes. 2) Steam-propelled models have speeds of 27 to 45 knots and anges of 4000 to 25,000 yds* (4,367 - 27,350 meters). 3) The lectric powered models are similar 4) but do not leave the telltale ake created by the exhaust of a steam torpedo."
[ENCYCLOPEDIA 76]

Example Classification

1. **Constituency**
2. **Depth-identification (attributive)**
3. **Comparison**
4. **Depth-identification (attributive)**

FIGURE 2.4 The Constituency Schema

The contrastive schema (Figure 2.5) is used to describe something by contrasting it against something else* The speaker may contrast his major point against something more negative. The lesser item (to be contrasted against) is introduced first. The major concept is then described in more detail using one or more of the predicates shown in the second option of the schema• The closing sequence makes a direct comparison between the two. This schema dictates the structural relation between the two concepts (the use of A and ~A (not A) in the schema represent the major and lesser concepts), but is less restrictive about which predicates are used.

In the sample paragraph, the contrastive schema is used to show how people form a bad self-image by comparing themselves against those in the movies. In the first sentence, the movie standard is introduced. In the second and third sentence, real-life occupations and the feelings associated with them are described. Finally, a direct comparison is made between the two situations and an inference drawn: "people form a low self-image of themselves."

Compare and Contrast Schema

Positing/Attributive (~A)
{Attributive (A) /
  Particular illustration/Evidence (A) /
  Amplification (A) /
  Inference (A)
  Explanation (A) } +
{Comparison (A and ~A) /
  Explanation (A and ~A) /
  Generalization (A and ~A) /
  Inference (A and ~A) } +

" 1) Movies set up these glamorized occupations. 2) When people find they are waitresses, they feel degraded. 3) No kid says I want to be a waiter, I want to run a cleaning establishment. 4) There is a tendency in movies to degrade people if they don't have white-collar professions. 5) So, people form a low self-image of themselves, 6) because their lives can never match the way Americans live -- on the screen."

[TERKEL 72]

Example Classification

1. Positing (~A)
2. Attributive (A)
3. Evidence (A)
4. Comparison;explanation (A and ~A)
5. Inference (A and ~A)
6. Comparison;explanation (A and ~A)

FIGURE 2.5 The Compare and Contrast Schema

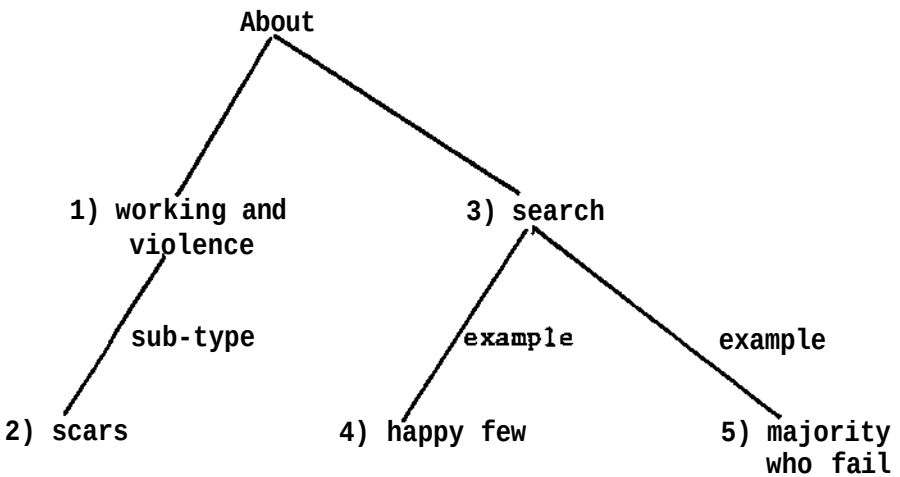
2.2.1 Predicate Recursiveness -

Although the examples above only show how the schemas work at the paragraph level, there is evidence that such organization also occurs at higher levels of text. The schemas were found to apply to a sequence of paragraphs, with each predicate in the schema matching an entire paragraph, instead of a single proposition. The Introduction to Working, for example, covers three major topics, each of which is introduced and closed within four or five paragraphs. The first topic group follows the attributive schema (the text for this topic group is reproduced in Appendix A); each paragraph in the group matches a single rhetorical predicate.* Figure 2.6 shows a tree representing the first topic group of the Introduction. Paragraphs are numbered nodes in the tree. The tree is described by the predicates listed at the bottom of the figure which is an instantiation of the attributive schema.

*Again, note that the analysis is somewhat subjective.

Introduction to Working

Topic Group 1



1) Attributive
2) Restriction
3) Attributive
4) Particular-illustration
5) Particular-illustration

FIGURE 2.6 Introduction to Working

Thus, the predicates do indeed seem to function recursively as Grimes suggests. Schemas, since they consist of predicates, also function recursively; that is, each predicate in the schema can expand to another schema. The structure of a text when described by the schemas is, therefore, hierarchical. Each node in the hierarchical

structure corresponds to a predicate. The predicate can either be interpreted as a single predicate or can be expanded to another set of predicates representing the schema named.

2.2.2 Summary Of Results -

The analysis of texts and transcripts shows that patterns do occur across a variety of text styles. It appears, however, that the patterns are very loose. Each schema contains a number of alternatives, indicating that a speaker has a wide variety of options within each type of structure. Moreover, since it is difficult to precisely define a predicate, the interpretation of each predicate in the schema allows for additional speaker variation.

It should be noted, furthermore, that the schemas are descriptive and not prescriptive. Any discourse norm developed over a period of time will eventually be broken in order to achieve a desired literary effect. Poetic license, in fact, is based on the breaking of norms. It may be that norms at the discourse level are broken to create implicatures similarly to the creation of implicatures at the sentence level [GRICE 75]. All this points to the fact that the schemas do not function as grammars of text.

The schemas are useful, however, in identifying common means for achieving discourse goals. They are intended to loosely identify normal patterns of usage. In addition, the alternatives encoded in

each schema provide enough variety to produce different texts for the type of generation required in this dissertation.

^•^ Related Research Using Rhetorical Predicates

One computational use made of rhetorical predicates has been in the interpretation of arguments [COHEN 81R]. Cohen's goal is to determine argument structure. She uses linguistic clues in the text to aid in determining the rhetorical function of a proposition and thereby, the supporting relations between propositions in the text. Some of the predicate types which Cohen uses include claim, evidence, and inference. It should be noted that Cohen assumes an "oracle" which does the classification of propositions as predicates.

Another proposed use of rhetorical predicates is in the generation of paragraphs [JENSEN 81]. Jensen assumes that the content of the paragraph has already been determined. The function of each proposition is then determined and is used to aid in the development of paragraph style. By identifying the underlying structure between propositions, they can be combined appropriately in text.

2.4 Use Of Schemas

In the TEXT system, schemas describing discourse structure are used to guide the generation process. They are used to decide what is said first, what next, and so forth. The four schemas noted in Figures 2.2 - 2.5 above (identification, attributive, constituency, and compare and contrast) are used in the TEXT system with minor variations.

The identification, constituency, and attributive schemas were modified by eliminating several predicates for which no corresponding information exists in the specific application. Specifically, the renaming predicate was eliminated since synonyms are not represented or used in the TEXT system, and the Cause-effect predicate was eliminated since no process information is represented.

The compare and contrast schema was modified to allow for equal discussion of the two items in question. Recall that the contrastive schema which emerged from the text analysis called for contrasting a major concept against a minor one. The minor concept, had, in most cases, either been discussed in the preceding text, or was assumed by the writer to be familiar to the reader. Thus, more discussion of the major concept was provided. Since no history of discourse is currently maintained in the TEXT implementation (see Chapter 5 for suggestions for future incorporations) and no user model other than a static one, is constructed, the system does not know whether the user has more knowledge about one concept than another and the comparison, therefore, must be equally balanced between the two. An example of an equally

balanced comparison taken from the texts analyzed is shown below in (A) (the basic outline of the compare and contrast schema used in TEXT is shown).

(A) "Made by" vs. "Produced by"

Similarities

Each listing also states that the wine was "produced and bottled by," or "made and bottled by," or "cellared and bottled by" a particular vintner. In the case of California wines, this is a very rough guide to how much of the wine in the bottle was actually fermented and finished by the company that put it into the bottle.

Differences

If the label states "produced and bottled by," then at least 75 percent of the wine was fermented and finished by that winery. If the label says "made and bottled by," then only 10 percent of the wine need have been produced by the winery, and the other 90 percent or some portion of it may have been bought from another source and blended into the final product. If the label says anything else -- "cellared," "vinted," "bottle," "perfected," or any long and glorious combination of these words, then none of the wine in the bottle need have been produced by that winery.

Inference

The fact that the label says simply "Bottled by Jones Brothers Winery" doesn't mean the wine is no good, however. It may be excellent. Its goodness will simply depend on the ability of the Jones Brothers to buy good wine, rather than on their ability to make it.

[NELSON 77]

2.4.1 Associating Technique With Purpose -

In the texts I analyzed, different rhetorical techniques were found to be used for different discourse purposes. In the TEXT system, this association of technique with discourse purpose is achieved by associating the different scheraas with different question-types. For example, if the question involves defining a term, a different set of schemas (and therefore rhetorical techniques) is chosen than if the question involves describing the type of information available in the knowledge base.

In the first case, the identification schema is used. (In fact, it is only used in response to a request for a definition.) On the other hand, the purpose of the attributive schema is to provide detailed information about one particular aspect of any concept and can therefore be used in response to a request for information. In situations where an object or concept can be described in terms of its sub-parts or sub-classes, the constituency schema is used. It may be selected in response to requests for either definitions or information. The compare and contrast schema is used in response to a question about the difference between objects. It makes use of each of the three other schemas (see Section 2.10). A summary of the assignment of schemas to question-types is shown in Figure 2.7.

Schemas used for TEXT

1. identification

   -requests for definitions

2. attributive

   -requests about available information

3. constituency

   -requests for definitions

   -requests for available information

4. compare and contrast

   -requests about the differences between objects

FIGURE 2.7 Schemas used for TEXT

It should be noted that the compare and contrast schema has many uses and is an expository device frequently used in many of the texts analyzed. This schema is appropriate as the response structure for any question type when an object similar to the questioned object has been discussed in the immediately preceding discourse or is assumed to be familiar to the reader. In such situations, it serves two purposes: 1) it can point out the ways in which the questioned object differs from a concept familiar to the user; and 2) it can be used to parallel

the structure of an earlier answer. This type of response would require using the one-sided compare and contrast schema that most of the analyzed texts used. Although use of the compare and contrast schema for questions other than "What's the difference ..." questions was not implemented in the TEXT system, it would be a straightforward extension if a discourse history record were implemented.

2.5 Selecting A Schema

Once a question has been posed to the TEXT system, a schema must be selected for the response structure which will then be used to control the decisions involved in deciding what to say when. On the basis of the given question, a small set of schemas is selected as possible structures for the response. This set includes those schemas associated with the given question-type (see Figure 2.7 above). A single schema is selected out of this set on the basis of the information available to answer the question.

In response to requests for definitions and information, the constituency schema is selected when the relevant knowledge pool contains a rich description of the questioned object's sub-classes and less information about the object itself. When this is not the case, the identification schema is used for definition questions and the attributive schema is used for information questions. The test for what kind of information is available is a relatively simple one. If the questioned object occurs at a higher level in the hierarchy than a

pre-determined level, the constituency schema is used. Note that the higher an entity occurs in the hierarchy, the less descriptive information is available to describe the set of instances it represents since the larger the class, the less common features occur across it. The pre-determined level is the level at which entity-classes in the database occur in the hierarchy. Thus, above this level the constituency schema will be used and below it the attributive or identification schema will be used. This process assumes a hierarchically structured knowledge base and could not be done on an unstructured one (see Chapter 4.3 for a description of the knowledge base used in the TEXT system).

(B) and (C) below show two examples of a request for a definition. For the question "What is a guided projectile?" (B) the constituency schema is selected since more information is available about the guided projectile's sub-classes than about the guided projectile itself, while the identification schema is selected for the question "What is an aircraft-carrier?" (C).

EXAMPLE B

(definition GUIDED)

Schema selected: constituency

identification

constituency

attributive

attributive

evidence

evidence

attributive

Message through dictionary. Entering tactical component

A guided projectile is a projectile that is self-propelled. There are 2 types of guided projectiles in the ONR database: torpedoes and missiles. The missile has a target location in the air or on the earth's surface. The torpedo has an underwater target location. The missile's target location is indicated by the DB attribute DESCRIPTION and the missile's flight capabilities are provided by the DB attribute ALTITUDE. The torpedo's underwater capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUM_OPERATING_DEPTH). The guided projectile has DB attributes TIME_TO_TARGET__UNITS, HORZ_RANGE__UNITS and NAME.

EXAMPLE B

EXAMPLE C

(definition AIRCRAFT-CARRIER)

Schema selected: identification

identification

analogy

particular-illustration

amplification

evidence

Message through dictionary. Entering tactical component

An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships. Mine warfare ships, for example, have a DISPLACEMENT of 320 and a LENGTH of 144. All aircraft carriers in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, BEAM of 252, ENDURANCE_RANGE of 4000, ECONOMIC_SPEED of 12, ENDURANCE_SPEED of 30 and PROPULSION of STMTURGRD. A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL_NO are CV.

EXAMPLE C

2.6 Filling The Schema

Once a schema has been selected, it is filled by matching the predicates it contains against the relevant knowledge pool (see Chapter 4.4 for discussion of the relevant knowledge pool). Semantics associated with each predicate define the type of information it can match in the knowledge pool. The semantics defined for TEXT are particular to the database system and would have to be redefined if the schemas were to be used in another type of system (such as a tutorial system, for example). The semantics are not particular, however, to the domain of the database. When transferring the system from one database to another, the predicate semantics would not have to be altered.

Before describing predicate semantics in more detail, it is important to note the difference between a rhetorical predicate and a proposition. A rhetorical predicate specifies a generic type of speech act. It has arguments associated with it which can take any value of a given type. The number and types of arguments associated with a predicate depend upon its semantics. A proposition is an instantiation of a predicate; the predicate arguments have been filled with values from the knowledge base. Furthermore, although predicates, loosely speaking, match propositions in the knowledge base, information in the knowledge base is not stored in the same formalism as are the propositions selected for the answer (see Chapter 4.3 for a description of the knowledge base representation). Instead, pieces of the knowledge base are selected as values for the predicate arguments.

The semantics defined in TEXT for a predicate indicate the type of information in the knowledge base that can satisfy each predicate argument. A single predicate may match several types of information in the database. The attributive predicate, for example, may be satisfied by the database attributes of an entity or by its distinguishing descriptive attributes (see Chapter 4.3). The semantics for the attributive predicate, therefore, indicate that the following two English sentences both attribute information to the missile:

1. The missile has database attributes TIME_TO_TARGET_&_UNITS, LETHAL_RADIUS_&_UNITS, ALTITUDE, SPEED, and PROBABILITY_OF_KILL. (database attributes)
2. The missile has a target location in the air or on the earth's surface. (distinguishing descriptive attribute)

The constituency predicate, on the other hand, has only one interpretation. It matches the sub-classes of an entity in the generalization hierarchy and would translate to an English sentence like: "There are two types of water-going vehicles in the ONR database: ships and submarines."

The semantics of the predicates are represented as functions. Associated with each predicate is a function (named <predicate>-fn - e.g. "attributive-fn") that accesses the relevant knowledge pool and retrieves values for the predicate arguments. Each predicate function takes a particular data type (or types) as its argument(s). It returns a set of propositions which match the predicate in the relevant knowledge pool. The attributive predicate, for example, takes an

entity as a given argument and searches for its database attributes and distinguishing descriptive attributes in the relevant knowledge pool.

The evidence predicate also takes an entity as its first argument, but has as an optional second argument a distinguishing descriptive attribute of the entity. In that case, it will only find the supporting database attributes for the given descriptive attribute. Otherwise it will provide a separate proposition for each descriptive attribute of the entity.*

The values for the arguments which are passed to the predicate functions are, in some cases, supplied by focussed elements in the previous discourse. In other cases, the function extracts an instance of the data type it is looking for from the most recent proposition which contains it.

As discussed above, the predicate semantics also specify the arguments of the predicate and their ordering. This frame-like specification is called the message formalism in the TEXT system. Each predicate has its associated formalism. When a predicate is evaluated, one or more of its arguments are given and the others are filled by values in the database to form a proposition. The predicates which may

---

*An entity has more than one descriptive attribute if it has more than one parent in the hierarchy. For example, the missile has one descriptive attribute distinguishing it from all other guided projectiles (its target location) and a second descriptive attribute distinguishing it from all air-operated entities (e.g. that it is a self-propelled weapon). Evidence for the two attributes would be provided in two separate propositions.

be matched by different types of data in the knowledge base have an argument which specifies the matching type used in the particular proposition.

Complete specification of the predicates and their formalism is given in Figure 2.8. Note that the formalism is closely tied with the data-types of the knowledge base which are described in Chapter 4.3. For each predicate, the given argument, its formalism, and an instantiated example of its formalism are listed. Angle brackets are used in the formalism to indicate which arguments must be instantiated. Other arguments are constants. Data types are explicitly indicated for each predicate that can match more than one type of information in the knowledge base.

The schema is filled by stepping through it, using the predicate semantics to select propositions which match the predicates. For cases where a single predicate has several types and matches more than one proposition in the knowledge base, the focus constraints are used to select the most appropriate proposition (see Chapter 3 for a description of the focus constraints). In places where alternative predicates occur in the schema, all alternatives are matched against the relevant knowledge pool, producing a set of propositions (if more than one predicate succeeds). Again, the focus constraints are used to select the most appropriate proposition. When optional predicates occur in the schema, both the optional predicate and the predicate which would succeed it are matched against the knowledge pool. If the optional predicate has no match, the successor's match is selected. If

both predicates match, the focus constraints are used to select most appropriate proposition*

After a proposition has been selected, it is marked in order prevent repetition in a single answer. Since a proposition ma composed of pieces of information in the knowledge pool, each piece information is marked by adding the property "used" to it. selecting propositions, this property is checked to determine whe it has already been said. Since no tracking of discourse is done i now, the "used" slate is wiped clean after the generation of answer.

FIGURE 2.8 Predicate Semantics

Predicate	Formalism
Attributive	
given-argument: entity	
type: db-attributes*	
sub-type: attributes only	(attributive db <entity> <naming-attr> <topic-attr> <duplicate-attrs> <db-attrs>)
example:	(attributive db SHIP (name NAME) (topics DIMENSIONS) (duplicates (FUEL (FUELJTYPE FUEL_CAPACITY))) (attrs MAXIMUMJSPEED))
sub-type: attributes and values	(attributive db <entity> (<attrl val2>) ... (<attrn> <valn>))
example:	(attributive db AIRCRAFT-CARRIER (PROPULSION STMTURGRD) (ENDURANCE_SPEED 30) (ECONOMICJSPEED 12) (ENDURANCE_RANGE 4000) (BEAM 252) (FLAG BLBL) (FUELJTYPE BNKR) (REMARKS 0))
type: distinguishing descriptive attribute	(attributive def <entity> <parent> <attr-name> <attr-value>)
example:	(attributive def MISSILE GUIDED TARGET-LOCATION SURFACE-AIR)
	(attributive def ECHO-II-SUBMARINE SUBMARINE ((FLAG) (PROPULSIONJTYPE)) (((FLAG (RDRD))) ((PROPULSION_TYPE (NUCL)))))

*Any type of db attributes (naming-attrs, topic-attrs, etc.) an optional. If none are present in the knowledge pool, nil is returned.

Evidence

given-argument: entity
optional distinguishing descriptive attribute

(evidence based-db <entity> <def-attr>
<based-db1> ... <based-dbn>)

example:

(evidence based-db MISSILE TARGET-LOCATION
(indicated-by DESCRIPTION) (HAVE ALTITUDE))

Constituency

given-argument: entity

(constituency <entity> (<sub-class1> ...
<sub-classn>))

example:

(constituency WATER-VEHICLE (SHIP SUBMARINE))

Identification

given-argument: entity

(identification <entity> <super-ord>
(restrictive <attr-name> <attr-value>)**
(non-restrictive <attr-name> <attr-value>)***)

example:

(identification SHIP WATER-VEHICLE
(restrictive TRAVEL-MODE SURFACE)
(non-restrictive TRAVEL-MEDIUM WATER))

**Restricts the class of super-ordinates to only those having the given attribute-value pair.

***All super-ordinates, and thus entities, have the given attribute-value pair.

Amplification

given-argument: entity
plus either a descriptive attribute, database attribute, or relation on which to amplify

type: amplification on db-attributes

sub-type: attributes only

(amplification db <entity> <old-db>
<naming-attr> <topic-attrs> <duplicate-attrs>
<db-attrs>)

example:

(amplification db AIRCRAFT (topics ROLE FUEL
CEILING FLIGHT_RADIUS) (name NAME)
(attrs PROPULSION MAXIMUM_SPEED CRUISE_SPEED))

sub-type: attributes and values

(amplification db <entity> <old-db>
(<attr1> <val1>) ... (<attrn> <valn>))

example:

(amplification db AIRCRAFT-CARRIER (((LENGTH
(1039 1063))) ((DISPLACEMENT (78000 80800))))
(PROPULSION STMTURGRD)(ENDURANCE_SPEED 30)
(ECONOMIC_SPEED 12) (ENDURANCE_RANGE 4000)
(BEAM 252) (FLAG BLBL)(FUEL_TYPE BNKR)
(REMARKS 0))

type: amplification on descriptive attributes

sub-type: new descriptive attribute

(amplification def <entity> <old-def>
<parent> <attr-name> <attr-value>)

example:

(amplification def MISSILE (TARGET-LOCATION
SURFACE-AIR) AIR FUNCTION
SELF-PROPELLED-TO-TARGET)

sub-type: evidence for descriptive attribute

(amplification evidence <entity> <old-def>
<parent> <based-db1> ... <based-dbn>)

example:

(amplification evidence SHIP (TRAVEL-MODE
SURFACE) (HAVE DRAFT) (HAVE DISPLACEMENT))

type: amplification on relations

(amplification rel <entity> <old-relations>
<new-relations>)

example:

(amplification rel SHIP (POSSESSION-06
POSSESSION-03) (CARRY-01))

Analogy

given: entity plus an optional database attr-range pair*

type: relation

sub-type: no values

(analogy rel <entity> <rel-name>
<related-entities>)

example:

(analogy rel SHIP ON GUIDED GUNS)

sub-type: with values

(analogy rel <entity> <rel-name>
<related-entity value pairs>)

example:

(analogy rel AIRCRAFT-CARRIER ON (6 GUNS)
(3 MISSILES) (20 TORPEDOE))

type: range-comparison

(analogy range <entity> <entity ranges>
<comparison-attr1> ... <comparison attrn>)

example:

(analogy range AIRCRAFT-CARRIER (((LENGTH
(1039 1063))) ((DISPLACEMENT (78000 80800)))
(larger-than-all LENGTH) (larger-than-most
DISPLACEMENT))

Particular-illustration

given-argument: entity
either database attributes, attr-range pairs,

---

*A database attr-range pair is a database attribute associated wi
constant numeric range which indicates the values over whic
attribute ranges for the particular entity.

abstract attributes*

type: database attributes
(particular-illustration <entity> <given db-attrs>
<attr value>1 ..... <attr value>n)

example:
(particular-illustration SHIP ((name NAME)
(topics SPEED_DEPENDENT_RANGE DIMENSIONS)
(duplicates (FUEL (FUEL_TYPE FUEL_CAPACITY)))
(attrs (MAXIMUM_SPEED)))
(OFFICIAL_NAME DOWNES) (ENDURANCE_RANGE 2200)
(ECONOMIC_RANGE 4200) (LENGTH 438) (BEAM 46)
(DRAFT 25)(FUEL_TYPE BNKR) (FUEL_CAPACITY 810)
(PROPULSION STMTURGRD)(MAXIMUM_SPEED 29))

type: attribute range pairs
(particular-illustration <new-entity>
<old-attr-range> <attr range>1 ... <attr range>n)

example:
(particular-illustration MINE-WARFARE-SHIP
(((LENGTH (1039 1063))) ((DISPLACEMENT (78000
80800)))) (LENGTH (144)) (DISPLACEMENT (320)))

type: abstract attrs
(particular-illustration abstract <entity>
<abstract-attrs> <attr1> ... <attrn>)

example:
(particular-illustration abstract (topics
ROLE FUEL CEILING FLIGHT_RADIUS) (ROLE
REMARKS DESCRIPTION PRIMARY_ROLE) (FUEL
FUEL_TYPE FUEL_CAPACITY REFUEL_CAPABILITY)
(CEILING MAXIMUM_CEILING COMBAT_CEILING)
(FLIGHT_RADIUS COMBAT_RADIUS CRUISE_RADIUS))

Explanation

given-argument: entity

type: distinguishing descriptive attribute
(explanation def <entity> <parent> <attr-name>

---

*Abstract attributes are those occurring in the topic hierarchy. They represent a set of related database attributes.

<attr-value>)

example:

(explanation def AIRCRAFT AIR-VEHICLE
TRAVEL-MODE FLIGHT)

type: based database attributes

(explanation based-db <entity> <attr-value pair>
<based-db>)

example:

(explanation based-db ECHO-II-SUBMARINE
(((FLAG) (PROPULSION_TYPE) (((FLAG (RDRD)))
((PROPULSION_TYPE (NUCL)))))) (CLASS ECHO II))

Classification

given-argument: entity

type: greater than one breakdown*

(classification <entity> (<based-db-attr
<sample sub-type>)1 ... (<based-db-attr>
<sample-sub-type>)n)

example:

(classification AIRCRAFT (FLAG (example
BLBL-AIRCRAFT)) (PROPULSION (example
JET-AIRCRAFT)))

type: one breakdown

(classification <entity> (<based-db-attr>
(<sub-type1> ... <sub-typen>)))

example:

(classification AIRCRAFT-CARRIER (CLASS
(KITTY-HAWK-SHIP FORRESTAL-SHIP)))

Inference

given-argument: 2 entities

*If an entity has two or more sets of mutually exclusive sub-types, only the attributes on which each set was based (and an example of a sub-type) is given in order to avoid putting too much information into a single sentence.

type: very close

sub-type: below database entity class

```
(inference <entity1> <entity2>
   <comparison1 db-attr1> ...
   <comparisonn db-attrn>)
```

example:

```
(inference OCEAN-ESCORT CRUISER
   (smaller DISPLACEMENT)
   (larger LENGTH))
```

sub-type: database entity class and above

```
(inference <entity1> <entity2>
   (same <attr-name>) (different
   <attr-name>)(<entity1> <based-db1>)
   (<entity2> <based-db2>) )
```

example:

```
(inference MISSILE TORPEDOE
   (same TRAVEL-MEANS (different
    TARGET-LOCATION) (MISSILE
    (INDICATED-BY DESCRIPTION)
    (HAVE ALTITUDE) ) (TORPEDOE
    (SOME-TYPE-OF DEPTH)) )
```

type: very-different

```
(inference <entity1> <entity2>
   very-different-entities)
```

example:

```
(inference DESTROYER BOMB
  very-different-entities)
```

DISCOURSE STRUCTURE

type: class difference

sub-type: below database entity class

(inference <entity1> <entity2>
two-kinds-of-entity)

example:

(inference WHISKY-SUBMARINE KITTY-
HAWK-SHIP two-kinds-of-entity)

sub-type: database entity class and above

(inference <entity1> <entity2>
(same <attr-name>) (different
<attr-name>))

example:

(inference GUN BOMB (same FUNCTION)
(different ROLE))

FIGURE 2.8 Predicate Semantics

2.7 An Example

To see exactly how a schema is filled, consider the process of answering the question "What is a ship?" (in functional notation "(definition SHIP)"), Two schemas are associated with definitions: constituency and identification. A test on the generalization hierarchy indicates that the ship occurs at a level where a large amount of information is available about the entity itself. The identification schema is therefore selected and the process of schema filling begins.

The first predicate in the schema is identification (reproduced in Figure 2.9). The relevant knowledge pool constructed for this question, is shown in Figure 2.10 (see Chapter 4.4 for the determination of relevant information). Since this is the first statement of the answer and no preceding discourse exists to provide a context for the predicate to use, the current focus (which is initialized to the questioned object — see the focus algorithm in Chapter 3, Section 3.2.9) is passed as argument to the identification function. In this case, the current focus = SHIP. The identification predicate is matched against the relevant knowledge pool and the ship's super-ordinate in the hierarchy, plus both the ship's and its super-ordinate's descriptive attributes are selected, as dictated by the semantics of the predicate. Note that the identification predicate has only one type and therefore, only one proposition matches it:

(identification SHIP WATER-VEHICLE (restrictive TRAVEL-MODE SURFACE) (non-restrictive TRAVEL-MEDIUM WATER))

Identification Schema

Identification (class & attribute / funct
{Analogy/Constituency/Attributive/Renaming}*
Particular-illustration/Evidence+
{Amplification/Analogy/Attributive}
{Particular-illustration/Evidence}

FIGURE 2.9 The Identification Schema Revisited

The second step in the schema specifies an optional alternat The alternative includes the descriptive predicates anal constituency, and attributive* Each of these predicates is mat against the relevant knowledge pool. Since each of these predic takes an entity as its given argument, both "SHIP" and "WATER-VEHI are passed to the various predicate functions ("SHIP" is the cur focus of the first proposition and ftWATER-VEHICLE" is a member of potential focus list and these are the only entities mentione far.)* Since quite a bit of information remains about the SHIP in relevant knowledge pool, each of these predicates matches and t propositions are produced. Since the only remaining information a the WATER-VEHICLE is its sub-classes, only the constituency predi matches for the WATER-VEHICLE. The 4 matched propositions are:

1. (analogy rels SHIP ON GUIDED GUNS)
2. (constituency SHIP (AIRCRAFT-CARRIER FRIGATE ...))
3. (attributive db SHIP (name OFFICIALJNAME) (tc SPEEDJDEPENDENT_RANGE DIMENSIONS) (duplicates (FUEL FUEL_ FUEL_CAPACITY)) (attrs PROPULSION MAXIMUM_SPEED))

4. (constituency WATER-VEHICLE (SHIP SUBMARINE))

Since the alternative is optional, its succeeding step (an alternative between particular-illustration and evidence) is also matched against the relevant knowledge pool. The same entities are passed as given arguments to the predicate functions. Since the second argument required by the particular illustration predicate (either database attributes, attribute-range pairs, or abstract attributes) does not exist in the discourse so far, there is nothing to illustrate and the particular-illustration predicate fails. The evidence function takes an optional descriptive attribute as a second given argument and since one exists in the discourse, the supporting database attributes (see Chapter 4.3 for a description of the knowledge base data types) for it are returned. No supporting database attributes for WATER-VEHICLE exist in the relevant knowledge pool, so this step matches one proposition:

1. (evidence based-db SHIP (TRAVEL-MODE SURFACE) (HAVE DRAFT) (HAVE DISPLACEMENT))

One proposition is then selected from this set of five by applying the focus constraints. In this case, the proposition matching the evidence predicate is selected, although the reasoning behind the choice is not discussed here since it depends on the focus constraints (see Chapter 3.2.9 for the focus algorithm). The answer created so far and the updated relevant knowledge pool (information occurring in the answer is marked as used) are shown in Figure 2.11. It should be noted

that the identification schema encodes more alternatives than the schemas and is therefore less efficient. Less restrictive sc necessarily entail more inefficiency than others as more proce must be done to explore the additional choices.

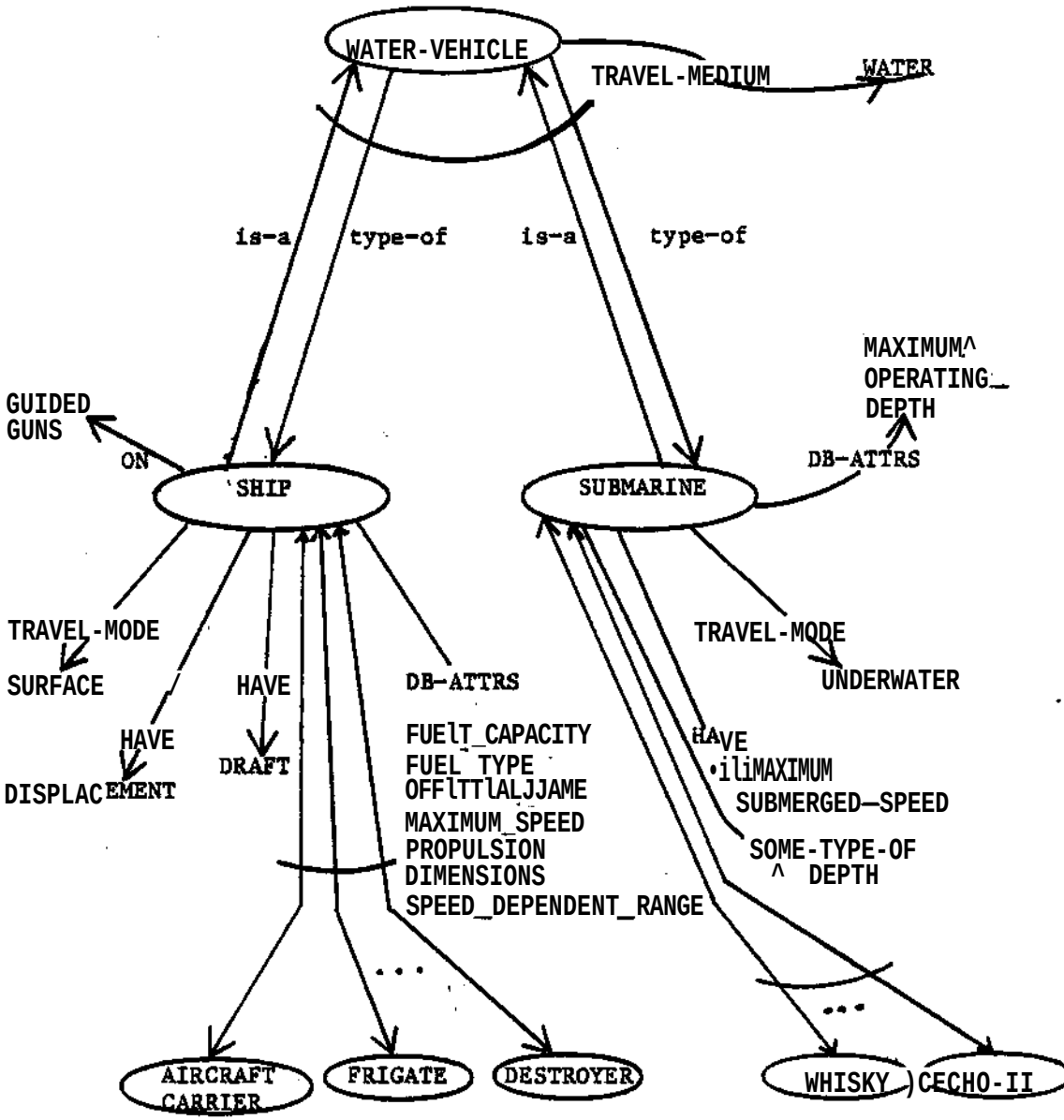
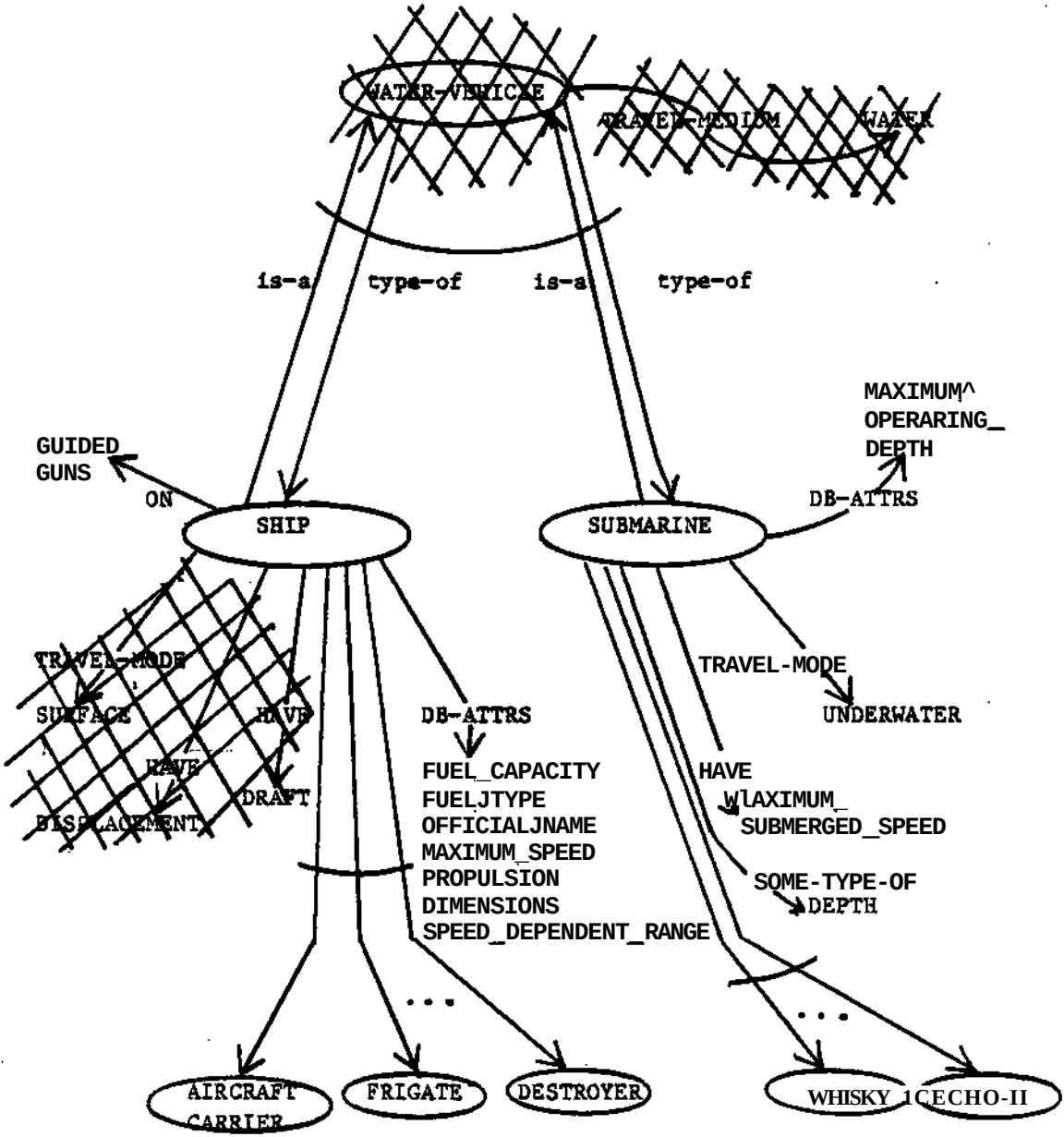


FIGURE 2.10 Sample Relevant Knowlege Pool

FIGURE 2.11 The Updated Relevant Knowledge Pool



Selected Propositions

(identification SHIP WATER-VEHICLE (restrictive TRAVEL-MODE SURFACE) (non-restrictive TRAVEL-MEDIUM WATER))
(evidence based-db SHIP (TRAVEL-MODE SURFACE) (HAVE DRAFT) (HAVE DISPLACEMENT))

Eventual Translation

The ship is water-going vehicle that travels on the surface. Its surface going capabilities are provided by the DB attributes DRAFT and DISPLACEMENT.

FIGURE 2.11 Updated Relevant Knowledge Pool and Selected Information

2.8 Schema Implementation

The schemas were implemented using a formalism similar to an augmented transition network (ATN). An ATN is a graph representation of a grammar and allows for actions on its arcs which may set or test various registers. The ATN formalism was originally developed to parse sentences. When parsing a sentence, taking an arc involves consuming a word from the input string and augmenting a syntactic parse tree to include the new word and its category. Notable features of the ATN include recursion and backtracking (see [WOODS 70]).

For generation, the ATN is used to build discourse instead of a parse tree. Taking an arc corresponds to the selection of a proposition for the answer and the states correspond to filled stages of the schema. No input string is consumed; instead the relevant knowledge pool is consumed, although it is not consumed in any order and it need not necessarily be completely exhausted when the graph is exited. The main difference between the TEXT ATN implementation and a usual ATN, however, is in the control of alternatives. In the TEXT system, at each state all possible next states are computed and a function that performs the focus constraints is used to select one arc from the set of possibilities. Thus, although all possible next states are explored, only one is actually taken.

The TEXT system originally used limited lookahead to avoid uncontrolled backtracking.* An arbitrary number of lookahead steps was used to eliminate traversing an arc which led to a blocked state (i.e.

- no propositions could be matched from this state). This feature was implemented, but it was found that, in practice, a blocked state was never reached. For reasons of efficiency, this feature was eliminated and the extra processing involved in lookahead avoided.

2.8.1 Arc Types -

The arc-types used in the schema implementation include:

1. fill <predicate> (to match predicate against the knowledge pool and retrieve propositions)
2. jump <state> (to jump to specified state)
3. subr <subr-state> (to proceed to start state of a subroutine graph. This arc type was included for a cleaner representation of the ATN graphs. They could have been implemented without this arc type)
4. end-subr (to return to state following subroutine call in main graph)
5. push <schema> (to recursively call a schema. All registers are saved)
6. pop (to return from a recursive schema call. All registers are restored).

2.8.2 Arc Actions -

The implementation allows for both pre-actions and post-actions to

---

*This refers to backtracking an arbitrary number of states and not to the one step next state exploration described above.

be associated with an arc. Pre-actions are performed before the arc i taken and usually include such things as resetting the relevan knowledge pool before taking a recursive push (see Section 2.9 below) Post-actions are performed after an arc has been taken and includ adding a proposition to the message, updating the focus record, an proceeding to the next state.

Arcs may also have tests. A test is performed before decidin whether to take an arc. If a test succeeds, the arc is taken. On test used in the TEXT system is on the question-type, since schemas ma be used for different purposes and sometimes a particular rhetorica technique is appropriate for one question-type and not for another.

2.8.3 Registers -

The registers used in the TEXT system maintain information abou the focus records, the message so far, and the question type. Th registers include the following:

CF (current focus)

GF (global focus)

PFL (potential focus list)

kpool (the relevant knowledge pool)

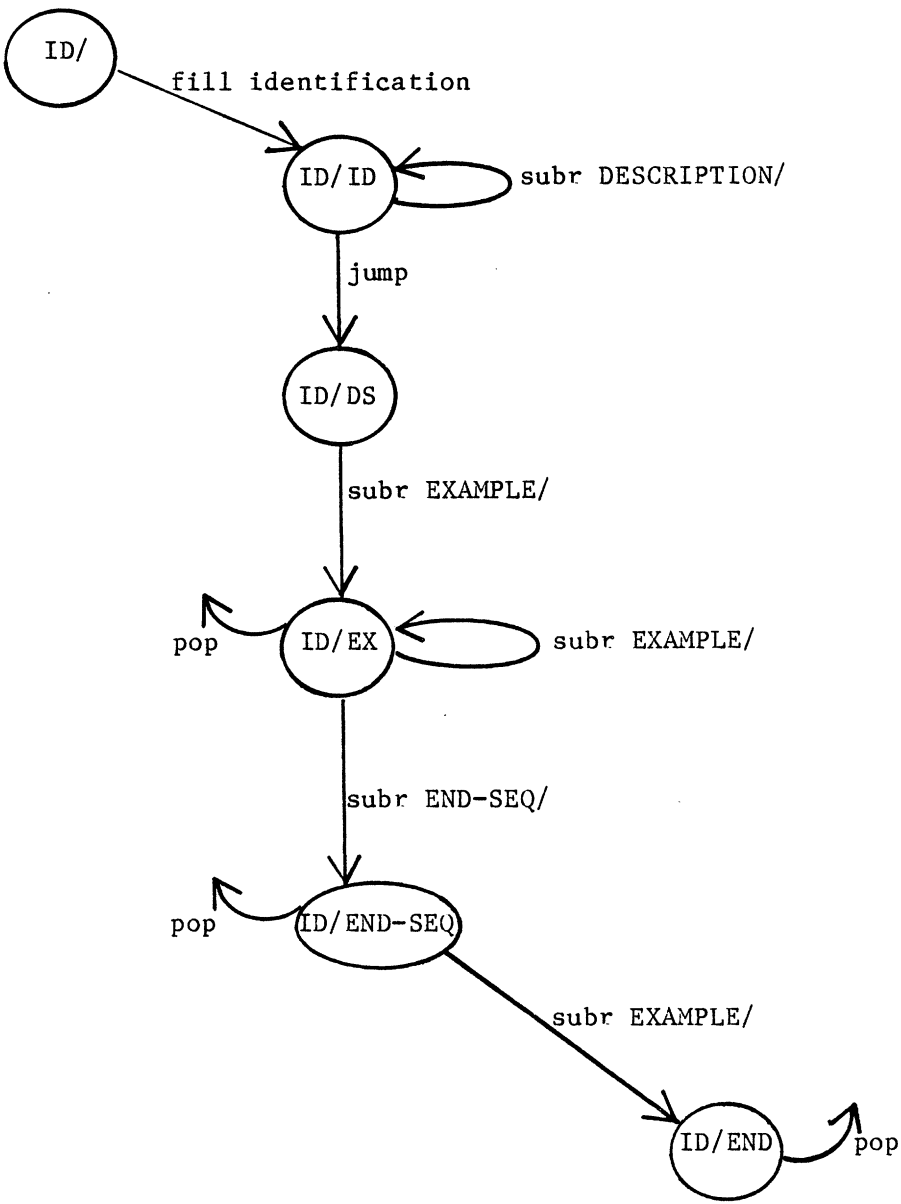
current-discourse (the message so far)

question-type

2.8.4 Graphs Used -

Figures 2.12 - 2.14 show the graphs that represent three of the schemas used in the TEXT system. The graph representing the compare and contrast schema is shown in Section 2.10.

IDENTIFICATION SCHEMA



IDENTIFICATION SCHEMA SUB-GRAPHS

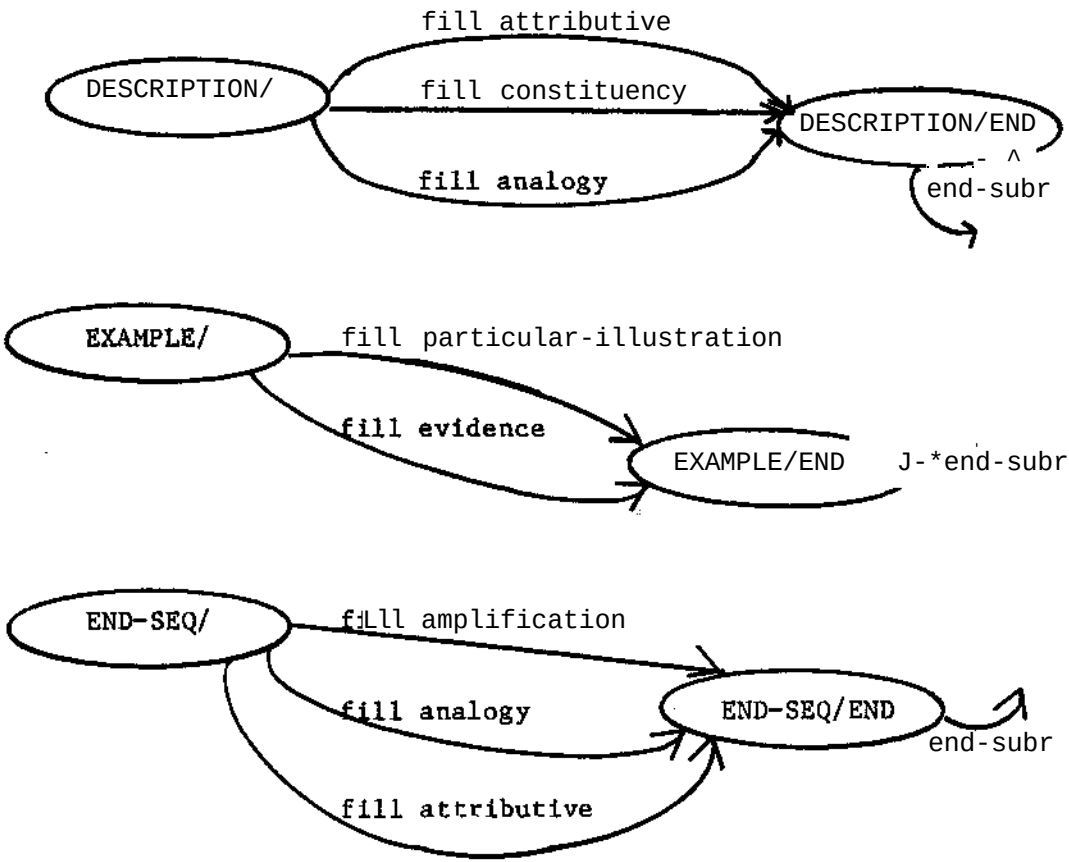
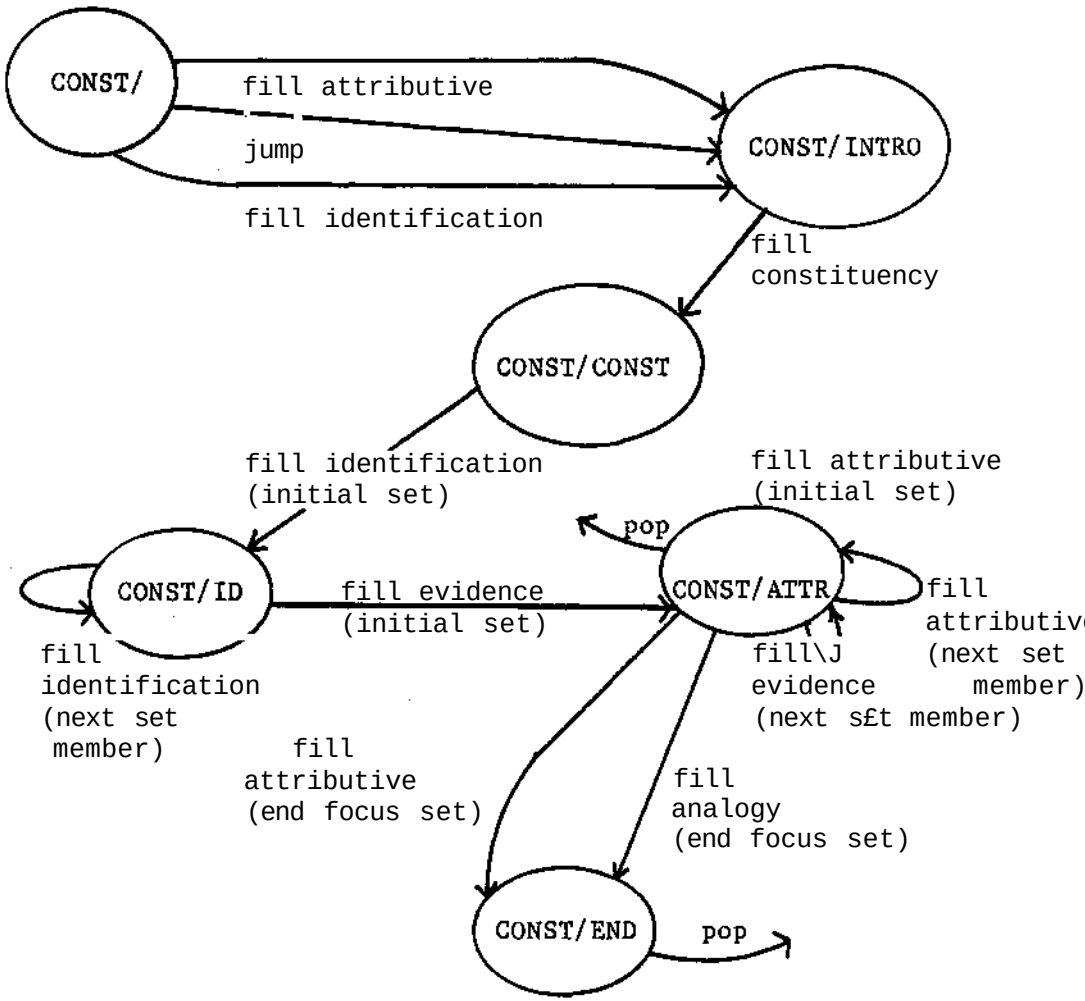


FIGURE 2.12 The identification graph and sub-graphs



(Parenthetical items are directives to focus routines and control th switch to constituents. See Chapter 3.)

FIGURE 2.13 The Constituency Graph

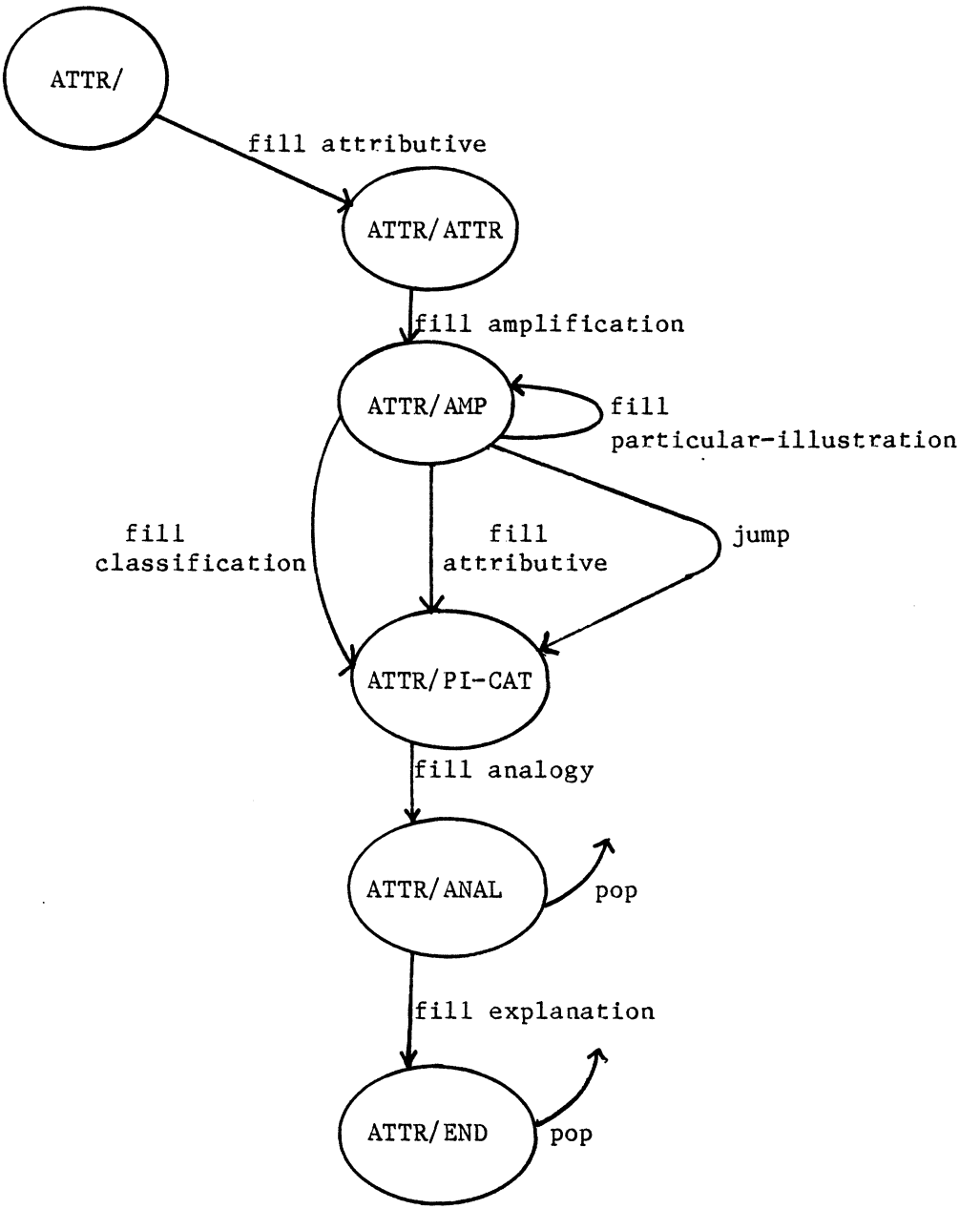


FIGURE 2.14 The Attributive Graph

2.8.5 Traversing The Graph -

The first step in traversing the graph is accessing the start-state of the selected schema. Arcs are taken from the given state until the first fill-arc on each path is reached. If an outgoing arc from the state is a jump-arc, the jump is taken and all arcs from the next state taken until the respective fill-arcs are reached. Similar processing occurs for subr- and push-arcs. The predicates on each path-initial fill-arc are matched against the relevant knowledge pool and any matching propositions are retrieved. After the focus constraints select the most appropriate proposition, the post-actions on the arc whose predicate matched are performed and the successor function applied to the next state.

In order to use subr-arcs and push-arcs, two stacks are maintained, a subr-stack and a push-stack. When a subroutine is taken, the return state is pushed onto the subr stack. When an end-subr is encountered, the first state on the subr-stack is taken. When a push is taken, the states are saved on the push-stack in the same manner. For a push, the registers (which include the focus records) are saved as well.

The use of the stacks is complicated by the fact that all outgoing arcs of a state are traversed until the first fill arc of each successor is encountered, although only one of these arcs is actually taken. If one of the outgoing arcs is a subr or push arc, the return state must be stacked, but if it is not the arc actually taken, then it

is no longer needed on the stack. Further complications arise if more than one outgoing arc of the state is subr or push arc. All return states must be remembered, but only one of them will actually be stacked.*

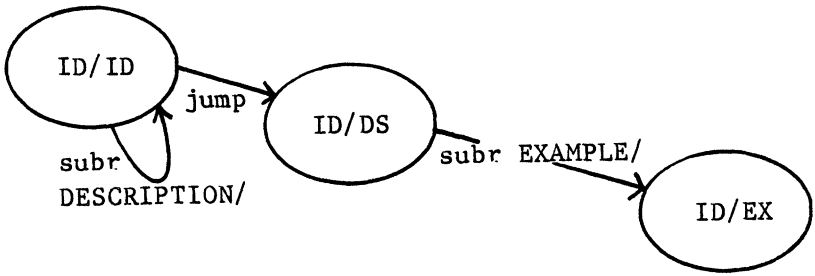
To handle this problem, a tentative-subr and a tentative-push stack are used to stack return states of outgoing arcs. Any stack actions that must be performed in traversing an arc are stored on the tentative stacks under the destination state, the state reached by traversing an arc. If a path of arcs must be taken from a single outgoing arc to reach a fill-arc, stack actions are carried along from the initial state of each arc to its destination state until the final destination state is reached. The tentative stacks look like association lists, with destination states as keys. After the focus constraints determine which arc (or path) should actually be taken, the stack actions associated with the destination state actually reached are retrieved from the tentative stacks and performed on the subr and push stacks. The tentative stacks are then cleared for the next step through the graph. Note that since only states are recorded on the stacks, the graphs must be written so that it is not possible to reach the same state via more than one subr-arc. This can be avoided through the use of jump-arcs.

*The problem of remembering information computed down one path of a non-deterministic mechanism is a common one. In the original ATN implementation, a well-formed substring table was used to handle this problem. In Planner, the finalize predicate is used for this same purpose.

A diagram of graph traversal and effects on the stacks are shown in Figure 2.15. The example described in Section 2.7 above is continued from the point where the identification predicate has matched against the knowledge pool. This corresponds to the second step in the schema and the state id/id in the graph representing the identification schema. At this point in processing, either the attributive, constituency, or analogy arc can be taken next. These are represented by the description/ subroutine. These steps are optional, however, and can be omitted entirely, in which case either the evidence or particular-illustration predicate would be selected. Recall that processing calls for all steps to be tested and the focus constraints select the best match. Only the portion of the graph needed for this step in the schema is reproduced in Figure 2.15.

After an arc (or path) has been selected by the focus constraints, the post-actions on the arc are performed (the focus records are updated and the proposition added to the message) and the proposition is marked as used in the relevant knowledge pool. The successors of the destination-state are then found and the process repeated until the schema is finally exited.

FIGURE 2.15



successors (id/id):

outgoing arc = subr description/

jump id/ds

tentative-subr-stack = nil

1) traverse subr-arc to description/

tentative-subr-stack =

((description/ id/id))

2) traverse fill-arcs (3 of them) to description/end

(carry state associated with each fill-arc initial state (description/) to its destination-state (description/end))

tentative-subr-stack =

((description/end id/id)

(description/end id/id)

(description/end id/id)

(description/ id/id))

3) traverse jump-arc to id/ds (no stack actions)

4) traverse subr-arc to example/

tentative-subr-stack =

((example/ id/ex)

(description/end id/id)

(description/end id/id)

(description/end id/id)

(description/ id/id))

5) traverse fill-arc to example/end (only one succeeds)

tentative-subr-stack =

((example/end id/ex)

(example/ id/ex)

(description/end id/id)

(description/end id/id)

(description/end id/id)

(description/ id/id))

6) 5 matching predicates (as discussed in Section 2.7):

attributive

analogy

constituency (WATER VEHICLE)

constituency (SHIP)

evidence

7) focus constraints dictate that the evidence arc is taken (destination-state = example/end)

subr-stack =

(id/ex)

tentative-subr-stack = nil

8) repeat for successors (example/end)

FIGURE 2.15 Traversing the Graph

2.9 Schema Recursion

As discussed in Section 2,2.1, the rhetorical predicates function recursively; they describe the structure of text at all levels. For example, a single sentence may be used to attribute information to an entity or a longer sequence of text may be used for the same purpose. The analysis of texts was made in order to discover just how predicates are combined to form a longer sequence of text having a specific function. Thus, the resulting schemas describe combinations of predicates which serve the function of a single predicate. For this reason, each schema is associated with a single predicate and is given its name.

Schema recursion is achieved by allowing each predicate in a schema to expand to either a single proposition (e.g. a sentence) or to a schema (e.g. a text sequence). The structure for a text generated from this application of schemas will be a tree structure, with a sub-tree occurring at each point where a predicate has been expanded into a schema. Propositions occur at the leaves of the tree.

Schemas, therefore, are similar in concept to hierarchical plans. Each predicate in the schema is a generation goal which can be achieved either by fulfilling a number of sub-goals (the predicate expands to a schema) or producing a single utterance (the predicate expands to a proposition).

Figure 2.16 illustrates the use of schema recursion. The identification schema is used in response to the question "What is a hobie cat?". The first step the speaker takes is to identify the hobie cat as a class of catamarans (1). To do so, however, he also provides a definition of a catamaran, assuming that his listener knows little about sailing and simply identifying the hobie cat as a catamaran is not adequate for him. The identification predicate expands to the identification schema, where the speaker identifies the catamaran as a sailboat (2) and provides an analogy between the two, which consists of their similarities (3) and differences (4). Note that these two steps are dictated by an analogy schema. After pointing out a catamaran to the listener (4), he pops back to the original identification schema to provide additional information about the hobie-cat (5) and finally, cites two types of hobie-cats, the 16-ft. and the 14-ft. (6).

ID Schema	ID Schema	Analogy Schema
identification ->	identification	
	analogy ->	similarities
		differences
	particular-illustration	
attributive		
particular-illustration		

A hobie cat is a brand of catamaran, which is a kind of sailboat. Catamarans have sails and a mast like other sailboats, but they have two hulls instead of one- That thing over there is a catamaran. Hobie cats have a canvas cockpit connecting the two pontoons and one or two sails. The 16 ft. hobie cat has a main and a jib and the 14 ft. hobie cat has only a main.

FIGURE 2*16 Schema Recursion

Full recursion, such as is illustrated in the above example, is not currently implemented in the TEXT system. In order for the system to be fully recursive, a schema must be written for each rhetorical predicate. Right now, schemas for only four of the predicates (out of a total of 10 predicates) are written. (In the above example, the

analogy schema shown is assumed to correspond to the compare and contrast schema, but this would require more analysis to verify). Writing the extra schemas would require further examination of text samples.

Another, perhaps more interesting side to the recursive use of schemas is the question of when recursion is necessary. Clearly, there are situations where a simple sentence is sufficient for fulfilling a communicative goal, while in others, it may be necessary to provide a more detailed explanation. One test for recursion hinges on an assessment of the user's knowledge. In the above example, the speaker provided a detailed identification of the hobie cat, because he assumed that the listener knew very little about sailing* In order to achieve comprehensive treatment for providing more detailed information a full user-model would have to be developed to determine how much detail is needed for each user at different times.

Another test for recursion hinges on the amount of information available about a given concept in the knowledge pool. No matter how much detail a user needs to understand a concept, it cannot be supplied if nothing more is known about the concept. On the other hand, if a speaker knows a lot about a concept he is discussing, he will probably want to say it unless he's sure the listener already knows about it. Neither user modelling nor assessments of the amount of information which can be talked about have been implemented in the TEXT system.

The machinery for actually performing the recursive push to an associated schema (i.e. entering a new schema and saving the states associated with the old on a stack) is implemented, so that once the extra schemas are written and sufficient tests for providing detailed information developed, full recursion would not be difficult-

There are situations where a full-blown user model is not necessary to determine that recursion is necessary. One such case has been implemented in the TEXT system, where recursion is used in answering a question about the difference between two very different items. In this case, simply asking the question signifies to the system that the user has no idea what these two items are. Since the most appropriate information to include in the answer is about generic classes (see Chapter 4.4), it is the only information provided in the relevant knowledge pool. Therefore, double identification of the two questioned objects is necessary (as was the case in identifying a hobie cat). When a question is asked about two very different items, it triggers the tagging of the super-ordinates of the questioned objects as unknown to the user.

For example, in asking about the difference between a destroyer and a bomb, the questioner indicates that he doesn't understand that one is a vehicle and the other a destructive device, two objects with totally different functions.* During schema filling, the presence of an

*Note that the system does not address itself to the question of why the user thinks they are similar, another possible way of answering the question.

unknown tag indicates that the user needs more detailed information and a recursive push is performed. In (D) below the answer to the question "What's the difference between a destroyer and a bomb?" is shown. Two pushes were taken, both from one identification predicate to the next (proposition 1 to 2 and proposition 3 to 4), resulting in a double identification. Note that since no information other than identificational information is available in the relevant knowledge pool, the early pop is taken.

EXAMPLE D

(differense DESTROYER BOMB)

Schema selected: c&c-identification

proposition selected:
1) (identification DESTROYER SHIP (restrictive ((DRAFT)) (((DRAFT (15 222))))) (non-restrictive TRAVEL-MODE SURFACE))

focus: DESTROYER

proposition selected:
2) (identification SHIP VEHICLE (non-restrictive FUNCTION TRANSPORTATION))

focus: SHIP

proposition selected:
3) (identification BOMB FREE-FALLING (restrictive TARGET-LOCATION SURFACE) (non-restrictive TRAVEL-MEANS GRAVITY-PULL))

focus: BOMB

proposition selected:

4) (identification FREE-FALLING DESTRUCTIVE-DEVICE (non-restrictive FUNCTION LETHAL-KILL))

focus: FREE-FALLING

proposition selected:
5) (inference DESTROYER BOMB very-different-entities)

focus: (DESTROYER BOMB)

Message through dictionary. Entering tactical component

A destroyer is a surface ship with a DRAFT between 15 and 222. A ship is a vehicle. A bomb is a free falling projectile that has a surface target location. A free falling projectile is a lethal destructive device. The bomb and the destroyer, therefore, are very different kinds of entities.

EXAMPLE D

2'10 The Compare And Contrast Schema

The compare and contrast schema, as intimated above, is significantly different in format from the other schemas. It dictates a contrastive structure without specifying which predicates are to be used. Use of predicates varies, depending upon what is being talked about. To achieve this variation, while allowing the schema the same guiding role, the compare and contrast schema makes use of one of the three other schemas as part of the response depending on the semantic information available about the two entities.

Since the type of information included in the relevant knowledge pool for this kind of question was dependent on the conceptual similarity* of the two entities, this classification is available for deciding which schema to use. If the two entities are very close in concept, the attributive schema is used since detailed information about each of the entities is available in the knowledge pool. If the entities are very different in concept, the identification schema is used since the only information available in the knowledge pool is hierarchical classification. For entities in between these two classifications, the constituency schema is used since the class difference in the hierarchy can be discussed as well as some of the entities' attributes.

The compare and contrast schema is shown in Figure 2.17 using the ATN formalism. Note that the three other schemas are used for the contrastive portion of the answer. The first step in the schema is to identify the commonalities of the two entities. During this portion, the two entities are treated as a set and the identification schema is used to describe the set as an entity. This step is optional if no commonalities exist, which is the case for entities which are very different in concept. At this point, a test for conceptual similarity determines the path followed and the schema used. The schema is called twice (once for each entity) and thus, the contrast is set up over a several sentence sequence which corresponds to a single application of

*For another approach to determining similarities, or drawing analogies, see [WINSTON 79].

the embedded schema. An exception to this is the constituency schema which itself includes a description of the class difference and then focuses on each of the two entities in turn. The schema concludes with a direct comparison between the two entities via the inference predicate.

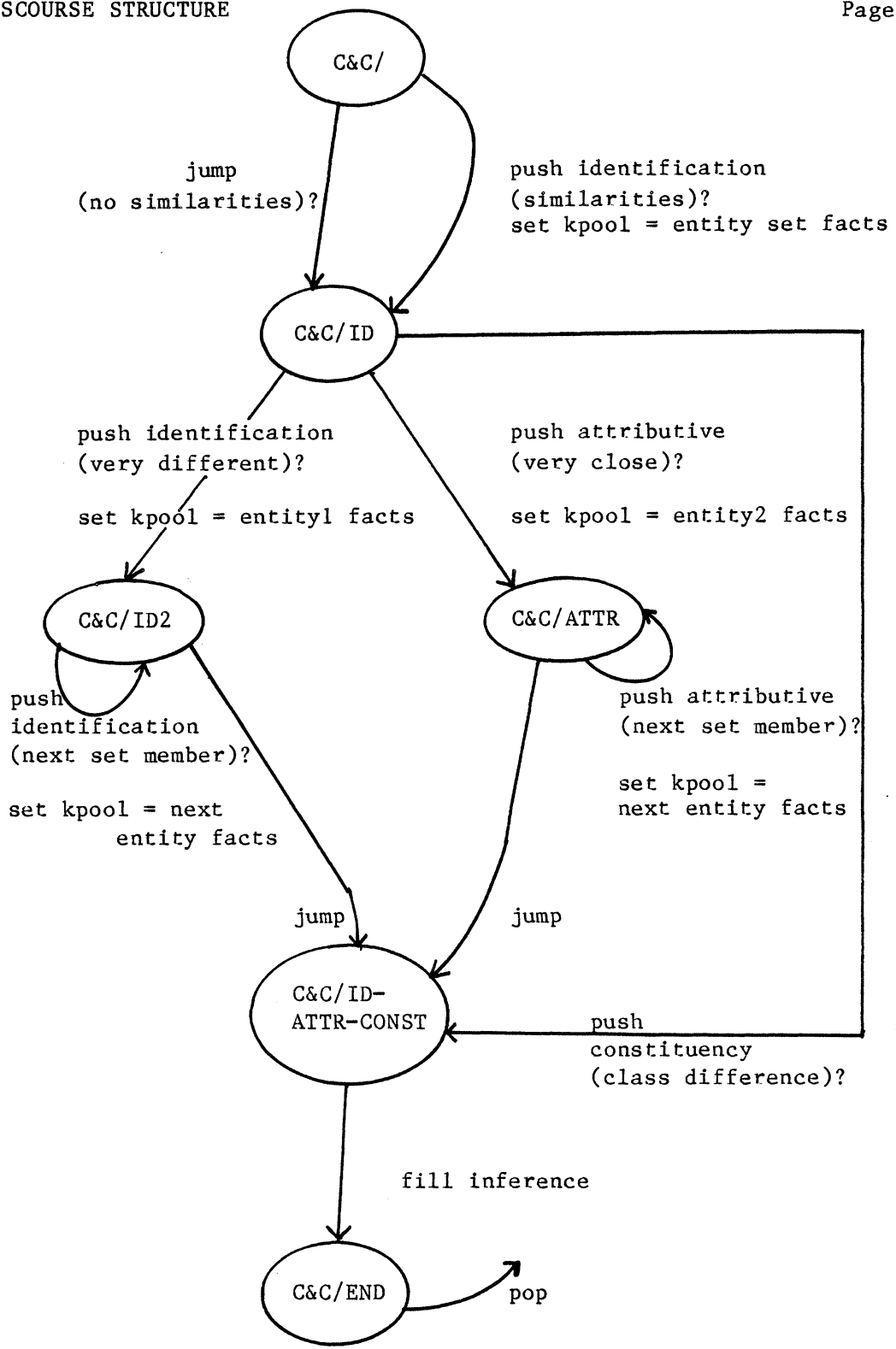


FIGURE 2.17 The Compare and Contrast Graph

Using other schemas within a single schema requires the recursive machinery. Such use is achieved by pushing to that schema. The environment of the compare and contrast schema is saved and the knowledge pool reset for the sub-dialogue. In the case of a push to the attributive and identification schema, the knowledge pool is reset to contain information about one of the entities for each push. When the sub-discourse is complete, a pop returns the process to the original discourse environment.

2.11 Conclusions

Schemas have been used in the text system to model common patterns of text structure. The schemas are loose and embody a number of alternatives, thereby allowing for a good deal of structural variety in text. Moreover, it was shown that schemas are not grammars of text; many experienced and talented writers purposely break norms in order to achieve a striking literary effect. Rather, the schemas describe common patterns of description.

Schemas are used in the TEXT system to guide the generation process. They initiate the process of what to say next; their decisions are monitored by the focusing mechanism which selects between alternatives. Since the schemas were shown to be recursive, describing text structure at many levels, they are much like a hierarchical plan for text. The schemas were implemented through the use of an ATN-like mechanism.

The schemas developed for the TEXT system encode structi suitable for description of static information. Other text types \ require different kinds of schemas and probably different kinds predicates as well. Descriptions of processes involving cause effect, reasoning involved in explanation, and narrative are examples of different text types which will require additi< examination of text to determine commonly used means of description explanation.

3.0 FOCUSING IN DISCOURSE

Focusing is a prevalent phenomenon in all types of naturally occurring discourse. Everyone, consciously or unconsciously, centers their attention on various concepts or objects throughout the process of reading or writing, speaking or listening. In all these modalities, the focusing phenomena occurs at many levels of discourse. For example, we expect a book to concern itself with a single theme or subject; chapters are given headings, indicating that the material included within is related to the given heading; paragraphs are organized around topics; and sentences are related in some way to preceding and succeeding sentences. In conversation, comments such as "Stick to the subject ...", "Going back to what you were saying before ..", or "Let's change the subject ..." all indicate that people are aware that the conversation centers on specific ideas and that there are conventions for changing that focus of attention.

The use of focusing makes for ease of processing on the part of participants in a conversation. When interpreting utterances, knowledge that the discourse is about a particular topic eliminates certain avenues of interpretation from consideration. Grosz [GROSZ 77] discusses this in light of the interpretation of definite referring expressions. She notes that although a word may have multiple meanings, its use in an appropriate context will rarely bring to mind any meaning but the relevant one. Focusing also facilitates the

interpretation of anaphoric, and in particular, pronominal, reference (see [SIDNER 79]). When the coherence provided by focusing is missing from discourse, readers and hearers may have difficulty in determining to what a pronoun refers.

When speaking or writing, the process of focusing constrains the set of possibilities for what to say next. Having decided that he wants to talk about the weather, for example, the speaker need not consider what he could say about yesterday's movie. When a speaker or writer has not decided ahead of time on the specific themes he wants to convey, he will experience difficulty in proceeding. Incoherent text or conversation is often the result in such a situation.

Focusing also influences how something is said. Changing what is focused on may involve marking the move for the hearer by using a different syntactic form. Continuing discussion of the same topic may require pronominalization. The use of marked syntactic structures can highlight new information about a previously mentioned item.

This use of focusing is what makes a sequence of sentences a whole. The fact that a sequence of sentences is about something, makes that sequence connected, coherent, and in some sense, a unit. Intuitively then,a text is a connected, coherent sequence of sentences. In order to generate texts, some account of the use of focusing must be made.

3.1 Computational Theories And Uses Of Focusing

Focusing has been used effectively as a computational tool in the interpretation of discourse by several researchers in artificial intelligence. Although theories about the process of focusing were developed specifically for use in the interpretation of discourse, some of the ideas developed are applicable to the generation of natural language. Some background on previous work in this area is presented before discussing the use of focusing in this research.

3.1.1 Global Focus - Grosz [GROSZ 77] identified the role of focusing in the interpretation of referring expressions in dialogue. In particular, she was concerned with the distinction between two types of focus: global and immediate. Immediate focus refers to how a speaker's center of attention shifts or remains constant over two consecutive sentences. Both the ordering of sentence constituents and the interpretation of sentence fragments are affected by the immediate focus. Global focus, on the other hand, describes the affect of a speaker's center of attention throughout a set of discourse utterances on succeeding utterances. A speaker's global focus encompasses a more general set of objects than his immediate focus. In her work, Grosz concentrated on defining the representation and use of global focus. Thus, Grosz left open the problem of defining and using immediate focus.

Grosz represented global focus by partitioning a subset of the entire knowledge base containing those items in focus at a given time in the discourse from the remaining knowledge base. Determining what is focused on throughout discourse was part of the theory she developed. She distinguished between items that were explicitly focused on, as a result of having been mentioned and those that were implicitly in focus by virtue of their association with mentioned items. Knowing which items are focused makes further interpretation of discourse easier. Considering only a subset of the knowledge base at a given time limits the search for referents of definite noun phrases occurring in the discourse and makes it more likely that the correct referent will be found.

Grosz's representation of focus is, in fact, slightly more complicated than this. In the implementation of a focusing mechanism Grosz termed the subset of the knowledge base that contains items in focus a <u>focus space</u>. A focus space is "open" (i.e. - its contents are currently in focus) if items within it have been recently mentioned. By not bringing items into the focus space until mentioned the efficiency of the search for referents is increased. Items are "implicitly" in focus if they are related to items in an open focus space, but have not yet been mentioned. Mention of one of these items opens the implicit focus space. The old open focus space remains open but is stacked. An open focus space is closed only when a pop to a stacked open focus space occurs. In this case, conversation returns to an earlier topic thereby closing recent discussion. The highly

structured task domain, in which this work was done, was used to guide changes in focus.

3.1.2 Immediate Focus -

Sidner [SIDNER 79] extended Grosz's work with an extensive analysis of immediate focus. She used focus for the disambiguation of definite anaphora and thus, like Grosz, for aiding in the interpretation of discourse. She was able to explain types of anaphora which Grosz did not consider, particularly the use of pronouns. A major result of her work was the specification of detailed algorithms for maintaining and shifting immediate focus.

Tracking immediate focus involves maintaining three pieces of information: the immediate focus of a sentence (represented by the current focus), the elements of a sentence which are potential candidates for a change in focus (represented by a potential focus list), and past immediate foci (represented by a focus stack). Current focus indicates that constituent of a sentence being focused on. The potential focus list records constituents within the sentence that are candidates for a shift in focus. The potential focus list is partially ordered. The focus stack is updated every time a change in focus occurs. When conversation shifts to a member of the previous potential focus list, the old focus is stacked and the current focus becomes the new focus. When conversation returns to an item previously discussed, the stack is popped to yield that item.

Because the concept of focusing is meaningful only in the context of at least two sentences, Sidner's algorithms specify rules for maintaining and shifting focus from one sentence to the next* Briefly, she claims that the speaker has four options:

1. continue talking about the same thing
   (current focus remains the same)

2. switch to talk about an item just introduced
   (current focus becomes a member
   of the previous potential focus list)

3. return to a topic of previous discussion
   (current focus becomes the popped member of the focus stack)

4. talk about an item implicitly related to the current focus
   (general world knowledge is needed to determine that such a switch has been made)

These rules.are only part of an algorithm which is used to determine the referent of an anaphoric expression in the incoming sentence. Tracking the focus of the current sentence is part of the process of determining the referent of an anaphoric expression.

3.2 Focusing And Generation

In previous research in computational linguistics, the use of focusing has been considered as a factor in the comprehension of discourse and in particular, definite anaphora in discourse. This research shows how it can be used as a tool for the generation of discourse. The use of a focusing mechanism provides constraints on the possibilities for what can be said. Global focus constrains the entire knowledge base, producing a subset containing items which can be talked

about. Immediate focus further constrains the subset since after any given utterance a smaller set of choices will be possible. Furthermore, the use of focusing provides a computationally tractable method of producing coherent and cohesive discourse. Use of a focusing mechanism ensures discourse connectivity by ensuring that each proposition* of the discourse is related through its focused argument to the previous discourse.

The following sections describe how to make use of the focusing mechanisms and guidelines developed by both Grosz and Sidner in the generation process. Several problems arise in adapting this work to generation. Since it considered interpretation, there was no need to discriminate between members of the set of legal foci; when more than one possibility for global or immediate focus existed after a given sentence, the next incoming sentence would determine which of the choices was taken. The kinds of choices that must be made in generation, as well as the extensions which must be included in the focusing mechanism to accomodate these decisions, are described in the following sections.

3.2.1 Global Focus And Generation -

In the TEXT system, a relevant knowledge pool, which contains information determined by the system to be relevant to the input question, is constructed for each answer. It is equivalent to Grosz's

*A proposition corresponds to a single sentence in the generated text.

concept of an open focus space. The relevant knowledge pool contains those items which are in focus over the course of an answer. It contains all that can be talked about further. Since it is a subset of the entire knowledge base, it contains a limited amount of information.

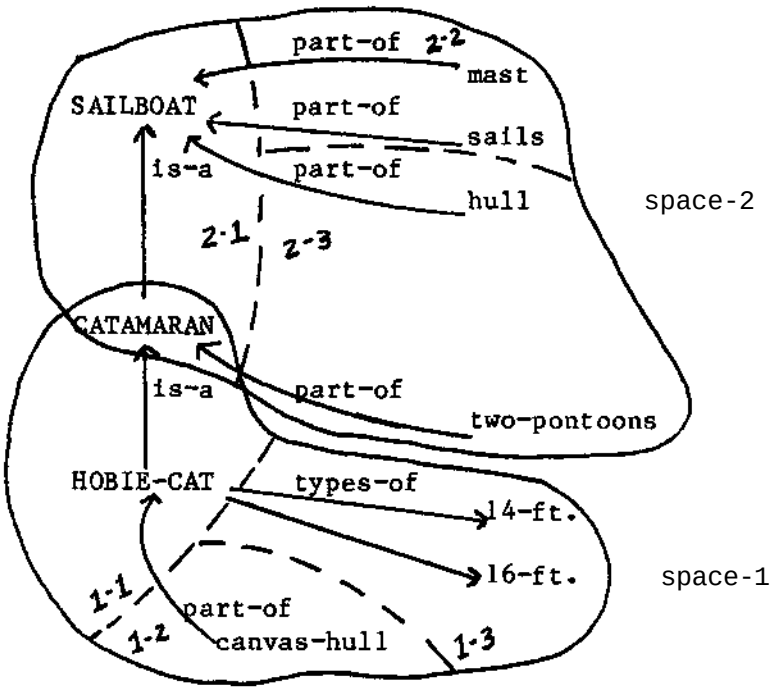
I claim, that in generating discourse, one way that global focus may shift is when a recursive push on a schema is taken (see Chapter 2.9). That is, when it is necessary to provide a more detailed description of a particular concept (in identifying it, attributing information to it, providing an analogy about it, etc.), it is necessary to describe information related to the concept in question and is therefore implicitly in focus. Such information is not part of the open focus space (it has not been mentioned previously), but it is related to the information explicitly in focus. When the push is taken, the focus shifts to this information and it remains in focus for the duration of the new schema. Thus, a new open focus space has been created. Note that the old focus space remains open; due to the nature of recursive pushes on schemas, the text will continue where it left off when the task of providing more detailed information (the push) is completed. When the pop from the sub-schema occurs, the new open focus space is closed and the old open focus space again becomes the active one.

As an example, consider the problem (described in Chapter 2.9) of defining a hobie cat to someone who knows nothing about sailing. Remember that in identifying a hobie cat as a type of catamaran, it was also necessary to define a catamaran for the listener. In this case,

the features of a catamaran are implicitly in focus when talking about the hobie cat. When the push is made to identify the catamaran, the features of the catamaran are focused. They remain in focus throughout the definition of the catamaran (the hobie cat is not discussed now) and when the discussion is finished, mention of the hobie cat brings the old open space back into focus and closes the space containing catamaran features (see Figure 3.1).

It should be noted that this feature of shifting global focus is not currently implemented in the TEXT system, although the design has been worked out. Its implementation is dependent upon the implementation of full recursion (see Chapter 2, Section 2.9), thus requiring the development of a user-model, a major research effort.

1. A hobie cat is a brand of catamaran,
2. which is a kind of sailboat.
3. Catamarans have sails and a mast like other sailboa
4. but they have two hulls instead of one*
5. That's a catamaran there.
6. Hobie cats have a canvas cockpit connecting the tw and one or two sails.
7. The 16 ft. hobie cat has a main and a jib and t hobie cat has only a main.



For sentence 1: Space 1-1 open
Space 2-1, 1-2, 1-3 implicit

For sentence 2: Space 1-1 open but stacked,
Space 2-1 open
Space 2-2, 2-3 implicit

For sentence 3: Space 1-1 open but stacked
Space 2-1, 2-2 open
Space 2-3 implicit

For sentence 4: Space 1-1 open but stacked
Space 2-1, **2-2,** 2-3 open

For sentence 5: same

For sentence 6: Space 1-1, 1-2 open and active
Space-2 closed

For sentence 7: Space 1-1, 1-2, 1-3 open
Space 2 closed

Figure 3.1 Global Focus Shifts

3.2.2 Immediate Focus And Generation -

The previous sections and chapters have shown that the speaker is limited in many ways to what he will say at any given point. He is limited by the goal he is trying to achieve in his current speech act which in the TEXT system is to answer the user's current question. To achieve that goal, he has limited his scope of attention to a set of objects relevant to this goal, as represented by global focus or the relevant knowledge pool. The speaker is also limited by his higher-level plan of how to achieve the goal (the schema). Within these constraints, however, a speaker may still run into the problem of deciding what to say next.

In the TEXT system an immediate focusing mechanism is used to select between these remaining options. It constrains the process of filling the selected schema with propositions from the relevant knowledge pool. Recall that the schemas describe normal patterns of discourse structure and encode a number of alternatives. Hence, they only partially constrain the choice of what to say. During the process of schema filling, more than one proposition may match the next predicate in the schema. This can occur either because 1) alternative predicates appear in the schema and propositions in the relevant knowledge pool match more than one alternative or 2) more than one proposition matches a single predicate. The decision of which proposition is most appropriate is made by the focusing mechanism. It eliminates any propositions whose current focus does not meet the legal restrictions specified by Sidner. That is, the focus of the next

proposition must be either the same as the current focus of the last proposition, a member of the potential focus list of the last proposition, or a member of the focus stack.

The representation of immediate focus and guidelines for shifting and maintaining focus used in the TEXT system follow Sidner. As each proposition is added to an answer*, its focus (termed current focus) and its potential focus list (a partially ordered list of items within the proposition that are potential candidates for a shift in focus) are recorded. A focus stack is maintained throughout the course of an answer and it is updated every time the current focus changes. When the current focus shifts to a member of the potential focus list, the old current focus is stacked. When the current focus shifts to a member of the focus stack (conversation returns to a topic of previous discussion) the focus stack is popped to return to a previous focus.

Although this information sufficed for interpretation, mechanisms are needed for generation which can decide among focus alternatives. In interpretation of discourse, this is not necessary because the choice is dictated by the incoming sentence. For generation, however, the speaker may have to decide between any of these valid foci at any given point. Figure 3.2 shows the choices that a speaker may have to make. The following sections describe how the TEXT system selects between these alternatives.

Choice := CF (new sentence) = CF (last sentence)
vs.
CF (new sentence) E PFL (last sentence)

Choice := CF (new sentence) = CF (last sentence)
vs.
CF (new sentence) E focus-stack

where CF = current focus; PFL = potential focus list

FIGURE 3.2 Choices between valid foci

3.2.3 Current Focus Versus Potential Focus List -

The choice between current focus and items on the potential focus list corresponds to choosing between continuing to talk about the same thing or starting to talk about something introduced in the last sentence.

As an example, consider the following situation. Suppose I want to tell you that John is a new graduate student. Suppose I also want to tell you that new graduate students typically have a rough first semester and I want to tell you a lot of other things about John: what courses he's taking, what he's interested in, where he lives. If I decide to tell you all the other things about John first, when I finally get around to telling you about the first semester of new graduate students, I will somehow have to re-introduce it into the conversation, either by reminding you that John is a new graduate student, by relating it to to rough times, etc. If, on the other hand, I first told you that new graduate students typically have a rough

first semester, I would have no trouble in continuing to tell you the other facts about John. In fact, in continuing talking about John, I will be returning to a topic of previous discussion, a legal focus move. In other words, the current focus of the next sentence will be a member of the focus stack. Note that discussing new graduate students after an ensuing conversation about John is not a legal focus move, since "new graduate students" never became the focus of conversation, but was only a potential focus list member.*

Thus, for reasons of efficiency, when one has the choice of remaining on the same topic or switching to one just introduced, I claim the preference is to switch. If the speaker has something to say about an item just introduced and he does not present it next, he will have to go to the trouble of reintroducing the topic at a later point. In summary,

Choice := CF (next sentence) = CF (last sentence)
vs.
CF (next sentence) E PFL (last sentence)

Preference := CF (next sentence) E PFL (last sentence)

Reason := if preference is not taken, speaker will have to re-introduce PFL-member (last sentence) at a later point.

If this rule is followed, it will have the effect of producing

*In this example, I am ignoring the effect of discourse structure and planning on the choice of what to say next. Clearly these phenomena are intertwined; it is difficult to construct a situation where focus of attention or discourse structure alone determines the resulting discourse.

"topic clusters" within the discourse. In the imagined conversation about John, one instantiation would produce the discourse shown below. Two topic clusters are John as a new graduate student and the courses John is taking.

John is a new graduate student.
New graduate students typically have a rough first semester.
John is taking four courses: intro to programming, graphics, analysis of algorithms, and hardware.
Graphics is the most interesting one.
John lives at graduate towers.

Several consecutive moves to potential focus list members are not a problem. In fact, they occur frequently in written text. In the following example, taken from "Pseudo-silk" in Future Facts [ROSEN 76], focus shifts in every case but one.

1. Finally in November 1973, two Japanese scientists, Seigo Oya and Juzo Takahashi, announced that they had synthesized a fiber which "very much resembles silk."

2. The base for their pseudo-silk is glutamic acid,
   focus = their pseudo-silk (a fiber which "very much resembles silk")

3. one of the 20 amino acids that make up all proteins
   focus = one (glutamic acid)

4. and a chemical long used in the production of monosodium glutamate,
   focus = a chemical (glutamic acid)

5. the controversial seasoning found in many meals,
   focus = the controversial seasoning (monosodium glutamate)

6. ranging from baby food to egg rolls.
   focus = <gap> (meals)

If this rule were applied indefinitely, however, it would result in a never-ending side-tracking onto different topics of conversations. The discourse would be disconcerting and perhaps incoherent« However, TEXT is operating under an assumption that information is being presented in order to achieve a particular goal (i.e. - answer a question). Only a limited amount of information is within the speaker's scope of attention because of its relevance to that goal. Hence only a limited amount of sidetracking can occur.

3.2.4 Current Focus Versus Focus Stack -

The choice between current focus and returning to an item on the focus stack corresponds to the choice between continuing talking about the same thing or returning to a topic* of previous discussion.

Consider an extension of the discourse about John. Suppose I have already told you that John is a new graduate student and that new graduate students have a rough first semester (the first two sentences of Figure 3,.3). Suppose that in addition to telling you the other facts about John (about his courses and where he lives), I also want to tell you that new graduate students are required to maintain a B or above average or they will not be allowed to continue their studies (a fact not mentioned in the last discourse). I have the choice of telling you this immediately after sentence 2 of Figure 3.3 (in which

*Topic is used here loosely to refer to the subject or theme of a discourse. It does not refer to the linguistic notion of topic.

case CF (new sentence) = CF (last sentence) = new graduate students) or of telling you the other facts about John first (in which case CF (new sentence) = focus-stack member = John).

If I should decide to tell you the other facts about John first, I would not run into the same problem of re-introducing a topic since "new graduate students" had been focused on. There is, however, something odd about this choice. Here, the issue of global focus is more important than that of local focus. Having switched the local focus to "new graduate students", I opened a new focus space for discussion. If I switch back to John, I close that focus space (see [GROSZ 77] for a detailed discussion of opening and closing focus spaces), thereby implying that I have finished that topic of conversation. I am implying, therefore, that I have nothing more to say about the topic, when in fact, I do. The preference I claim in this case, is to continue talking about the same thing rather than returning to a topic of previous discussion. Having introduced a topic (which may entail the introduction of other topics), one should say all that needs to be said on that topic before returning to an earlier one.

Choice :== CF (new sentence) = CF (last sentence)
vs.
CF (new sentence) E focus-stack

Preference :== CF (new sentence) = CF (last sentence)

Reason :== to avoid false implication of a finished topic

These two guidelines for changing and maintaining focus during the process of generating language provide an ordering on the three basic legal focus moves that Sidner specifies;

1. change focus to member of previous potential focus list if possible
   CF (new sentence) E PFL (last sentence)

2. maintain focus if possible
   CF (new sentence) = CF (last sentence)

3. return to topic of previous discussion
   CF (new senten.ce) E focus-stack

I have not investigated the problem of incorporating focus moves to items implicitly related to current foci, potential focus list members, or previous foci into this scheme. This remains a topic for future research.

3.2.5 Other Choices -

Even the addition of constraints induced by immediate focusing, however, is not sufficient for ensuring a coherent discourse. Although a speaker may decide to focus on a specific entity, he may want to convey information about several properties of the entity. The guidelines developed so far proscribe no set of actions for this situation. Rather than arbitrarily listing properties of the entity in any order, I claim that a speaker will group together in his discussion properties that are in some way related to each other.

Thus strands of semantic connectivity will occur at more than one level of discourse. An example of this phenomenon is given in discourses (1) and (2) below. In both, the discourse is focusing on a single entity (the balloon), but in (1), properties that must be talked about are presented randomly. In (2), a related set of properties (color) is discussed before the next set (size). (2), as a result, is more connected than (1).

1. The balloon was red and white striped. Because this balloon was designed to carry men, it had to be large. It had a silver circle at the top to reflect heat. In fact, it was larger than any balloon John had ever seen.

2. The balloon was red and white striped. It had a silver circle at the top to reflect heat. Because this balloon was designed to carry men, it had to be large. In fact, it was larger than any balloon John had ever seen.

This type of phenomenon is very common in literary texts. Consider the following example, taken from the introduction to Working [TERKEL 72].* Except for the last sentence, where the current focus changes to "daily humiliations", the focus remains unchanged throughout the paragraph (current focus = this book). An undercurrent of related

*Other literary techniques, which I am ignoring here, such as syntactic parallelism, also serve to make the text a cohesive unit. Again, it is difficult to single out any one of these devices as the main mechanism for achieving cohesiveness.

themes occur from one sentence to the next. Violence, phys violence, spiritual violence, and examples of violence are all rel properties of the book that are described.

> "This book, being about work, is, by its very nature, about violence - to the spirit as well as to the body. It is about ulcers as well as accidents, about shouting matches as well as kicking the dog around. It is, above all (or beneath all), about daily humiliations. To survive the day is triumph enough for the walking wounded among the great many of us."
>
> (p. xiii)
> [TERKEL 72]

This phenomenon manifests itself as links between the poten focus lists of consecutive propositions in discourse. Consider focus records for the first two sentences of the Working paragraph:

1. CF = this book
   PFL = violence
   {spirit; body}
   being about work
   is about

2. CF = this book
   PFL = {ulcers; accidents
   shouting matches; fist fights
   nervous breakdowns; kicking the dog around}
   is about

In this example the first item of the potential focus list (PFL) of sentence #2 (a list) provides instances of violence and is thus related to the first item in sentence #l's PFL. The elements of the list also exemplify the spiritual/physical dichotomy of violence and are thus related to the second item in sentence #l's PFL. Note furthermore that the PFL links are implicit links; ulcers and accidents are sub-types of violence, spiritual and physical. Although the current focus of a sentence is often a definite reference (pronominal or otherwise) to a previously mentioned item, definite reference across potential focus lists rarely occurs.

These potential focus links result in a layering of foci that corresponds to the speaker's global focus. More than one thing is focused on at a time (global focus) and one of them is distinguished as immediate focus. In the generation process, this phenomenon is accounted for by further constraining the choice of what to talk about next to the proposition with the greatest number of links to the previous potential focus list. This constraint ensures that the text will maintain the global focus of the speaker when possible. If application of the guidelines discussed above does not narrow down the possibilities to a single proposition, links between potential focus lists are examined to select the single proposition with the most links to the previous discourse (i.e. that proposition containing the greatest number of links to elements already mentioned). The ordering of focus maintaining and shifting rules when updated to include this constraint becomes:

1. shift focus to member of previous PFL
   CF (new sentence) E PFL (last sentence)

2. maintain focus
   CF (new sentence) = CF (last sentence)

3. return to topic of previous discussion
   CF (new sentence) E focus-stack

4. select proposition with greatest number of implicit links to previous potential focus list
   PFL (new sentence) related-to PFL (last sentence)

3.2.6 A Focus Algorithm For Generation -

Before describing exactly how these guidelines for maintaining and shifting focus are incorporated into an algorithm that can determine what to say next, the assignment of focus and potential focus list to a proposition must be discussed. In TEXT, the assignment of focus involves a process of give and take between what the focus could be and what the guidelines about focus maintenance claim as preference. Initially, a default focus is assigned to a proposition. This focus can be overridden, however, if another item within the proposition allows for the application of one of the higher rules on the ordered list of guidelines.

3.2.7 Selecting A Default Focus -

Selecting a default focus is a simple look-up procedure. A single argument of each predicate is singled out as the one most likely to be focused on. This information is stored in a table and the entry for the given predicate accessed when needed. Use of a default focus

implies that a predicating act has a marked and unmarked syntax associated with it, the unmarked dictated by the default focus. This does not seem unlikely. Consider the attributive predicate. In its usual use, it attributes features to an entity or event. The unmarked use assumes an entity has been focused on: the entity is being talked about and some of its features are being described (see Sentence 1 below). The opposite case, of associating talked-about-features with a different entity is less usual (see Sentence 2 below).

1. The chimpanzee has fine control over finger use.
2. Fine control over finger use is also common to the chimpanzee.

3.2.8 Overriding The Default Focus -

The default focus of a proposition is overridden if taking a different predicate argument as focus will allow the application of a more preferable guideline for focus movement. For example, if the default focus of a proposition is the same as the current focus of the last proposition, guideline #2 would apply (CF (new sentence) = CF (last sentence)). If, however, another predicate argument of the proposition is a member of the previous proposition's potential focus list, that argument is selected as the proposition's focus, since it allows for the application of guideline #1 (CF (new sentence) E PFL (last sentence)). The assignment of focus to propositions is made when a proposition is selected which ties' in most closely with the preceding

discourse according to the focus constraint guidelines.

Although the default focus can be overridden, it is useful since it provides some indication of the usual way of presenting information. It is necessary for determining the focused argument of discourse initial propositions where no information exists to override it. Since it indicates the most likely case, it can result in savings in processing time within each guideline application. If, for example, one proposition of a set of possible next propositions has an argument that is a member of the previous potential focus list, it will be selected by application of guideline #1. If that argument is the proposition's default focus, the proposition will be selected after the default focus of each proposition is checked for membership in the previous potential focus list (one test per proposition). If default focus is not represented, the proposition will only be selected after each argument of each proposition is checked for membership in the previous potential focus list (many tests per proposition).

Moreover, if following a single guideline of the focus constraints would allow more than one proposition argument to qualify for focus, the default focus indicates which of these to use. If, for instance, a proposition has several arguments which occur in the previous potential focus list, the default focus is selected as the argument to be focused on.

3.2.9 The Focus Algorithm -

The algorithm for using focus constraints in the selection propositions is given below. This algorithm does not specify exac how a schema is selected or filled and is, therefore, not a compl algorithm for the strategic component (see Chapter 2 for details these processes).

I. Select schema

II. Initialization of focus records
    GF (global focus) = argument of goal
      (e.g. - for (definition AIRCRAFT-CARRIER),
       GF = AIRCRAFT-CARRIER
       for (differense OCEAN-ESCORT CRUISER),
       GF = {OCEAN-ESCORT CRUISER} (the set)
    CF (current focus) = GF
    PFL (potential focus list) = nil

III. For each entry in schema:
    1. Select all propositions in relevant knowledge pool that match the predicate (see Chapter 2.6 for a description of predicate semantics)

       If schema allows for a choice of predicates, select all possible propositions for each predicate

    2. If options exist (i.e. - more than one proposition matched)
       Use immediate focus constraints:
       A. Select proposition(s) with default-focus E PFL
          Set proposition-focus = default-focus
       B. If none exist, select proposition(s) with other argument-entry E PFL
          Set proposition-focus = other argument-entry
       C. If none exist, select proposition(s) with default-focus = CF
          Set proposition-focus = CF
       D. If none exist, select proposition(s) with other argument-entry = CF
          Set proposition-focus = other argument entry
       E. If none exist, select proposition(s) with default-focus E focus-stack

Set proposition-focus = default-focus
F. If none exist, select proposition(s) with other argument-entry E focus-stack
Set proposition-focus = other argument entry
G. If none exist, set proposition-focus of each with proposition = default-focus

3. If options exist (i.e. - more than one proposition remains after application of immediate focus constraints) Use potential focus list links
A. Select that proposition with greatest number of links to PFL

4. Record predicate, proposition pair
A. Add to message
1. Add CF, PFL, focus-stack for last proposition to message
*2** Add new proposition to message
B. Update focus records
1. If proposition-focus E focus-stack
Pop focus-stack
CF = proposition-focus
2. If proposition-focus ~= CF
Stack CF on focus-stack
CF = proposition-focus
3. Else no change to CF and focus-stack
4. Set PFL:
A. member #1 = default-theme* of proposition if proposition-focus ~= default-theme
Else = default-focus
B. last-member = predicate (corresponding to sentence verb)
C. Other PFL members = arbitrary listing of other predicate argument in proposition

Note that if no suitable focus for the next proposition is found (2G), the proposition's default focus is used and the proposition a

*As is the case for focus, a table of proposition arguments

to the message. This type of conversational move is the equivalent of a total shift in focus. Since the strategic component maintains a focus record, the tactical component could use this information to select an appropriate syntactic cue to signal this kind of abrupt shift.

Note also that the potential focus list of each proposition is only partially ordered. Its first member is the default theme and last member, the predicate, or what will eventually be the verb of the sentence. Other entries are set in arbitrary order.

The focus algorithm makes a breadth-first search of all possible next propositions. In other words, each possible proposition is retrieved and then the focus constraints are applied. Another approach would be a depth-first search of the possibilities, retrieving only propositions that have an argument which meets the first focus preference and if that fails, retrieving propositions which meet the second focus preference, etc. To determine which propositions meet a focus preference, however, it is necessary to retrieve all propositions and examine their arguments, making the depth-first search a more expensive alternative.

3.2.10 Use Of Focus Sets -

Sidner notes that a discourse need not always focus on a single central concept. A speaker may decide to talk about several concepts at once and yet, the resulting discourse is still coherent. She terms

this type of phenomenon "co-present foci". She gives the following discourse as an example of the use of co-present foci:

1. I have 2 dogs.
2. The one is a poodle;
3. the other is a cocker spaniel.
4. The poodle has some weird habits.
5. He eats plastic flowers and likes to sleep in a paper bag.
6. It's a real problem keeping him away from the flowers.
7. My cocker is pretty normal,
8. and he's a good watchdog.
9. I like having them as pets.

In this discourse, a set of two elements is introduced as focus in sentence #1. Each element of the set is specified in 2 and 3 by "the one .... the other" construction (Sidner, in fact, relies heavily on this type of construction to identify the use of co-present foci). The discourse then proceeds to focus on each element of the set in turn.

Sidner notes that the use of co-present foci in discourse is a highly regulated phenomenon. For this reason, focusing on more than one central concept does not result in an incoherent discourse. She says:

> "Co-present foci reflect a special kind of structure that occurs in discourse. Several elements are introduced. When continuing discussion of one of the elements extends the discourse, the focus moves to that element. When that discussion is complete, the focus cannot simply move onto any other thing the speaker wants to mention. The discussion should return to the other elements, and those elements discussed. However, the discussion of one element for an extended part of the discourse may involve introduction and consideration of other elements. The real constraint in the foregoing analysis is that discussion should eventually <u>return</u> to the other elements via co-present foci. When it does not,

> the hearer is left to wonder why co-presence was used in the first place."
>
> [SIDNER 79], p. 195.

Although Sidner describes the restrictions on how co-present foci can occur, she does not describe the reasons for its use or for the focus moves to elements of the focused set. Again, for interpretation of discourse, the use of co-present foci is given by the incoming discourse, and there is no need to decide when its use is appropriate. Generation of discourse, however, requires that these kind of decisions be made.

Decisions to use co-present foci rest in part on the rhetorical techniques used in discourse and thus, the discourse structure (e.g. the decision to define an object in terms of its sub-classes) and in part on the discourse goal (e.g. the decision to answer a question about the difference between two objects). In the first case, definition of a concept in terms of its sub-classes suggests the use of the constituency schema, a particular structure for discourse. Use of the constituency schema implies focusing on the questioned object, followed by the introduction of its sub-classes and extended discussion of each of these in turn. In this case, the structure of the discourse forces the use of co-present foci and the changes in focus to set members.

In the second case, the discourse purpose is to provide a description of the differences between the two objects. Associated with this discourse purpose are the rhetorical techniques encoded in the compare and contrast schema. Although the exact structure dictated by this schema varies depending on the type of information available in the relevant knowledge pool (see Chapter 2.10), its basic outline is a discussion of the similarities between the two objects, followed by a discussion of their differences. Thus, the discourse will first center on the two objects and their common attributes; focus will then switch to the questioned objects in turn. Again, the structure of the discourse forces the use of co-present foci and the changes in focus to set members.

In the TEXT system, schemas control the introduction of focus sets and changes in focus to their elements. Arc actions on the schemas (see Chapter 2.8 for a discussion of the ATN nature of schemas) can force selection of a set as focus and dictate moves to their elements. The focus algorithm continues as usual, with the exception that it allows decisions involving that focus set to override its own. Once discussion involving the focus set is over, the focus algorithm proceeds normally.

3.3 Focus And Syntactic Structures

3.3.1 Linguistic Background -

Phenomena similar to what is called "immediate focus" have been studied by a number of linguists. Terminology and definitions for these vary widely; some of the names which have emerged include topic/comment, presupposition and focus, theme/rheme, and given/new. It should be noted that a major difference between these concepts and focus, as discussed in this chapter, is that focusing describes an active process on the part of speaker and listener. The item in focus is that item on which the speaker is currently centering his attention. These linguistic concepts describe a distinction between functional roles elements play in a sentence. A brief description of each of these linguistic concepts follows.

Topic/comment articulation is often used to describe the distinction between what the speaker is talking about (topic) and what he has to say about that topic (comment). Definitions of topic/comment for a sentence usually do not depend upon previous context, although some linguists (in particular, [SGALL et al. 73]) provide different definitions of the distinction for sentences that contain a link to previous discourse and for those that do not. Others who have discussed topic/comment articulation include Lyons [LYONS 68] and Reinhart [REINHART 81].

Presupposition has been used to describe the difference between information which a sentence structure indicates is assumed as true by the speaker. Presupposition has a very precise definition for formulations which consider meaning the equivalent of truth and has been analyzed by many (e.g. [WEISCHEDEL 75], [KEENAN 71]). It refers to all that must hold in order for a sentence to be true.

Focus labels information in the sentence which carries the import of the message. The focus of a sentence is usually determined by the position where phonological stress occurs (see [CHOMSKY 71], [QUIRK & GREENBAUM 73], [HALLIDAY 67]).

The given/new distinction distinguishes between information that is assumed by the speaker to be derivable from context (given) — where context may mean either the preceding discourse or shared world knowledge — and information that cannot be (new). The given/new distinction has been discussed by Halliday [HALLIDAY 67], Prince [PRINCE 79], and Chafe [CHAFE 76].

Theme/rheme is a distinction used in work by the Prague School of linguists (see [FIRBAS 66], [FIRBAS 74]), They postulate that the sentence is divided into a theme — elements providing common ground for the conversants — and a rheme — elements which function in conveying the information to be imparted. In sentences containing elements which are contextually dependent, the contextually dependent elements always function as theme. Thus, the Prague School version is close to the given/new distinction with the exception that a sentence

always contains a theme, while it need not always contain given information. Halliday also discusses theme/rheme, but he concludes that they are dependent on sentence order; theme is always the first syntactic constituent of a sentence and rheme is its remainder [HALLIDAY 67].

These paragraphs, provide only a brief overview of these various distinctions. Fuller discussions are provided in [PRINCE 79] and [CHAFE 76] which also give overviews of the conflicting descriptions and definitions of these concepts. What is important to this thesis is that each of these concepts, at one time or another, has been associated with the selection of various syntactic structures. For example, it has been suggested that focus, new information, and rheme usually occur towards the end of a sentence ([HALLIDAY 67], [FIRBAS 74], [LYONS 68], [SGALL et. al. 73]). In order to place this information in its proper position in the sentence, structures other than the unmarked active sentence may be required (for example, the passive). Structures such as it-extraposition, there-insertion, topicalization, and left-dislocation* have been shown to function

in the introduction of new information into discourse ([SIDNER 79], [PRINCE 79]), often with the assumption that it will be

*Some examples of these constructions are:
1. It was Sam who left the door open. (it-extraposition)
2. There are 3 blocks on the table. (there-insertion)
3. Sam, I like him. (left-dislocation)
4. Sam I like. (topicalization)

talked about for a period of time. Pronominalization is another linguistic device associated with these distinctions (see [AKMAJIAN 73], [SIDNER 79]).

3.3.2 Passing Focus Information To The Tactical Component -

Since focus information has been used to constrain the selection of propositions, a record containing each proposition's focus and its potential focus list is available for the tactical component to use when determining the specific syntactic structures and linguistic devices that should be used in the answer. The tactical component can examine this information to determine how a proposition is related to previous discourse: whether the focus has shifted to a new topic, whether a return to a previous topic was made, or whether, in extreme cases, a total shift in topic has been made and the proposition is totally unrelated to what came before. Some of the uses that can be made of this information and the linguistic effects that can be achieved are described in this section.

Pronominalization is a linguistic device that has long been linked with concepts such as focus of attention ([SIDNER 79], [AKMAJIAN 73], [MCDONALD 80]). Sidner uses its presence to aid in determining focus. If an entity remains in focus over a sequence of sentences, references to it can be pronominalized.* The following two sentences illustrate this:

John was late coming home.
He got caught in a traffic jam.

In the TEXT system, focus information is used in some limited situations to test whether pronominalization can be used. An example of an answer where pronominalization was selected is shown in (A) below. In the first sentence of the answer, the ship is being focused on and reference to it in the following sentence can therefore be pronominalized. Note that definite reference such as "the ship" would also be appropriate, but in the TEXT system, pronominalization is selected wherever it is determined possible.

A) (definition SHIP)

A ship is a water-going vehicle that travels on the surface. Its surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. Other DB attributes of the ship include MAXIMUM_SPEED, PROPULSION, FUEL (FUEL_CAPACITY and FUEL_TYPE), DIMENSIONS, SPEED_DEPENDENT_RANGE and OFFICIAL_NAME. The DOWNES, for example, has MAXIMUM_SPEED of 29, PROPULSION of STMTURGRD, FUEL of 810 (FUEL_CAPACITY) and BNKR (FUEL_TYPE), DIMENSIONS of 25 (DRAFT), 46 (BEAM), and 438 (LENGTH) and SPEED_DEPENDENT_RANGE of 4200 (ECONOMIC_RANGE) and 2200 (ENDURANCE_RANGE).

Focus information has also been shown to affect the use of different syntactic structures, as discussed in Section 2.3.1. Depending upon which constituent of the sentence is in focus, the passive construction might be selected over the active construction. There-insertion and it-extraposition can be used to introduce items as the focus of continued discussion.

---

*McDonald shows that some additional tests for pronominalization must be made before a pronoun can be used. See [MCDONALD 80] for a discussion of subsequent reference.

These constructions provide a mechanism for reordering the constituents of a sentence based on thematic information. The passive construction is used when the logical subject of the sentence is not in focus. Passivizing the sentence moves the focused constituent (logical object or beneficiary) to the surface subject position, thus allowing the focused information to appear as sentential subject. Passivization does not apply when the logical subject of the sentence is focused. Sentences with verb "be" cannot be passivized. In such cases, there-insertion must be used to achieve thematic reordering of constituents.

Currently in the TEXT system, focus information is used to discriminate between use of the passive and active construction. Relations in the ONR database are binary and can be described from the point of view of either entity. In the ONR database, weapons are associated through the relation "carry" with different vehicles. When answering a question about missiles, a weapon, the passive construction is used to describe the relation that holds between the missile and various vehicles, since the missile is in focus. When answering a question about the ECHO II, a type of submarine, the active construction is used in order to attribute information to the "ECHO II" and not to the weapons. These examples are shown in (B) and (C) below.

B) (differense MISSILE TORPEDOE)

The torpedo and the missile are self-propelled guided projectiles. The guided projectile's propulsion capabilities are provided by the DB attributes under SPEED_INDICES (for example, MAXIMUM_SPEED) and FUSE_TYPE. The guided projectile has DB attributes TIME_TO_TARGET__UNITS, HORZ__UNITS and NAME. The missile has a target location in the air or on the earth's surface. The missile's target location is indicated by the DB attribute DESCRIPTION and its flight capabilities are provided by the DB attribute ALTITUDE. Other DB attributes of the missile include PROBABILITY_OF_KILL, SPEED, ALTITUDE, LETHAL_RADIUS__UNITS and TIME_TO_TARGET__UNITS. Missiles are carried by water-going vehicles and aircraft. The torpedo has an underwater target location. Its underwater capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUM_OPERATING_DEPTH). Other DB attributes of the torpedo include FUSE_TYPE, MAXIMUM_DEPTH, ACCURACY__UNITS, HORZ_RANGE__UNITS, and TIME_TO_TARGET__UNITS. Torpedoes are carried by water-going vehicles. The torpedo and the missile, therefore, have the same travel means, although they have different target locations, reflected in the database by the torpedo's attribute DEPTH and the missile's attributes ALTITUDE and DESCRIPTION.

C) (information ECHO-II-SUBMARINE)

Echo IIs have a PROPULSION_TYPE of NUCL and a FLAG of RDRD. All echo IIs in the ONR database have REMARKS of 0, FUEL_TYPE of NUCL, IRCS of 0, MAXIMUM_OPERATING_DEPTH of 700 and NORMAL_OPERATING_DEPTH of 100. There are no sub-classes of echo II in the ONR database. Echo IIs carry 16 torpedoes, between 16 and 99 missiles and 0 guns. A submarine is classified as an echo II if its CLASS is ECHO II.

The use of there-insertion by the TEXT system is shown below in (D) in the answer generated to the question "What is a guided projectile?". Although the construction is associated with the constituency predicate in the constituency schema (i.e. a decision to use this construction is not based on a test of focus information), the constituency schema is used to introduce a set as the focus and then forces a shift in focus to each of the set members. Use of there-insertion in this situation is one way to introduce the set of

sub-classes as focus into the discourse.

D) (definition GUIDED)

A guided projectile is a projectile that is self-propelled. There are 12 types of guided projectiles in the ONR database: torpedoes and missiles. The missile has a target location in the air or on the earth's surface. The torpedo has an underwater target location. The missile's target location is indicated by the DB attribute DESCRIPTION and the missile's flight capabilities are provided by the DB attribute ALTITUDE. The torpedo's underwater capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUMjOPERATINGJDEPTH)• The guided projectile has DB attributes TIMEJTOJTARGET__UNITS, HORZ_RANGE__UNITS and NAME.

Several other linguistic effects, which were not implemented in the TEXT system, can be achieved through the use of focus information. In an examination of texts and transcripts (see Chapter 2 for a discussion of this analysis), I found that parallel sentence structure was often used when focus remained the same from one sentence to the next. Parallel structure can be used to increase the cohesiveness of the text through syntactic devices when the semantic cohesiveness is not as rich. The first paragraph of the Introduction to Working [TERKEL 72] illustrates the use of parallel sentence structure for consecutive sentences with the same focus (also see [HOBBS 78] on parallel structure):

> "This book ... is about violence - to the spirit as well as to the body. It is about ulcers, as well as accidents, ..."

When focus shifts in a proposition to an item just introduced into conversation in the last proposition, subordinate sentence structure can be used to combine the two propositions into a single complex sentence. A sample use of subordinate sentence structure for this

purpose is shown below. In the main sentence, "the happy few" are introduced as focus using there-insertion. In the subordinate sentence, focus is shifted to elements of the happy few (the Indiana stonemason and the Chicago piano tuner).

> "There are, of course, the happy few who find a savor in their daily job: the Indiana stonemason, who looks upon his work and sees that it is good; the Chicago piano tuner, who seeks and finds the sound that delights; ..."

Focus information can also signal the use of textual connectives or semantic markers. When there are no referential links from one sentence to the next (i.e. a proposition was selected which did not meet the legal criteria specified by Sidner), a textual connective can be used to provide the needed link from one sentence to the next. Reinhart [REINHART 81] discusses this phenomenon and cites an example from Barth [BARTH 79], which is reproduced below. The only linking device between the clauses is the semantic marker at the same moment. Without it, the two clauses would be semantically unrelated.

> "... when I reentered my office the clock in the tower of the Municipal Building was just striking two, and as if by a prearranged signal, at the same moment the raucous voice of a steam calliope came whistling in off the river ..."

3.4 Extensions

In the TEXT system, tests for pronominalization and syntactic constructions are made in the dictionary. In the dictionary, predicates are mapped into verbs and their arguments into the sentence

constituents; the output is an underlying syntactic structure for the sentence. If a test on the focus information indicates that a sentence should be passivized, the attribute-value pair "voice == passive" is added to the underlying structure (see Section 4.6 for a description of the grammar and underlying structure representation). In the dictionary, tokens in the message are replaced by their associated English words. During this replacement stage, a check for pronominalization is made and if the test is satisfied, a pronoun is used.

Eventually these tests will be incorporated into the grammar, where they belong. The functional grammar used in the TEXT system was selected because of the ability to encode tests on concepts such as focus or topic/comment directly into the grammar. Steven Bossie will be working for his master's thesis on incorporating the tests which are currently implemented as part of the dictionary plus some additional more sophisticated tests on focus into the grammar.

3.5 Conclusions

In the TEXT system, focus of attention has been used to constrain the choice of what to say next. The choices are constrained in two ways: 1) by global focus and 2) by immediate focus. When deciding what to say next, the system need only consider information that is relevant to the question currently being asked, as opposed to all information in the knowledge base. This information is contained in the "relevant knowledge pool". After every generated utterance, the

system is further limited by a set of immediate focus constraints to what it can say next.

The specification of legal focus moves extends work on immediate focus for interpretation [Sidner 79] to the generation process. In particular, guidelines for ordered application of these moves were developed for those situations where the constraints did not narrow down the set of choices to a single possibility. The use of focusing was also extended to allow for strands of semantic connectivity to occur at more than one level of the discourse. Using global and immediate focus proves to provide a computationally tractable approach to the problem of deciding what to say next. Furthermore, it guarantees the production of a semantically cohesive text.

Since focus information was used in producing the text message, it is available for the tactical component to use in making decisions about relatively sophisticated linguistic devices. In the TEXT system, tests for pronorainalization and varying syntactic structures (active, passive, and there-insertion) on the basis of focus information have been implemented. Other linguistic devices, such as parallel sentence structure, subordinate sentence structure, and semantic markers were shown to be related to focusing, but were not actually implemented in the TEXT tactical component.

4.0 TEXT SYSTEM IMPLEMENTATION

The TEXT system was implemented in CMU lisp (an extension of Franz Lisp) on a VAX 11/780. A portion of an Office of Naval Research (ONR) database was used to test the system. The ONR database portion used for TEXT contains information about VEHICLES and DESTRUCTIVE DEVICES. The ONR database was selected for TEXT in part because of its availability (it had been in use previously in a research project jointly with the Wharton School of the University of Pennsylvania) and in part because of its complex structure. Even using only a portion of the database provided a domain complex enough that users may be confused about its organization. The set of objects the portion contained was large enough and the information associated with them varied enough to allow for an interesting domain.

As discussed in Chapter 1, TEXT accepts three kinds of questions as input. These are:

1. What is a <e>?
2. What do you know about <e>?
3. What is the difference between <e1> and <e2>?

where <e>, <e1>, and <e2> represent any entity in the database. Since the TEXT system does not include a facility for interpreting English questions, the user must phrase his questions in the functional notation shown below which corresponds to the three classes of

questions.

1. (definition <e>))
2. (information <e>)
3. (differense <e1> <e2>)*

Note that the system only handles questions about objects in the database. Although the system can include information about relations when relevant to a question about a particular object, it can not answer questions about relations themselves. The decision not to include relations was made in part out of an effort to reasonably limit the size of the implementation, in part because of the impoverished use of relations in the ONR database, and in part because the database is static and not dynamic.

Four schema types were developed and used for the TEXT generation system. Each schema was formulated on the basis of an analysis of naturally occurring expository text. As mentioned earlier, this analysis revealed that certain patterns of usage of communicative techniques were found to recur frequently across a variety of texts.

4.1 Resources Used

The TEXT system source code occupies a total of 1176 K of memory with the following breakdown:

1. Knowledge base and accessing functions (not including database and database interface functions): 442K

2. Strategic component: 573K

3. Tactical component: 145K

The system, including the knowledge base, was loaded in entirety into memory for use of the TEXT system. Only the database remains on disk. No space problems were encountered during implementation with one exception: the particular Lisp implementation available does not allow for resetting the size of the recursive name stack. This meant that certain functions which were originally written recursively had to be rewritten iteratively since the name stack was not large enough to handle them.

Processing speed is another question altogether. Currently the response time of the TEXT system is far from being acceptable for practical use. The bulk of the processing time, however, is used by the tactical component. Since it was not the focal point of this dissertation, no major effort was made to speed up this component. To answer a typical question posed to the TEXT system, the strategic component (including dictionary interface) uses 3290 CPU seconds, an elapsed time of approximately one and a half minutes, while the tactical component uses 43845 CPU seconds, an elapsed time of approximately 20 minutes. Times vary for different questions. These statistics were obtained when using the system in a shared environment. An improvement in speed could be achieved by using a dedicated system. It should be noted, furthermore, that the strategic component is not

compiled, while the tactical component is. Thus, a further speed-up of the strategic component could be achieved through compilation. This length of response time is clearly unacceptable for practical use. The tactical component was designed as a non-deterministic, recursive process and this accounts for its slow speed. A sufficient investment of time in implementation effort, however, could result in a considerable savings in speed.

4.2 System Components

The remainder of this chapter describes the implementation methods used for the TEXT system components in some detail. Since the knowledge representation is the limiting semantic factor on the remainder of the system, its contents and structure are described in detail (Section 4.3) before setting out the processing components.

The strategic component consists of processes that handle the construction of the relevant knowledge pool, selection of an appropriate schema for the answer, filling of the schema with propositions from the relevant knowledge pool, and monitoring of schema filling through the use of focus constraints. On receiving a question, the TEXT system selects a set of possible schemas to use for the answer. Semantic processes then construct the relevant knowledge pool on the basis of the input question. Section 4.4 describes how relevant knowledge is selected. A characterization of the information available in this pool is used to select a single schema from this set and predicate semantics are then used to select propositions from the

relevant knowledge pool which match the predicates in the schema. This process was described in detail in Chapter 2 and is, therefore, not discussed further here. The focusing mechanism monitors the matching process• Where there are alternative predicates in the schema or where a predicate matches more than one proposition in the knowledge pool, the focusing mechanism selects the most appropriate proposition for the answer. Implementation of the focus constrains was presented in Chapter 3.

The output of the strategic component is passed to a dictionary interface between the strategic and tactical component. The dictionary translates each proposition in the message into a deep structure representation of the sentence to be generated. This involves translating the predicate into the sentence verb and mapping the instantiated predicate arguments into the case roles of the verb. This process entails the selection of lexical items for the instantiated arguments. A detailed discussion of this process is provided in Section 4.5.

Dictionary output is fed to the tactical component, which uses a functional grammar to translate the message into English. Section 4.6 describes the implementation of this component in the system.

Examples of TEXT system output are provided in Appendix B. The chapter concludes with a discussion of the practical aspects of the system and what could be done to improve those aspects. A diagram of the TEXT generation system is shown in Figure 4.2.1.

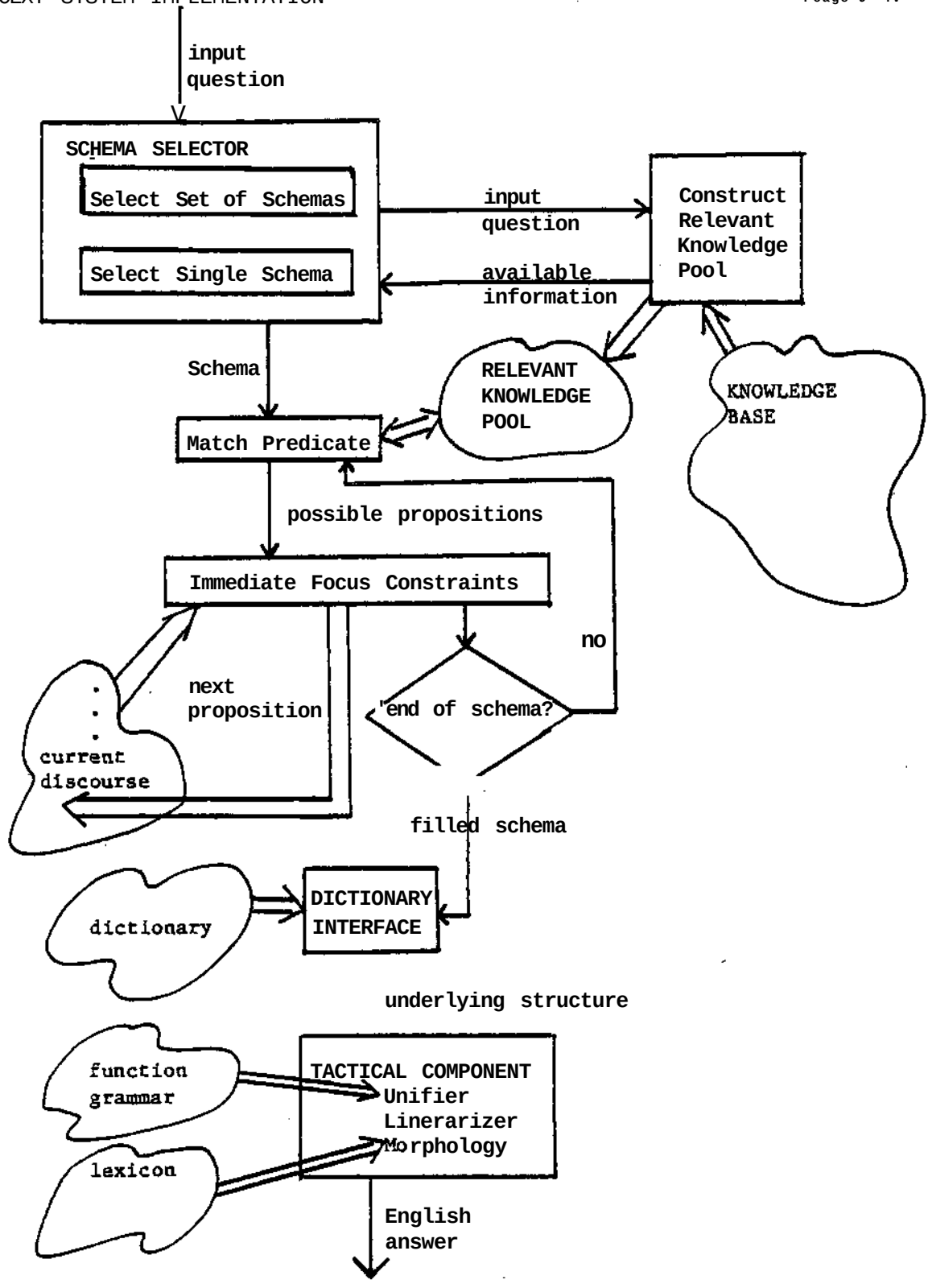


FIGURE 4.2.1 System Components and Flow of Control

4.3 Knowledge Representation

Answering questions about the structure of the database requires access to a high level description of the classes of objects in the database and the relationships between the classes. Despite the fact that knowledge representation is not the focal point of this dissertation, it is, nonetheless, an important aspect of a generation system. The information that is, and can be, represented in the knowledge base limits the overall expressive power of the generation system. Unless the generation system includes a powerful inferencing mechanism, information that is not contained in the knowledge base cannot be expressed. It was necessary, therefore, to either use or develop some type of representation which is suitable for answering questions about the database structure.

Since this research is not in knowledge representation, a decision was made to do very little development of new representation features. Instead, features used in data models of previous natural language database systems were selected for use in the TEXT system knowledge base. By restricting the representation to include features found in other database models, questions about how useful such a representation is for generation can be answered. However, because of the need to describe the data from a view point compatible with a naive user's (as well as describing the system's view), an additional information-type was developed for the TEXT knowledge base.

The data-types used in the TEXT knowledge base include entities, attributes, and relationships [CHEN 76]. A generalization hierarchy on entities ([SMITH & SMITH 77], [LEE & GERRITSEN 78]) was developed and characteristic features of each sub-type in the hierarchy were represented, indicating the basis for the splits in the hierarchy. A topic hierarchy on attributes [SCHUBERT et. al. 79] was also developed. Sub-types of entities which exist as physical records in the database were generated automatically and added to the generalization hierarchy before the knowledge base was used in order to avoid extensive inferencing [MCCOY 82]. Each of these features is discussed in detail in the following sections.

4.3.1 Basis -

The knowledge representation used in the TEXT system is based on the Chen entity-relationship model [CHEN 76]. This model consists of entities, relations between entities, and attributes of entities. An entity represents a class of instances in the database (each instance is composed of a list of values). In other words, an entity is an object-type occurring in the database. In this work, a class of objects in the database is termed an entity and an object in the database an instance.

Each entity has a set of attributes associated with it. An attribute is a function (given a name) from an entity into a value set. Restrictions on that value set are noted. The attribute LENGTH, for example, may map from the entity SHIP into the set of non-negative integers less than 1000. An instance is actually a tuple of values corresponding to the attributes associated with the entity. Each entity has a primary key which is one or more attributes whose value uniquely identifies an instance in the database.

Definition 4.3.1:
Entity: A class of instances in the database with common database attributes and relations.

Definition 4.3.2
Attribute: A function from an entity into a restricted value set.

Definition 4.3.3
Primary key: One or more attributes whose value uniquely identifies an instance of an entity in the database.

Relations occur between two or more entities. In the database used for the TEXT system, only binary relations occurred. The cardinality of the relation (whether it is one-to-one, one-to-many, many-to-one, or many-to-many) is indicated as well as the roles the entities play in the relation. This information is particularly useful for relations between more than two entities. Although the particular database used only allows for binary relations, a change in database would not require a change in knowledge representation formalism. Relations also may have attributes associated with them. The SHIP, for example, CARRIES GUIDED PROJECTILES. The attribute QUANTITY indicates

how many GUIDED PROJECTILES the SHIP carries. The CARRY relation is one-to-many, from SHIPs to GUIDED PROJECTILEs; the SHIP is the carrier and the GUIDED PROJECTILE is the possessed weapon.

Definition 4.3.4
Relation: A named tuple of entities with the following features:

1. A named role is assigned to each entity in the tuple.
2. A cardinality (either one or many) is assigned to each entity in the tuple. Cardinality specifies the number of instances of an entity which may participate in the given relation.
3. The tuple possesses attributes.

A simple example of the Chen entity-relationship model is shown in Figure 4.3.1. A circle represents an entity and a diamond represents a relation. Lines from relations to entities are labeled by the role the entity plays and its cardinality in the relation. Attributes are illustrated by labeled arrows into value sets (hatched circles). In the example, two entities, SHIP and the AIRCRAFT, are shown to be related through the CARRY relation. The SHIP is the carrier and the AIRCRAFT is the carried object. A single SHIP may carry more than one AIRCRAFT. A SHIP is shown to have four attributes: OFFICIAL_NAME, MAXIMUM_SPEED, DISPLACEMENT, and LENGTH; and AIRCRAFT has three: NAME, ALTITUDE, and FUEL_TYPE (Both entities actually have more attributes in the ONR database. These were not shown for reasons of clarity.).

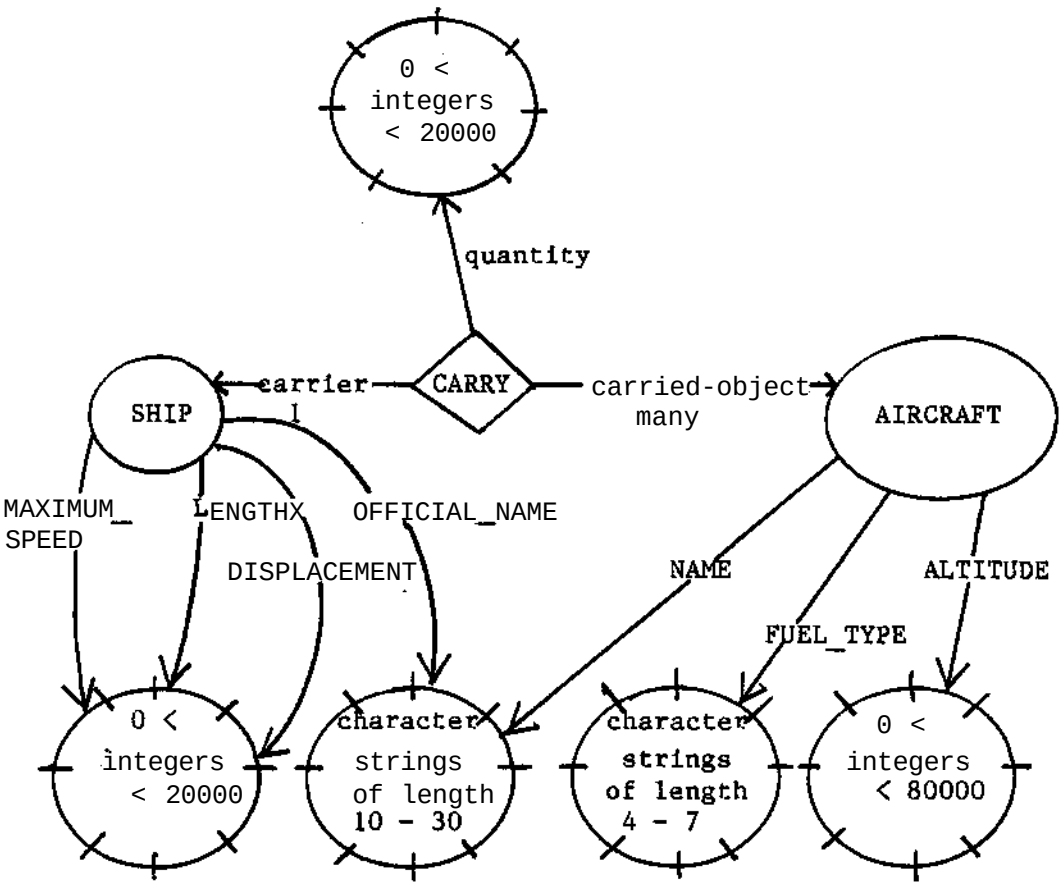


FIGURE 4.3.1 The Entity-Relationship Model

Note that the Chen entity-relationship model portrays a limited view of the data and does not contain a rich enough description for the type of generation required here. For example, ships can only be described in terms of types of values contained in the database, although other features of the ship may be important in describing it to a user.

4.3.2 Use Of Generalization -

Features that have been used in a number of database models for natural language database systems, but are not present in the Chen entity-relationship model, were also adopted for use in the TEXT system. One of these is the concept of generalization [SMITH and SMITH 77], a technique in modeling also used in semantic networks [HENDRIX 79]. In the TEXT system, a generalization hierarchy [LEE & GERRITSEN 78] on entities is used. A superordinate of a set of entities is formed if they have common features and can be grouped together as a class. In the ONR database, for example, the SHIP and the SUBMARINE are generalized to WATER-VEHICLE and AIRCRAFT is generalized as AIR-VEHICLE. Both the WATER-VEHICLE and the AIR-VEHICLE are generalized as VEHICLE. Part of the generalization hierarchy used in the TEXT system is shown in Figure 4.3.2.

The generalization hierarchy extends in depth to include sub-types of records in the physical database. For example, sub-types of the entity SHIP, for which a record exists in the database, include the AIRCRAFT-CARRIER, FRIGATE, DESTROYER, and CRUISER, among others. Some of the sub-types of the SHIP are shown in Figure 4.3.3. This portion of the hierarchy (along with the sub-types of the other leaves of the hierarchy shown in Figure 4.3.2) was generated automatically by a system called ENHANCE, which was developed by Kathleen F. McCoy (see [MCCOY 82] for a description of how this was done).

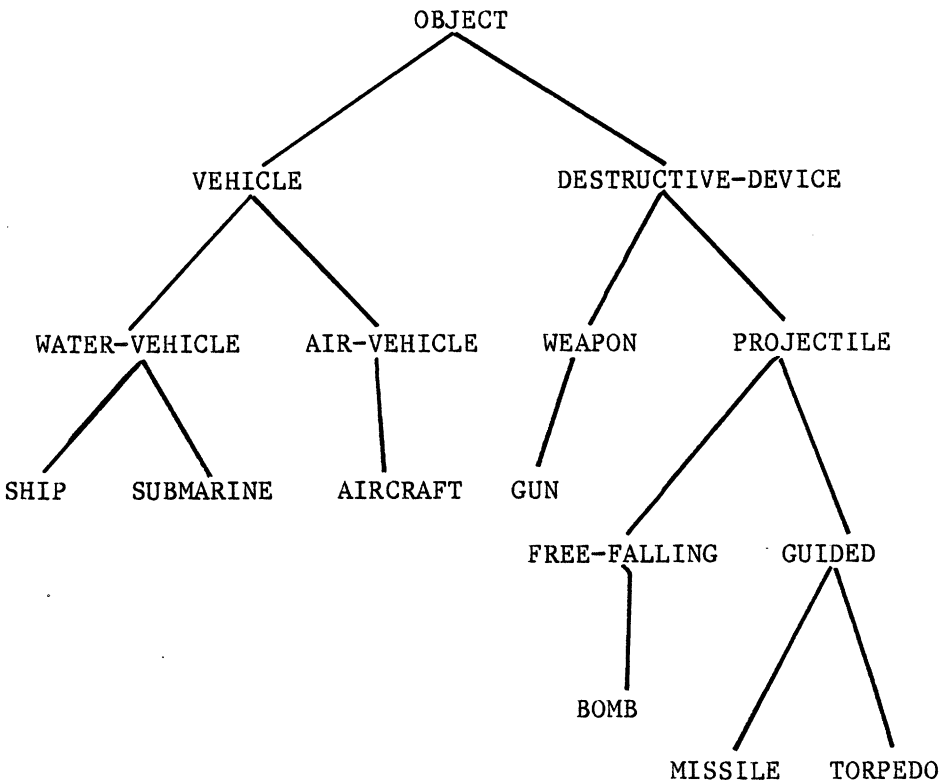


FIGURE 4.3.2 The Generalization Hierarchy

Entities for which records exist in the physical database are the only kind of entities used in Chen's representation. In this work, these types of entities are termed database entity classes. Use of a generalization hierarchy broadens the range of meaning of the term entity. It can refer to a generalization of a database entity class, a database entity class, or a sub-type of a database entity class. This distinction between the kinds of entities is especially important when representing the distinguishing characteristics of the sub-types of any

entity (see Section 4.3.3 and 4.3.4.2). In order to talk about distinction without confusion, the following definitions are adopte

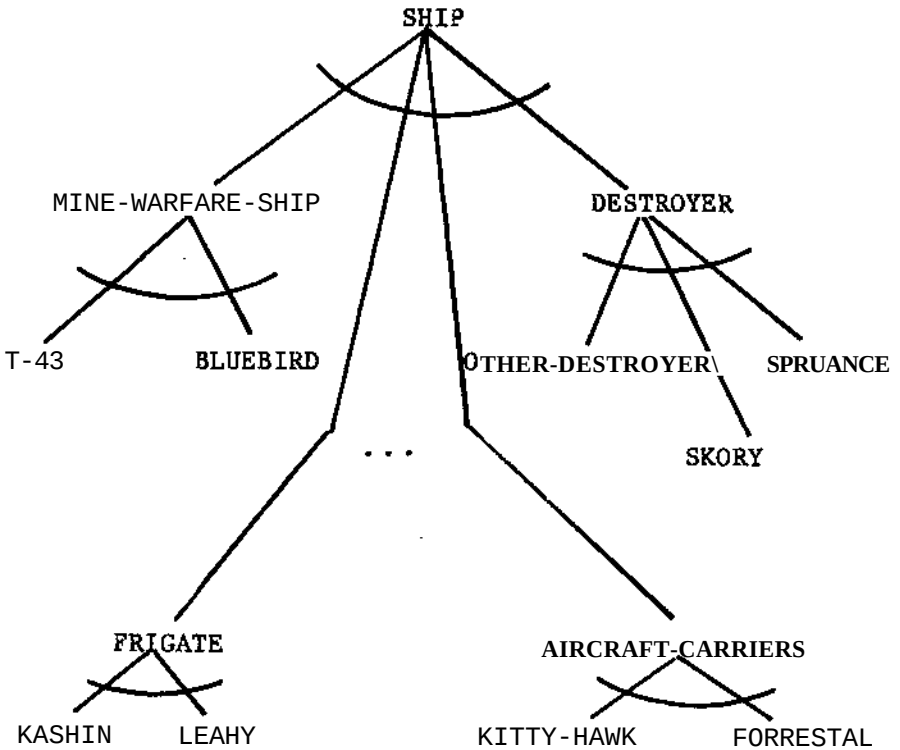
Definition 4.3.5
Database entity class: An entity for which a record exists in the physical database.

Definition 4.3.6
Database entity subset: A sub-type of a database entity class.

Definition 4.3.7
Database entity generalization: A generalization of a database entity class.

Definition 4.3.8
Entity (redefined): Any node in the generalization hierarchy. This includes database entity classes, database entity subsets, and database entity generalizations.

Figure 4.3.3 shows that mutual exclusion on database en subsets is also part of the generalization hierarchy. In fact, mu exclusion is used throughout the generalization hierarchy. Mu exclusion is used in a variety of knowledge representations (e.g [SMITH and SMITH 77], [BRACHMAN 79]). A set of sub-types is mutu exclusive only if no member of any sub-type is also a member of ano sub-type. If an entity has two sets of mutually exclusive sub-ty then an instance occurring in a sub-type of one set may also occur sub-type of the other set. For example, a DESTROYER (a member of mutually exclusive set) may also be a US SHIP (a member of ano mutually exclusive set), but a DESTROYER cannot also be a FRIG Mutual exclusion is represented graphically by drawing an arc ac the sub-type links of an entity.



Example Sub-classes formed for Entity Class SHIP
FIGURE 4.3,3

Definition 4.3.9
Mutual exclusion: A set of sub-types is mutually exclusive if and only if no member of any sub-type is also a member of another sub-type.

4.3.2.1 The Topic Hierarchy -

A hierarchy is also used on the database attributes in the TEXT system. Although this is not a feature common to many database models, it has been discussed and used in the AI literature (e.g. -

[SCHUBERT et. al. 79], [BRACHMAN 79]). In order to refer to e hierarchy without confusion, the hierarchy on attributes is termed topic hierarchy, while the hierarchy on entities is termed generalization hierarchy. The topic hierarchy in the TEXT sys proved to be especially rich and extremely useful in describ commonalities between entities. Attributes such as MAXIMUM_SPEED MINIMUM_SPEED were generalized to EXTREMEJSPEED, ENDURANCE_SPEED ECONOMICJSPEED generalized to REGULATEDJ3PEED, and EXTREMEJSPEED REGULATED_SPEED generalized to SPEED__INDICES. The topic hierarchy u for the TEXT system is shown in Figure 4.3.4.

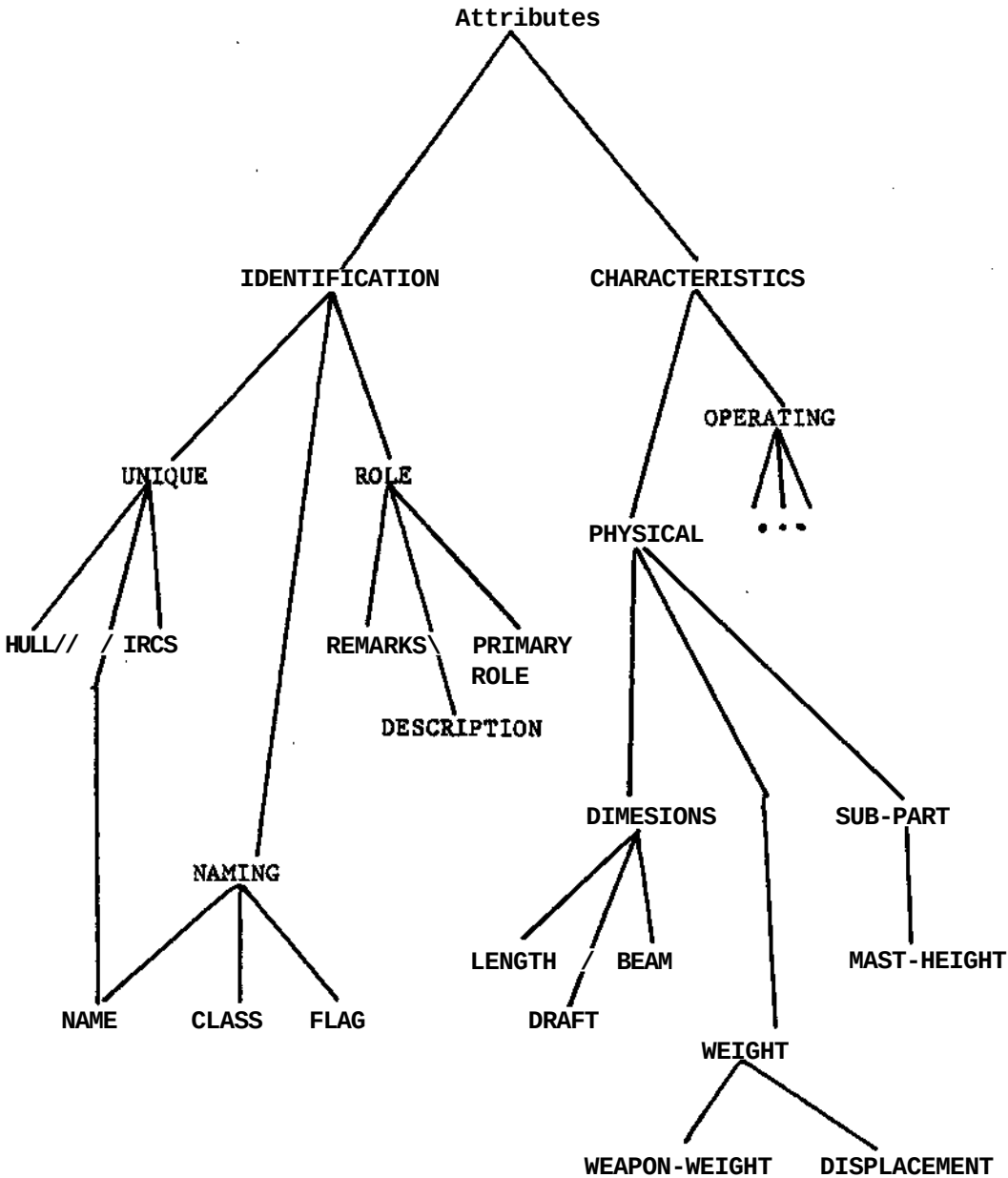
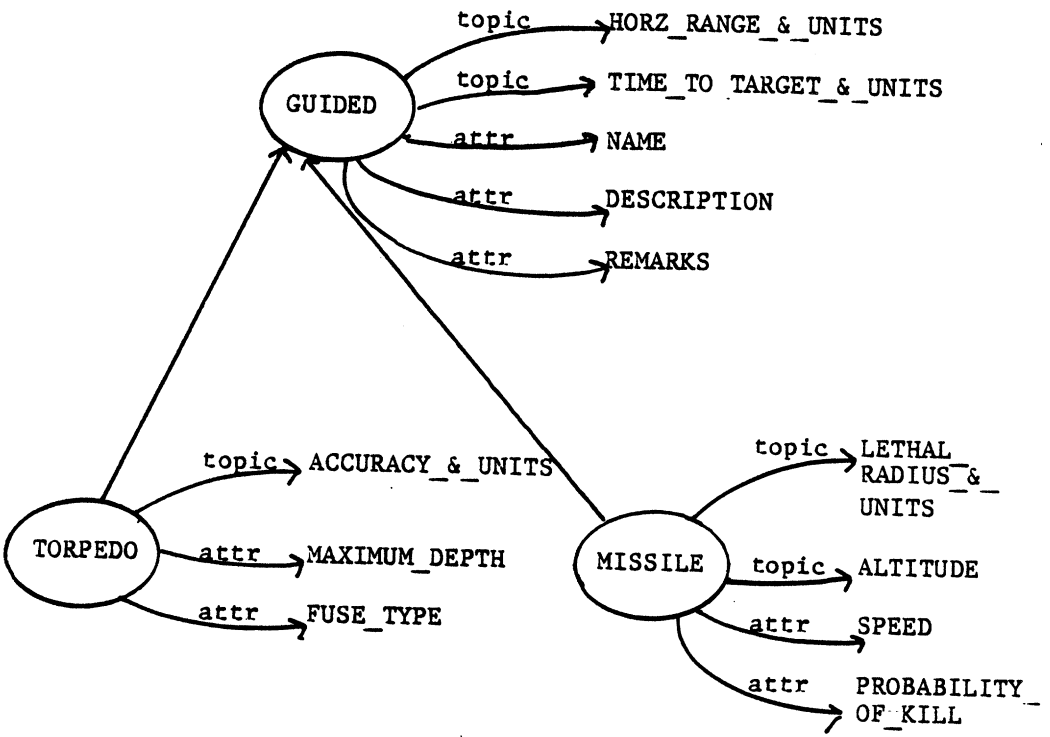


FIGURE 4.3.4 Part of the Topic Hierarchy

Both the generalization and the topic hierarchy allow for a more compact, as well as more abstract, representation of the data. In other words, both hierarchies allow for economies in the logical description of data as well as its physical storage. Use of the generalization hierarchy means that if both the SHIP and SUBMARINE have the attributes FUEL_TYPE and FUEL_CAPACITY, the attributes need not be duplicated for each entity but can be stored as attributes of their superordinate in the generalization hierarchy, WATER-VEHICLE (note that this would not be the case if there existed a third type of WATER-VEHICLE in the ONR database which did not have the attributes FUEL_TYPE and FUEL_CAPACITY). Furthermore, if an entity has the three attributes FUEL_TYPE, FUEL_CAPACITY, and REFUEL_CAPABILITY, all three attributes need not be associated with the entity. Instead, the superordinate of the three attributes, FUEL_INDICES, in the topic hierarchy can be attached to the entity. An example of the economy and abstraction gained by the combined use of these two hierarchies is shown in Figures 4.3.5 and 4.3.6. A sub-tree of GUIDED PROJECTILES is shown as well as the representation that would have to be used if the system did not include either of the two hierarchies.



Using Topic Hierarchy and Generalization Hierarchy
FIGURE 4.3.5

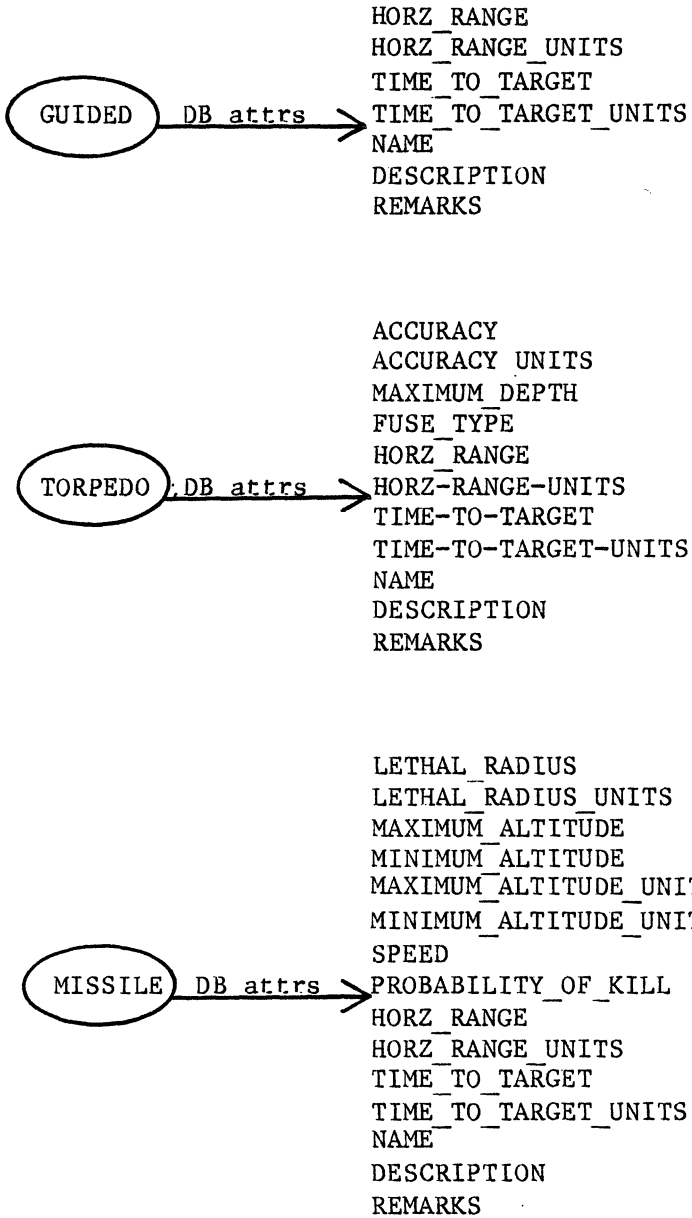


FIGURE 4.3.6 Without using Either Hierarchy

Creating a generalization hierarchy and topic hierarchy for a new domain is not a process that can be easily systematized. The best instructions that can be given for this process are to look for commonalities across entities to find an inclusive superordinate for some subset of the entities. Examination of hierarchies used in other systems to get some ideas of possible breakdowns is also helpful. Creating an acceptable hierarchy is a subjective process; there is no single "correct" hierarchy for a set of objects.

McCoy [MCCOY 82] has done some work on simplifying the process of generating the generalization hierarchy by developing a system that will automatically generate sets of sub-types for database entity classes. Transferring her system to a new domain requires specification by the designer of a set of domain-specific axioms which her system uses to form sets of sub-types, termed breakdowns, and to determine distinguishing characteristics of each sub-type (discussed below). The domain-specific axioms can be as simple (e.g. - a list of important database attributes) or as detailed (e.g. - a table of sub-class divisions desired) as the designer likes. The development of this system relieves the designer of the process of creating the generalization hierarchy from the level of database entity classes down when transferring the TEXT system to a new domain.

4.3,2.2 Relations -

Although a complete generalization hierarchy on relations was not included in the TEXT system, some generic information was used. Relations between entities are termed relation instances. Some instances of the same relation occur between different entities. For example, SHIPs carry GUNs, they carry MISSILEs, and they carry TORPEDOEs. Thus, the carry relation (called "ON" in the database) occurs between SHIPs and GUNs, SHIPs and MISSILEs, and SHIPs and TORPEDOEs. The similarity across these three different instances of the relation is captured by using a generic relation ON which has information about appropriate fillers of roles, attributes that each instance has (in this case, "QUANTITY"), and cardinality information (e.g. - whether the relation is one-to-many, etc.).

Definition 4.3.10
Relation instance: A relation between any two entities,

Definition 4.3.11
Generic relation: A representation of all relation instances of the same name between different entities. The representation includes entities that may participate in the relation, role names and their values, cardinalities, and attributes that are common to all instances.

By representing these two kinds of information about relations, greater variety can be achieved when describing them. Generic relations provide general information common to a class of instances and therefore allow for economy of description in a generated text when necessary. Instances allow for the provision of further detail in the

generated text.

4.3.3 Distinguishing Descriptive Attributes -

The use of the hierarchies described above allows for an abstract view of the data not provided by the Chen entity-relationship model; it describes the data in other than strictly database terms. In the interest of including more of this real-world view of the data, an additional feature was added to the TEXT knowledge base. Distinguishing descriptive attributes (DDAs), which are attribute name - value pairs, provide a view of the data not included in the database system point of view.

DDAs are attached to entities at each split in the hierarchy. They describe the basis for the partition in the hierarchy and are related to what Lee and Gerritsen call partition-attributes [LEE and GERRITSEN 78]. Lee and Gerritsen make an assumption, however, that doesn't always hold. They assume that there exists a single database attribute whose value in the database differentiates an entity into sub-types. This is not always the case.* For the database entity subsets of an entity, which have a large number of identical attributes, it is possible to find a single attribute, or set of

*Lee and Gerritsen's assumption may result from the fact that they are not working with an existing database and can choose to include whatever attributes and values that they like in the database itself. In this research, a meta-level representation is described for an existing database and it, therefore must be constructed within the confines of the values represented in the database.

attributes, whose value can partition the entity. For database entity generalizations, however, a single database attribute whose value can be used to partition the class may not exist. OBJECTS, for example, only have the attribute REMARKS which does not allow for a meaningful distinction. Yet, the partition of OBJECTS into VEHICLES and DESTRUCTIVE DEVICES is clearly a useful one. Above the database entity class level the different database attributes that each sub-type possesses, however, do indicate the basis for the partition. In this example, all VEHICLES possess attributes indicating their speed and fuel indices, while DESTRUCTIVE DEVICES possess attributes indicating their lethal indices.

A single attribute and value are formulated for database entity generalizations which provide an additional characterization of the partition. The choice of attribute and value is supported by the different database attributes that each sub-type possesses. These attributes correspond to the partition attributes of Lee and Gerritsen. The form of the DDA is different for entities which occur above the level of database entity classes in the hierarchy than for entities which occur below that level since values of database attributes differentiate database entity subsets while variation in database attributes possessed differentiates database entity generalizations. These two forms are described in more detail in the sections below.

4.3.3.1 PDAs For Database Entity Generalizations -

Each sub-type of a database entity generalization is given the same DDA name, but a different value which describes one way in which the entities are characteristically different. For example, the two main classes of entities in the ONR database, VEHICLE and DESTRUCTIVE DEVICE, both have DDA names of FUNCTION. The VEHICLE'S FUNCTION is TRANSPORTATION (DDA value) and the DESTRUCTIVE DEVICE has a FUNCTION of LETHALITY (DDA value).

DDAs consist of a single word attribute name and value (hyphenation is sometimes used). DDA names are either selected from a set of standard functional terms (e.g. - function, role) or from any of the cases (e.g. agent, patient, to-location, from-location, etc. See [PALMER 81] for a discussion of various verb cases) of higher-level DDA names or values in the generalization hierarchy. For example, one case of the VEHICLE'S DDA value, TRANSPORTATION, is travel-medium. TRAVEL-MEDIUM is the DDA name for WATER-VEHICLE and AIR-VEHICLE.

Definition 4.3.12
Distinguishing Descriptive Attribute (DDA) (Part I): An attribute-value pair associated with each entity in the hierarchy describing a characteristic difference between the entity and its siblings such that

attribute (entity) - attribute (sibling)
for all mutually exclusive sibling of entity
value (entity) ~= value (sibling)
for all mutually exclusive sibling of entity

4«3.3.2 Supporting Database Attributes -

Distinguishing descriptive attributes for database entity classes and their generalizations are supported by existing database attributes that illustrate the reasons for the chosen DDA name-value pair* The VEHICLE'S transportation function is supported by the fact that all VEHICLES in the ONR database have attributes describing their TRAVEL-MEANS, such as SPEED, FUELJTYPE, FUELjCAPACITY, etc. DESTRUCTIVE DEVICEs, on the other hand, have database attributes providing information on LETHAL_INDICES (e.g. - PROBABILITY_OF_KILL, LETHAL_ACCURACY, etc.)*

Definition 4.3.13
Supporting database attribute: A subset of an entity's attributes such that:
if El is a generalization of E2 and
DDA-name (E1,E2) = Al
DDA-value (E1,E2,A1) = VI
then y is a supporting database attribute of E2
if y is a database attribute of E2 and y
implies DDA-value (E1,E2,A1) = VI

The topic-hierarchy on attributes is very useful here, since it is rare that all entities in a class have exactly the same database attribute. For example, ships have MAXIMUM_SPEED, ENDURANCE_SPEED, and ECONOMIC_SPEED, the SUBMARINE has OPERATING_SPEED, and the AIRCRAFT has CRUISEJ5PEED. While none of these database attributes have exactly the same name, they each provide information about the speed capabilities of the various VEHICLES. In this case, it would be useful to associate the superordinate in the topic-hierarchy, SPEED__INDICES, with the entity VEHICLE and the specific types of SPEED with the SHIP,

SUBMARINE, and the AIRCRAFT. This capability is similar to role differentiation as described by Brachman [BRACHMAN 79].

In order to do so, a distinction is made between cases like this, where entities share related, but not identical attributes, and cases where entities share identical attributes. Each supporting database attribute is linked to its entity via either some-type-of or have. Have indicates that all entities in the generic class possess each database attribute occurring under the given topic in the topic hierarchy (or the database attribute if a leave in the hierarchy is given). Some-type-of indicates that all entities in the generic class possess attributes related by the given topic in the topic hierarchy. Thus, VEHICLE SOME-TYPE-OF SPEED INDICES indicates that each VEHICLE in the ONR database has some of the attributes occurring under SPEED_INDICES in the topic hierarchy.

The computational interpretation for this intuitive description of some-type-of follows a fairly rigid set of rules that can be used by a designer when determining how to associate the supporting database attributes drawn from the topic hierarchy with entities in the generalization hierarchy for a new domain. Working from the leaves of the topic hierarchy up, the following rules should be applied to each entity in the generalization hierarchy:

For each superordinate attribute in the topic hierarchy:

1. If each sub-type of entity is determined to have every attribute under a superordinate attribute, then use entity have superordinate attribute.

2. If each sub-type of entity is determined to have at least one attribute under a superordinate attribute and no sub-type shares attributes under the superordinate attribute, then entity some-type-of superordinate attribute.

3. If each attribute under a superordinate attribute is shared by 2 or more sub-types of entity in either have or some-type-of then entity some-type-of superordinate attribute.

4. If each sub-type of entity shares some attribute under a superordinate attribute in either have or some-type-of and there exists more than one attribute which is not shared, then sub-type some-type-of attribute-i, sub-type have attribute-j, for each applicable attribute-i, attribute-j.

Diagrams representing each of these cases are shown in Figures 4.3.7 -4.3.10. The distinguishing descriptive attributes and their supporting database attributes for the subtree under VEHICLEs are shown in Figure 4.3.11.

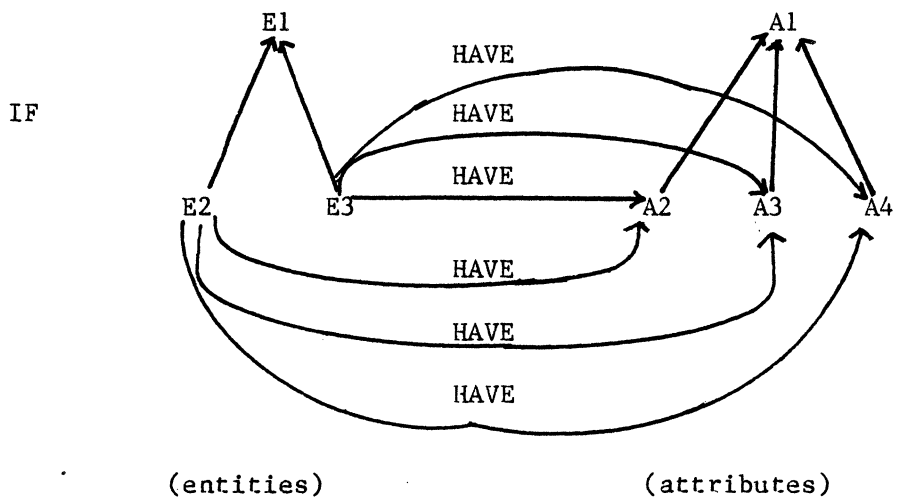


FIGURE 4.3.7 Attaching Supporting DB attributes - have

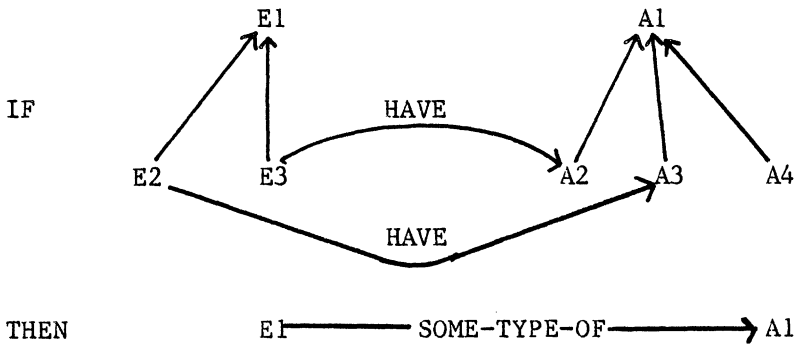


FIGURE 4.3.8 Some-type-of Interpretation (Part I)

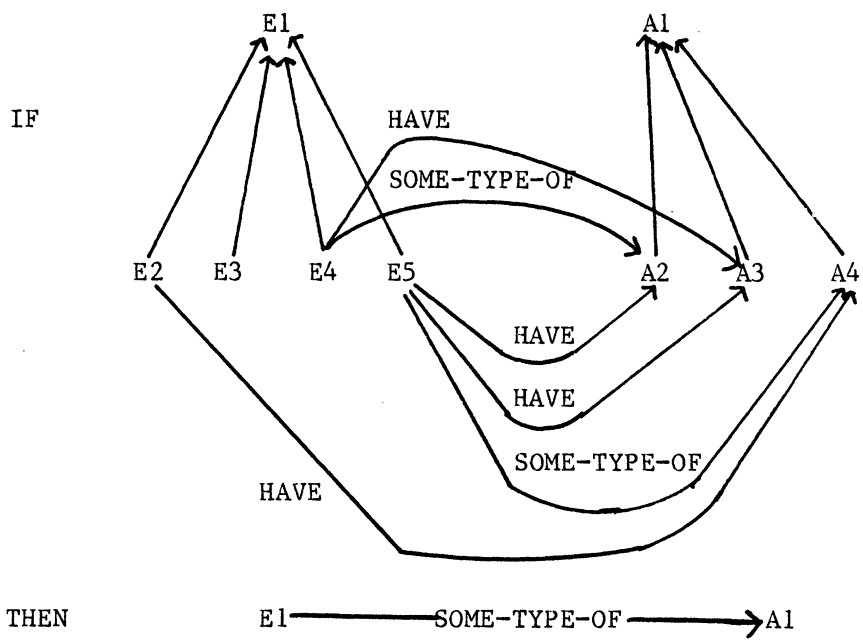


FIGURE 4.3.9 Some-type-of Interpretation (Part II)

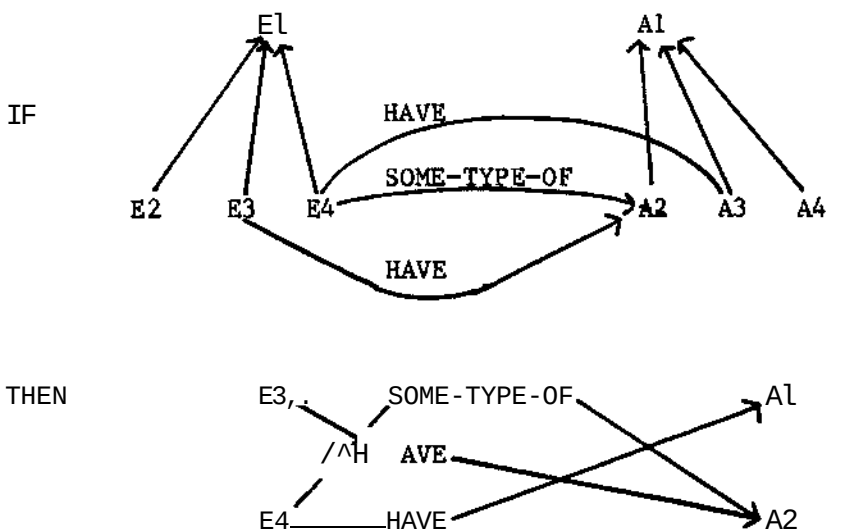


FIGURE 4.3.10 Some-type-of and Have

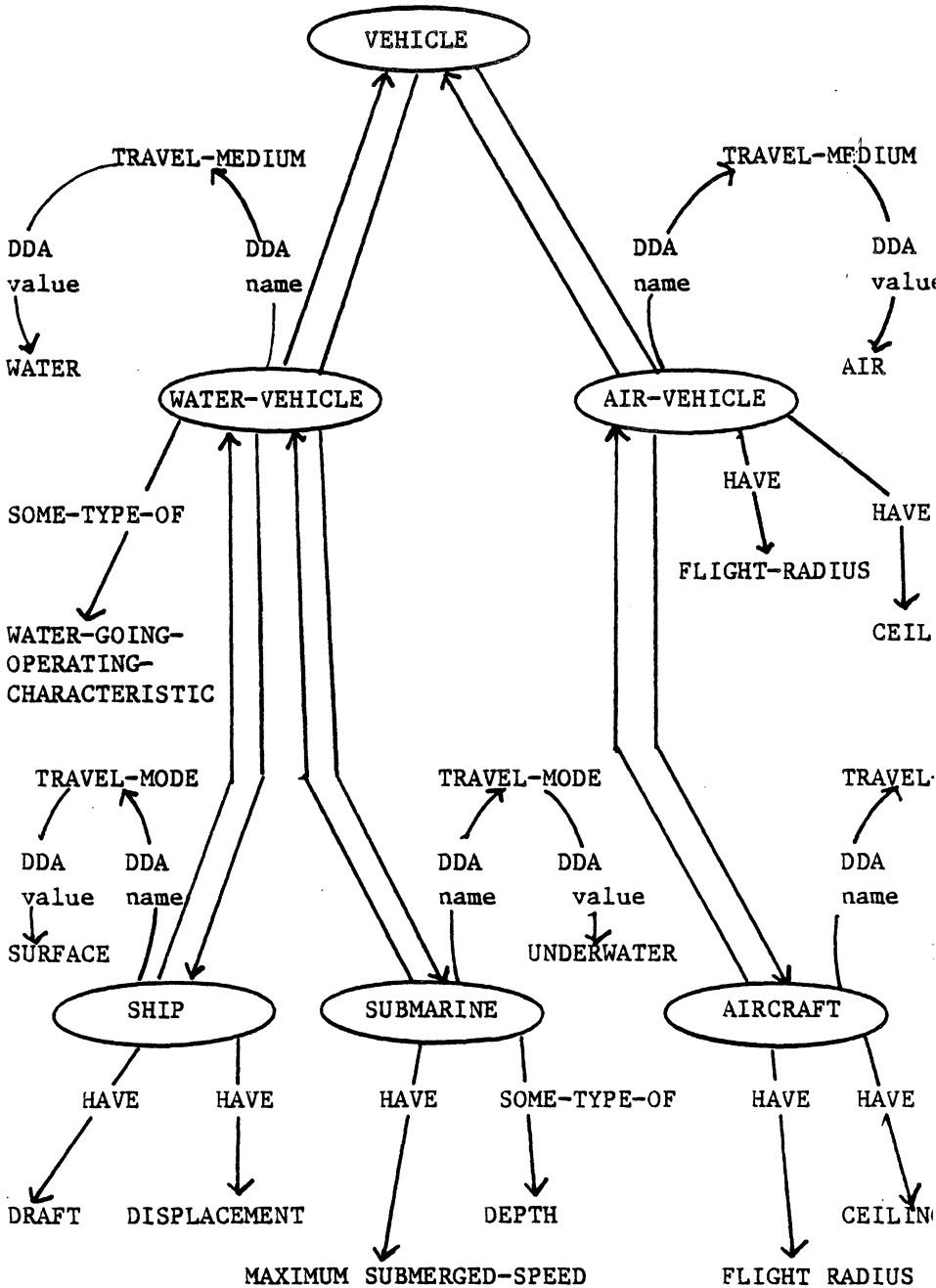


FIGURE 4.3.11 DDAs and Supporting Database Attribute

4.3.4 Database Entity Subsets - Database entity subsets have similar types of descriptive information associated with them. This descriptive information was generated automatically by McCoy's system along with the subset nodes. Each data-type associated with a subset has the same function as its counter-part in the upper half of the generalization hierarchy, but has a slightly different format.

Sub-sets are formed by McCoy's system on the basis of three sources of information: the knowledge base already formed by hand, actual database values, and a set of world knowledge axioms. The axioms fall into three classes: very specific, specific, and general. Very specific axioms dictate actual breakdowns that a database designer would like to see in the knowledge base. They specify both a sub-type name and a unique identifier (database attribute and value) of that sub-type. For example, sub-types of the SHIP may be formed on the basis of identifying characters in the HULL_NO. All AIRCRAFT-CARRIERs, in fact, are identified by the first two characters of the HULL_NO being CV. Specific axioms specify attributes which are important for the particular database domain. For example, CLASS, FUEL_TYPE, and FLAG (which specifies country), are important for a database containing information about military vehicles. Thus, a sub-type may be formed on the basis of the value of a SHIP's CLASS, such as KITTY-HAWK.

4.3.4.1 Based Database Attributes -

The uniquely identifying attribute on which the sub-type formation was based is called the based database attribute (counterpart to the supporting database attribute in the upper half of the hierarchy and to Lee and Gerritsen's partition attribute). They represent the defining attribute and value for the sub-type in the database. For the AIRCRAFT-CARRIER, the based database attribute is HULL_NO paired with an indication that the first two characters of the HULL_NO must be CV. Since some sub-types may be based on more than one value, a disjunction may be used. Cruisers, for example, are identified by a HULL_NO with first two characters of either CG or CL. The based database attribute, therefore, indicates the reason for the breakdown.

Definition 4.3.14
Based database attribute: Attribute-value pair of database entity subset whose value uniquely identifies instance as belonging to given sub-type.

4.3.4.2 DDAs For Database Entity Subsets -

McCoy's system selects as DDAs those database attributes whose collective value over the sub-type distinguishes that sub-type from every other mutually exclusive sub-type of the parent. In some cases, a single database attribute may be sufficient for forming a distinction. For example, ENHANCE chose attribute LENGTH as the DDA for SHIP sub-type AIRCRAFT-CARRIER since no other SHIP in the database has a LENGTH as large as the AIRCRAFT CARRIER's. The DDA value for a sub-type may be either a constant or a range. If all

AIRCRAFT-CARRIER's have a LENGTH between 1039 and 1063, but all other SHIPs have a LENGTH less than 1039, then the range is sufficient distinction. If an attribute has a constant value across a sub-type, and no other sub-type has the same value for that attribute, then it is sufficient distinction.

In some cases, no single attribute may be sufficient for distinguishing one sub-type from another. For example, an OCEAN-ESCORT may hypothetically have either a smaller LENGTH or a smaller BEAM than every other SHIP (i.e. - a DESTROYER may have a smaller LENGTH than the OCEAN-ESCORT, but have a larger BEAM, while a FRIGATE may have a smaller BEAM but a larger LENGTH). In such cases, a set of attributes and their values provide the distinguishing characteristics of a sub-type. If more than a single set of attributes distinguishes one sub-type from all others, McCoy's system uses world knowledge axioms to select that set providing the most meaningful distinction (see [MCCOY 82]).

Definition 4.3.15
DDA (part 2): Database attribute-value pairs that distinguish a database entity subset from all others such that:

No mutually exclusive sibling of entity has DDA-value (DDA-name sibling, parent) = DDA value (DDA-name entity, parent) OR No mutually exclusive sibling of entity has DDA-value (DDA-name sibling, parent) within the range of DDA value (DDA-name entity,parent).

4.3.4.3 Constant Database Attributes -

A database entity subset inherits all the database attributes of its ancestors, and, as mentioned earlier, has no additional attributes of its own. The value of all of its database attributes is, however, further constrained by its sub-typing. A database entity subset contains a restricted range of values across all of its attributes. In order to be able to describe these restrictions any attributes having constant values are recorded. In addition, the ranges of database attributes that appear as distinguishing descriptive attributes of other database entity subsets are also recorded. This is done so that comparisons can be made between these sub-types without an extensive amount of inferencing.

A database entity subset also inherits all the relations of its ancestors without having any additional relations of its own. Again, the values of the relation attributes are recorded as these are restricted in different ways across different database entity subsets. The SHIP, for example, has the relation ON with GUNs, MISSILEs, and TORPEDOEs. The DESTROYER, one database entity subset of SHIP, carries 2-8 GUNS, 2-40 MISSILES, and 8-99 TORPEDOES, while the PATROL SHIP carries 1-4 GUNs, and 2-8 MISSILES and 6-12 TORPEDOES.

A portion of one breakdown of the SHIP is shown in Figure 4.3.12. The descriptive information associated with two of the sub-classes is shown. Note that the set of distinguishing descriptive attributes is not the same for both sub-classes. While either LENGTH or DISPLACEMENT

distinguish the AIRCRAFT-CARRIER from all other SHIPS, the OCEAN-ESCORT has LENGTH in the same range as at least one other database entity subset of SHIP. DDAs for the OCEAN-ESCORT include DISPLACEMENT. Since LENGTH is a distinguishing descriptive attribute of another class, however, the values that the OCEAN-ESCORT's LENGTH ranges over are recorded as part of the constant values for its database attributes.

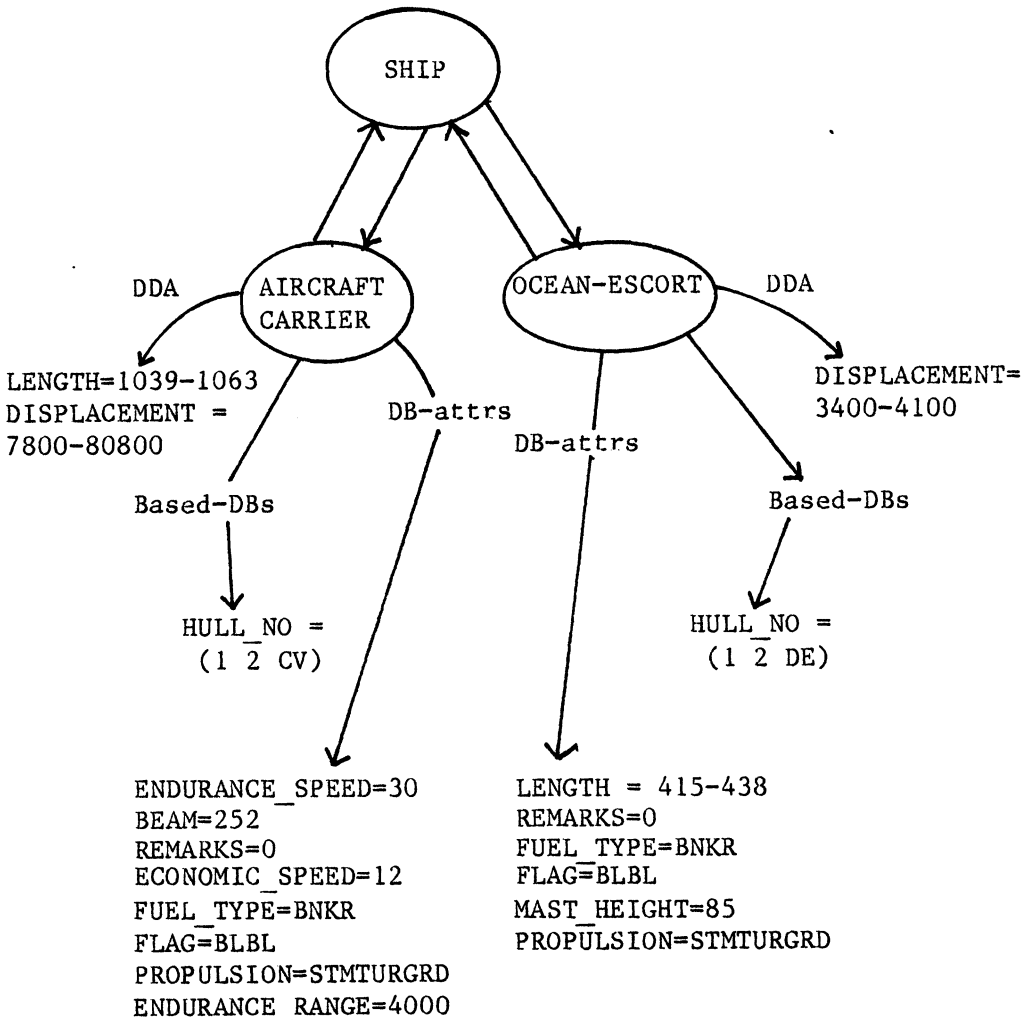


FIGURE 4.3.12 Information for Database Entity Subsets

4.3.5 Representation Summary -

The knowledge base used for the TEXT system consists of the following data types:

1. Entity:
   A class of instances which occur in the actual database. The term entity includes generalizations and sub-types.

2. Relations:
   Database relations between entities in the hierarchy. Includes both instances and generics.

3. Database attributes:
   Either attributes for which values exist in the database or generalizations of those attributes in the topic hierarchy. These are associated with both entities and relations.

4. Distinguishing descriptive attributes (names and values):
   Provide characteristic descriptive information on the distinguishing features of a sub-type.

5. Based database attributes:
   Uniquely identifying attribute-value pairs which indicate the basis for a breakdown on database entity classes.

6. Supporting database attributes:
   A subset of database attributes which indicate the basis for a breakdown on database entity generalizations and support the choice of a distinguishing descriptive attribute.

The knowledge base is implemented by maintaining a set of nodes which correspond to entities. Each node has a name (entity name) and a set of links which point to associated node information. Each link is labeled by the data type of the associated information, with the exception of links in the generalization hierarchy. These links are either labeled as "is-a" links or "type-of" links (the hierarchy is double-threaded. "is-a" points to a superordinate of an entity and

"type-of" points to a sub-type). A pictorial representation of a portion of the knowledge base used in TEXT (Figure 4.3.13) shows entities as circles and relations as diamonds. Link names can be any of the following: is-a, type-of, DDA-name, DDA-value, db-attr, have, some-type-of, based-db, <role-name> (e.g. - carrier), or mutual-exclusion.

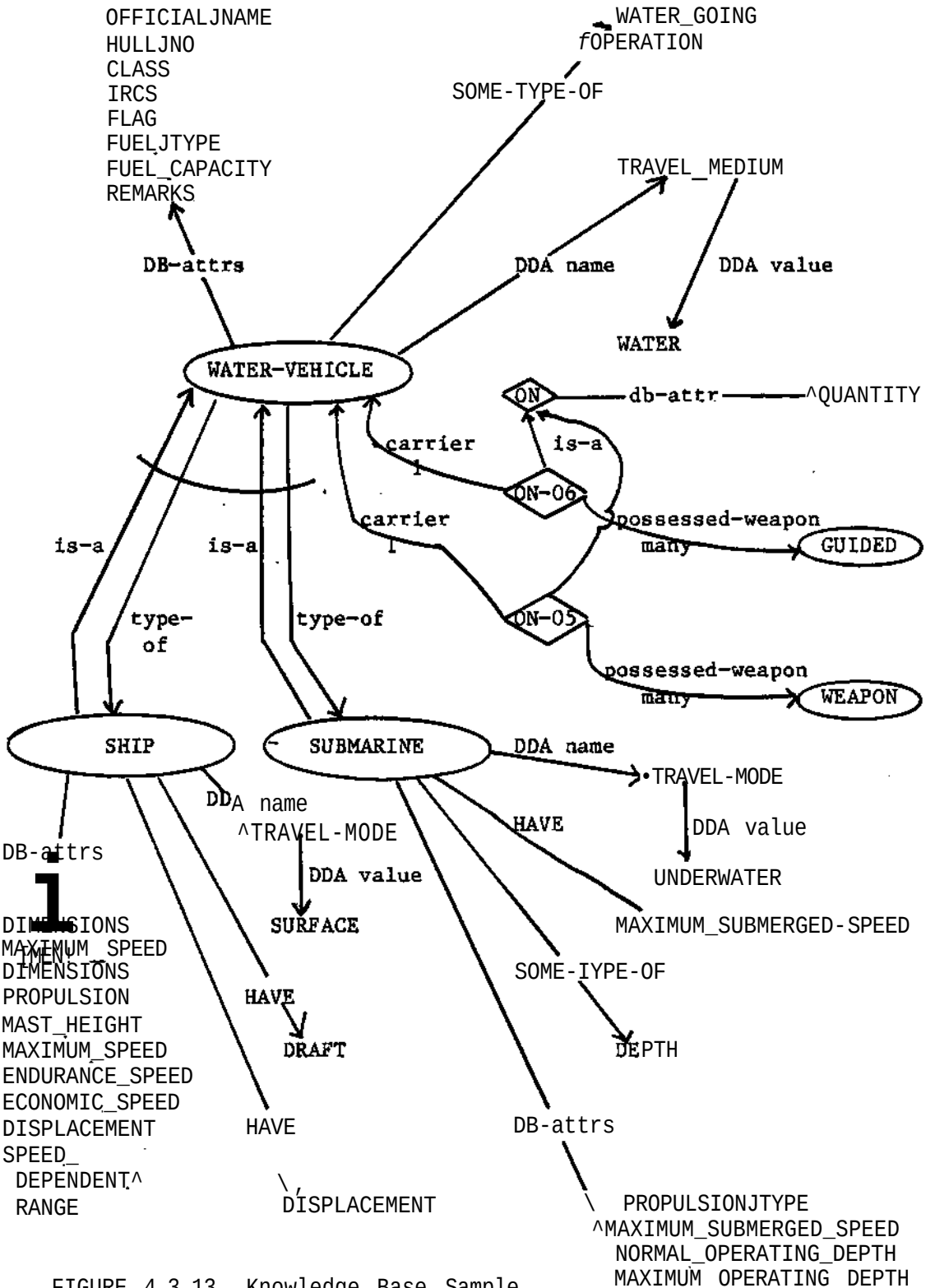


FIGURE 4.3.13 Knowledge Base Sample

4.3.6 Portability -

A frequent question asked of designers of natural language database systems regards how easy it is to transfer the system to a new database. In the TEXT system, the modules which must be changed are the knowledge base and the dictionary (see Section 4.5). Creating a knowledge base for a new domain is not an easy task. Several steps were taken, however, to simplify the process. The first of these was to use features that are used in many database models of database systems and therefore, familiar to database managers. Secondly, steps were taken to systematize the process of adding the new feature of information used in this model (distinguishing descriptive attributes and their supporting database attributes). And finally, a system was written to automatically create sub-types of database entity classes.

Part of the knowledge base must be formulated and typed in by hand. This includes the generalization hierarchy (working from the set of entities taken from the database schema upward), association of database attributes with the appropriate level in the hierarchy, association of database relations with the appropriate level in the hierarchy, and the creation of the distinguishing descriptive attributes (both name and value) and their supporting database attributes. Creation of both the generalization hierarchy and the distinguishing descriptive attributes is subjective to a certain extent. Names and values of distinguishing descriptive attributes for database entity classes and their generalizations should be developed by first examining the different kinds of database attributes

associated with nodes in the hierarchy. This information can be used to guide the selection of the dimension across which sub-classes of an entity vary. Choosing from a limited set of attribute names (standard functional terms, or cases of higher level attribute names or values) aids this process. Steps for specifying the supporting database attributes were given in Section 4.3.3.2. Note that only database attributes that support the distinguishing descriptive attribute are chosen. Thus, although all vehicles have the database attribute REMARKS, it does not indicate that its function is transportation and it is therefore not used as a supporting database attribute.

Running McCoy's system on a new database requires: 1) completion of the hand-generated hierarchy and associated information, 2) specification of a set of very specific axioms, if desired, and 3) specification of specific axioms. The very specific axioms are tables of sub-type names and unique sub-type identifier value (this is a value or partial field of a database attribute). They allow the user to specify apriori breakdowns. This step can be omitted if the user has no such breakdowns in mind. The specific axioms are a list of attributes considered important for the particular domain. The system attempts to form breakdowns based on the attributes specified. The system also generates all associated information for each sub-type specified in the breakdown. Both the very specific and specific axioms can be altered and the system rerun until an acceptable sub-typing is obtained.

4.3.7 Conclusion -

The knowledge base used in the TEXT system includes features standard to many database models. It draws primarily on work done by Chen [CHEN 76] and the Smiths [SMITH and SMITH 76] in data modeling. The reason for doing this was twofold: 1) the emphasis in this work was on generation of language and not on knowledge representation, and 2) to see how far generation could be pushed when using a relatively standard data model. Using a standard data model also makes the TEXT system more practical for actual use in a database system.

Since extended inferencing is not practical in the generation system, it was decided that a simple data model, such as the Chen entity-relationship model does not contain sufficient information for the task at hand. Such models represent only the types of values that are stored in the database for a particular entity are represented. Features such as the generalization hierarchy and the topic hierarchy were adopted in order to encode additional knowledge about the database concepts into the knowledge base. Distinguishing descriptive attributes, which provide real-world characteristics about sub-class distinctions, and sub-typing of entities based on world-knowledge axioms [MCCOY 82] were also added to the knowledge base for this reason.

Many issues in knowledge representation were not addressed by this work and are left for future research. The content of the knowledge base clearly limits the semantic power of any generation system. The

formalism chosen for the representation also, although less clearly, limits its expressive power. Although form can always be manipulated to produce the specific form necessary for the task at hand, there are situations where the manipulations required to do so are prohibitively expensive. This kind of situation is illustrated by the use of a system to automatically enhance the generalization hierarchy [MCCOY 82] before the generation system is used. Although the information produced by the system could be deduced only when needed, the time required to do so makes that option impractical. Researchers in artificial intelligence have been experimenting with the effectiveness of various formalisms including KL-ONE [BRACHMAN 79], semantic networks [HENDRIX 79], and first-order predicate calculus based formalisms (e.g. - [SCHUBERT et. al. 79], [MOORE 81]). Further development of research on issues of both content and formalism are extremely important to work done in natural language generation.

4.4 Selection Of Relevant Knowledge

The first step in answering a question directed to the TEXT system is to partition off a subset of the knowledge base that contains information relevant to the given question. This partitioning is done on the basis of the input question alone. The resulting subset is termed the relevant knowledge pool. As discussed in Chapter 3, the relevant knowledge pool is used to provide a limit on what needs to be considered when determining the content of the answer. It is similar to what Grosz has termed "global focus" [GROSZ 77] since its contents remain focused throughout the course of an answer.

Instead of developing a complex semantic reasoning engine, the approach taken in TEXT was to section off as much knowledge as could be considered relevant for the answer with the result that the system may err on the side of including too much information in the subset. Not all information in the subset need be included in the answer, however. The schemas determine exactly what information will be included from the relevant knowledge pool and in what order. Filling of the schema may end with information still remaining in the knowledge pool.

In determining which knowledge is relevant, a naive, infrequent user of the system is implicitly assumed. In situations where it is unclear whether more detail would be needed for this standard user, a choice was made not to include it. If a user-model were developed, the relevant knowledge pool could be dynamically expanded only in those specific situations where it is determined that more detail on a

concept is needed for a particular user (see Chapter 3.2.1).

4.4.1 Requests For Information And Definitions -

When responding to requests for information or for definitions, a relatively simple technique is used to partition the knowledge base. The area around the questioned object is sectioned off. All links of the questioned entity are preserved. These include links to its database attributes, its DDA name and value, either its supporting database attributes or its based database attributes, its relations, its super-ordinates in the generalization hierarchy, and its sub-classes in the hierarchy. The siblings of the questioned object are included in the relevant knowledge pool with all links preserved in case they are needed for analogies.* Descendents are also included with all links preserved for the cases where an entity is defined (or information available about it provided) in terms of its constituency. The only links included for all other entities selected for the relevant knowledge pool (these include the questioned object's parent and all entities related through database relations) are those which lead to pieces of knowledge already included. The parent, for example, would be included with only it subset links (links to the questioned object and its siblings). It should be noted that inheritance on database attributes and relations is preserved for the questioned

*Currently, only numerical comparisons are performed between an entity and its siblings. Tracking of discourse, however, would allow the system to make analogies to siblings that have been recently discussed.

object. Figure 4.4.1 shows the relevant knowledge pool that would b constructed for the question "What kind of information do you hav about ships?".

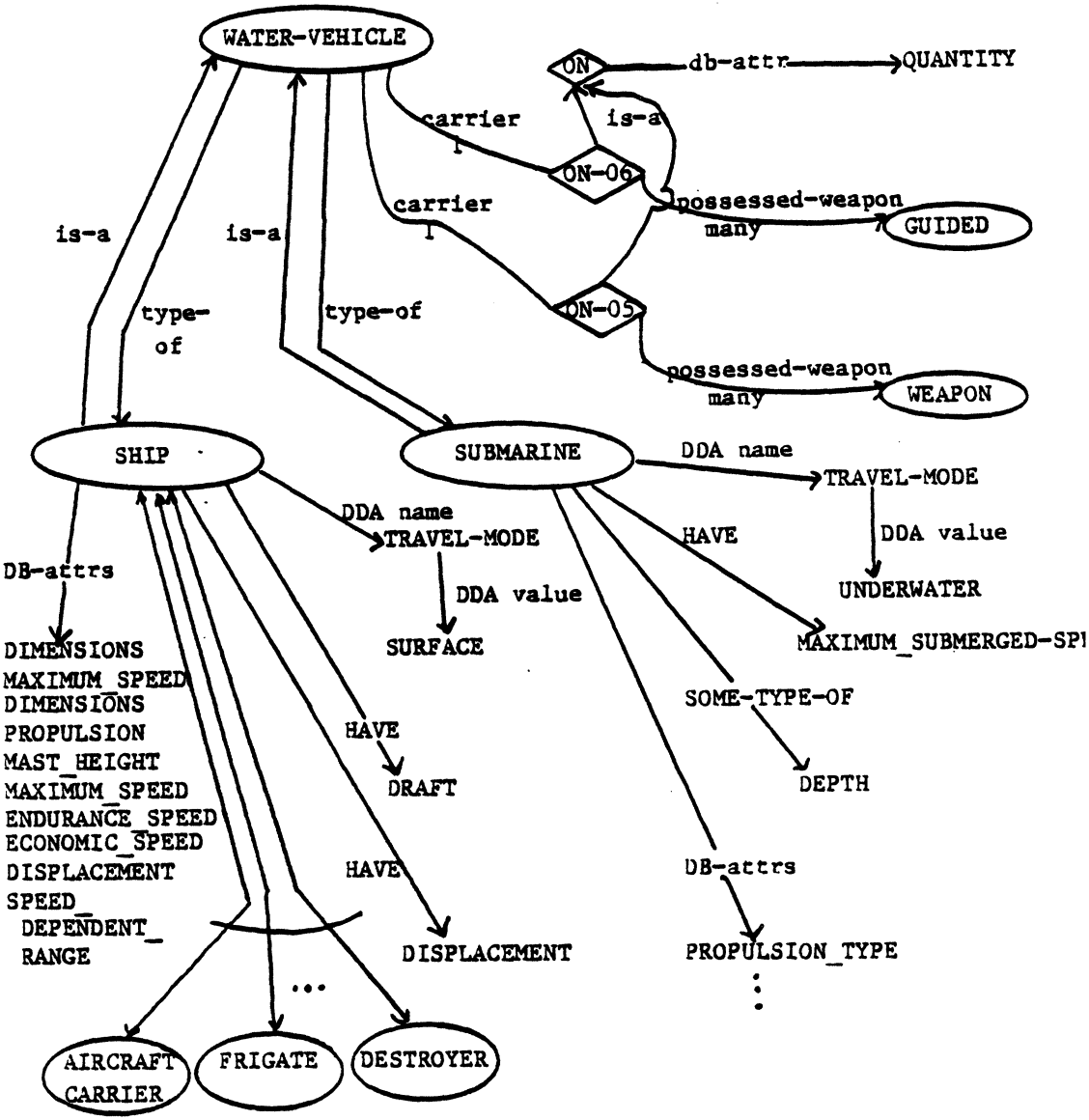


FIGURE 4.4.1 Relevant Knowledge Pool for Information and Definitions

^•4.2 Comparisions -

For questions about the difference between two entities, a slightly more complicated technique is used. The kind of information that is included in the relevant knowledge pool is dependent upon the conceptual closeness of the two entities in question. For two entities that are very similar, it is common to provide more detail when discussing their differences. When two objects are very different, a discussion of their differences could be endless and for this reason, the most salient distinction between the two is provided. This is a discussion of their generic class membership. Consider the questions shown in (1) and (2) below. Comparing the attributes of the part-time and full-time students (as in (3) below) can reasonably be part of an answer to question (1), but a comparison of the attributes of the raven and writing desk yields a ludicrous answer to question (2) (see (4) below). Instead, an indication that ravens belong to the class of animate objects, while writing desks are inanimate yields a better answer.

1. What is the difference between a part-time and a full-time student?
2. What is the difference between a raven and a writing desk?
3. A part-time student takes 2 or 3 courses per semester while a full-time student takes 3 or 4.
4. A writing desk has 4 legs while a raven has only 2.

4.4.2.1 Determining Closeness -

The TEXT system uses three categories in determining conceptual closeness. Two entities are classified as very close in concept if they are siblings in the generalization hierarchy. The ocean-escort and the cruiser are two entities in the ONR database that fall into this category since they are both sub-types of the SHIP. Two entities are classified as very different in concept if their common ancestor occurs at too high a level in the generalization hierarchy to provide useful information. An entity occurs at too high a level if it and all of its ancestors have no supporting database attributes for their DDAs. In other words, the concept is so vague that it has no database attributes that are common to all of its sub-entities. In the TEXT system, only the root node of the hierarchy happens to meet this description, although if the knowledge base was expanded to include more concepts this would not be the case. The DESTROYER and the BOMB are an example of two entities that are very different in concept since the only ancestor they share is OBJECT, the root node in the hierarchy.

Any two entities that don't fall into either of these classifications are categorized as class difference. These are entities that are not very close in concept but do have some similarities. An example of this type of category is the WHISKY and the KITTY HAWK. Although both are classes of water-going vehicles, the WHISKY is a submarine and the KITTY HAWK is a SHIP.

Figure 4.4.2 below depicts the generalization hierarchy wh illustrates the basis for these three types of classification.

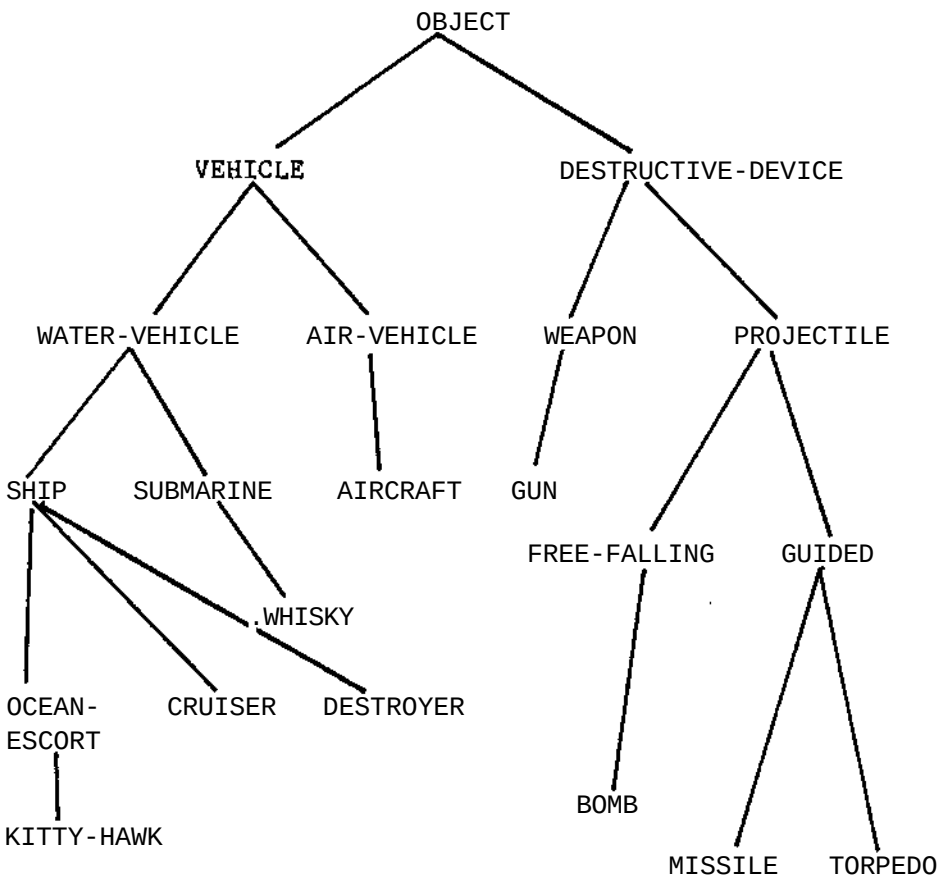


FIGURE 4.4.2 The Generalization Hierarchy

4.4.2.2 Relevancy On The Basis Of Conceptual Closeness -

For entities that are very close in concept, all links of the questioned objects are included in the relevant knowledge pool. As was the case for requests for definitions or for information, this includes the entity's database attributes, DDAs, relations, and supporting or based database attributes. The entities' common parent and its links are also included in the pool. No other entities (except those dictated by the entities' links) are included in the relevant knowledge pool. The partition that is constructed in response to the question "What is the difference between an ocean-escort and a cruiser is shown in Figure 4.4.3

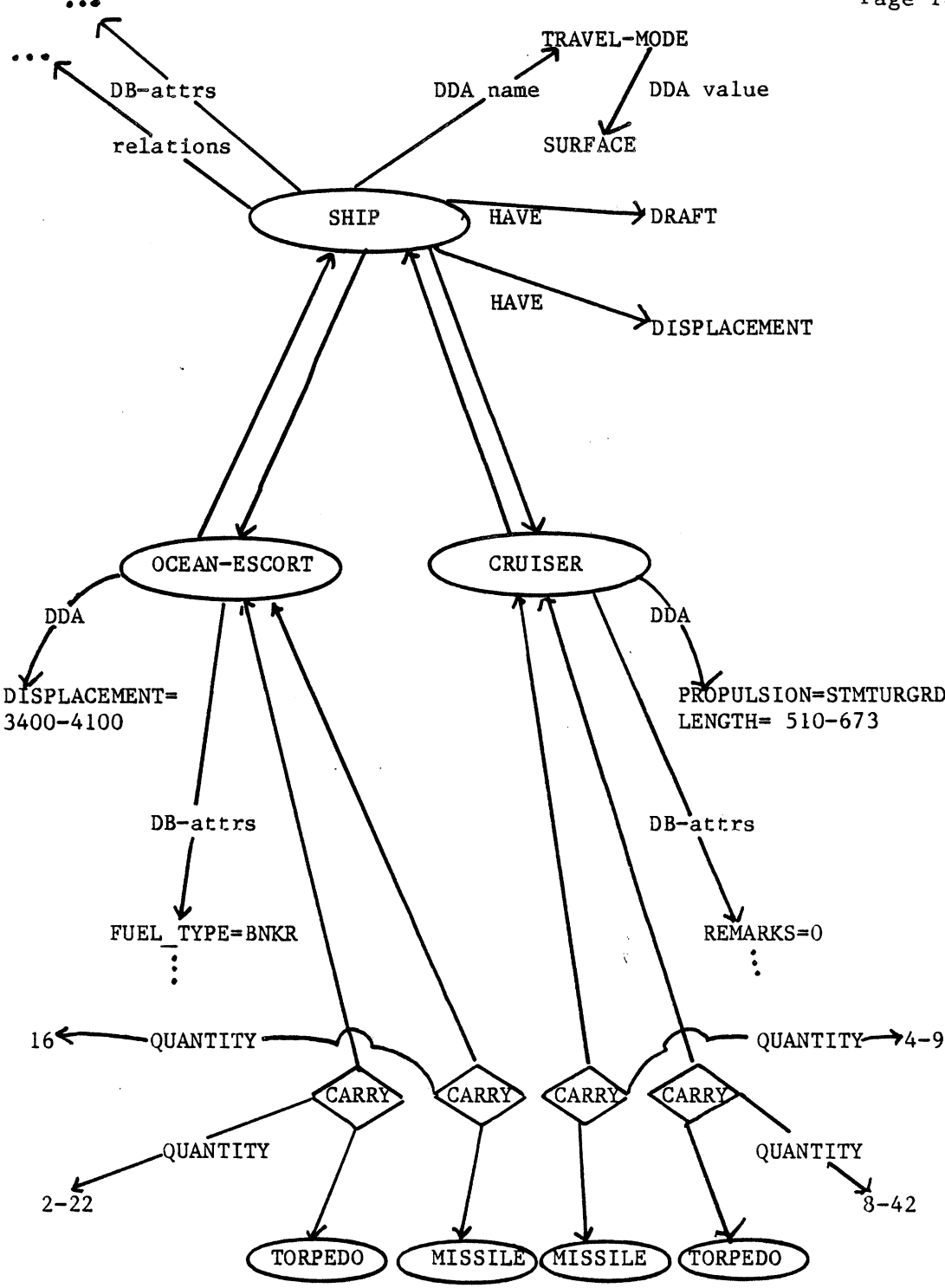


FIGURE 4.4.3 Relevant Knowledge Pool
for Category "very close"

In the case of entities which are very different in concept, only the superordinate links in the hierarchy and the reasons for the splits (DDAs) from the questioned objects to the nodes directly below the common ancestor are included. In order to avoid presenting long chains of superordinates when the common ancestor is very far away in the hierarchy, only the questioned objects, their parents, and the two nodes along the respective chains which are directly below the common ancestor are included in the knowledge pool. In addition, a search on common features of the questioned objects themselves is made in case there are any commonalities in database attributes, relations, or DDAs which were not common to all entities under the common ancestor. In practice, this rarely occurs for any two entities that are so different. The relevant knowledge pool that is constructed for the question "What is the difference between a destroyer and a bomb?" is shown in Figure 4.4.4.

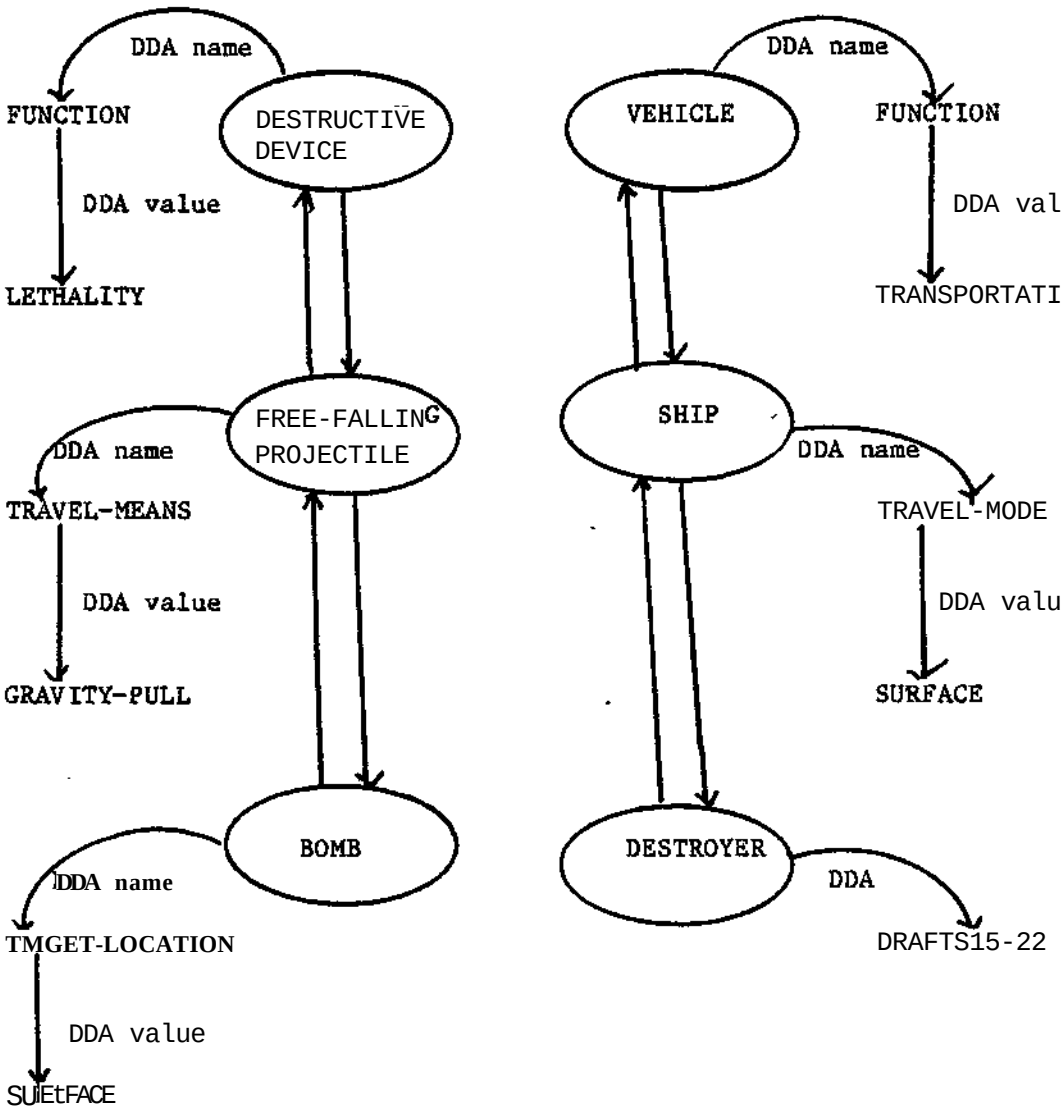


FIGURE 4.4.4 Relevant Knowledge Pool for Category "very different[11]

For entities that fall into the class difference categorization, the two questioned objects, their common ancestor, and the two children of the ancestor which are also ancestors of the questioned objects are included in the relevant knowledge pool. All links of each of these entities are included in the partition, with the exception of links that lead to other entities (e.g. some subset links of the common ancestor). No other entities are included. The relevant knowledge pool that is constructed for the question "What is the difference between the KITTY HAWK and the WHISKY?" is shown in Figure 4.4.5.

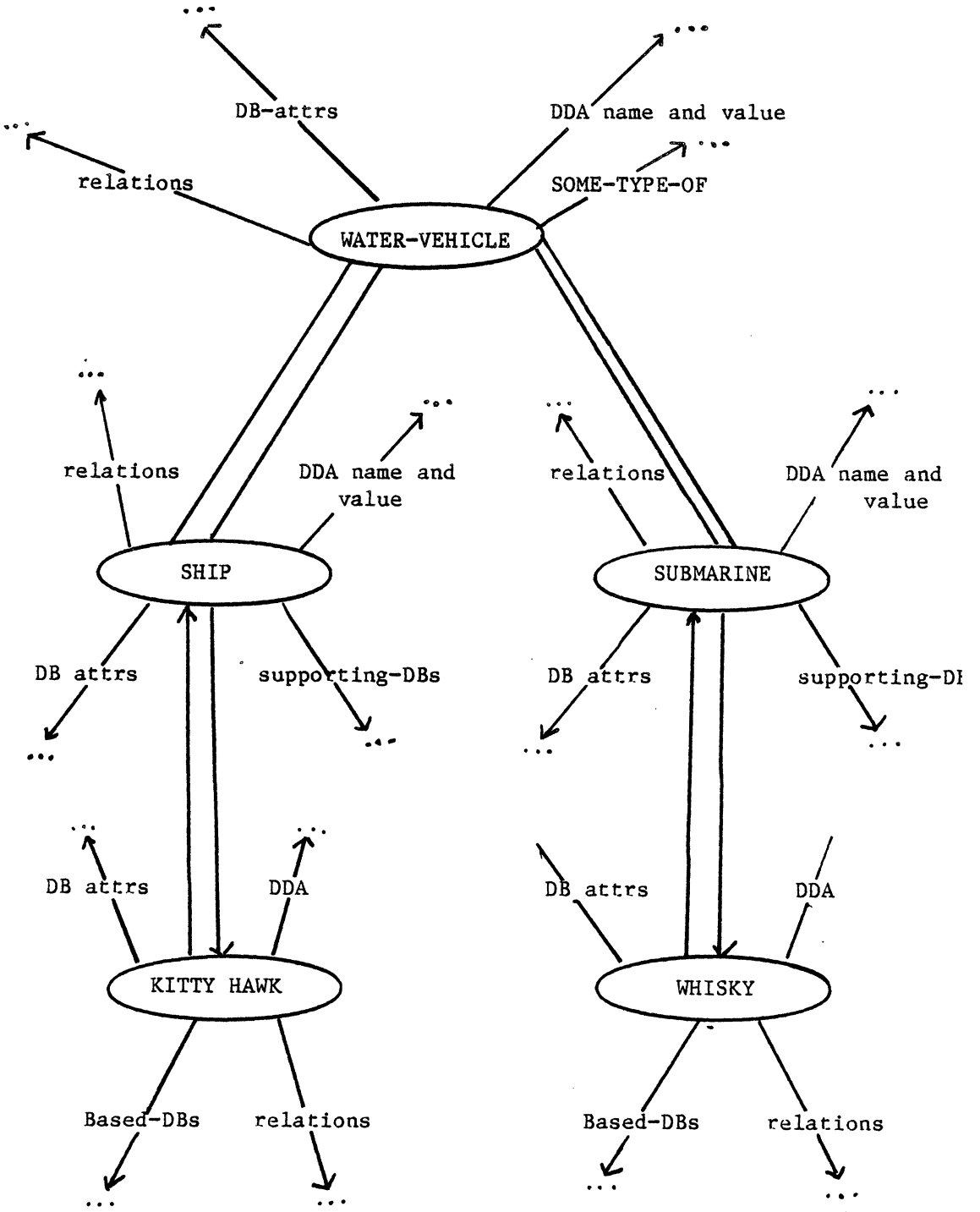


FIGURE 4.4.5 Relevant Knowledge Pool
for Category "class difference"

4.4.3 Conclusions -

The TEXT system does not embody an exceptionally sophisticated treatment of semantic questions of relevance, with the exception of the treatment of comparisons. It does, however, provide a powerful computational and practical advantage for the generation process. By partitioning the knowledge base into a subset of relevant knowledge, the system need not scan the entire knowledge base when determining what to include in the current answer. This means less work computationally for the system and it avoids the possibility of including totally unrelated material by chance. Since all that is included in the relevant knowledge pool is not necessarily included in the answer, a simpler semantic procedure can be used in constructing the relevant knowledge pool. The contents of the actual answer are further constrained by the schemas predicates and the focus constraints.

The development of a user model would mean that the relevant knowledge pool could be dynamically expanded or restricted if the user was found to need more explanation of a particular concept. In the current system, only concepts that are directly questioned are explained in detail. It is assumed that the user will ask about other concepts presented in the answer if he is unsure of their meaning.

4.5 The Dictionary

The dictionary stands as interface between the strategic and tactical component. Input to the dictionary is in message formalism; the dictionary takes a single proposition as input which consists of a predicate and its instantiated arguments (see Chapter 2.6 for discussion of message formalism). Dictionary output is the deep structural representation for the English sentence to be generated, specified in Kay's [KAY 79] functional notation (see Section 4.6.1 for details on the grammar formalism). The dictionary's task, therefore, involves: 1) the association of English words for tokens in the input proposition, and 2) the selection of an deep structure based on the predicate of the proposition. Note that the predicate of a proposition corresponds to the verb of the generated sentence and the verb of the sentence dictates its deep structure (e.g. - whether it takes 0, 1, or 2 objects, a complement, etc.). Once the verb has been selected, the semantic arguments of the structure are filled in with the instantiated arguments of the predicate.

The use of a dictionary for this purpose was based on McDonald's design of the linguistic component [MCDONALD 80]. A separate component is used rather than assuming that the message formalism is already in the deep structure representation because there may be cases where the choice of referents for the same message will vary depending upon the situation in which the message is used. In other words, the use of particular lexical items require decisions to be made about the best possible choice given the circumstances (e.g. -listener, previous

discourse, etc.). Similar decisions may need to be made about the deep structure, or verb, of the generated sentence. In the TEXT system, these decisions have been simplified since no research was done on referential choice. Although a data-type instance may translate into a different syntactic category depending upon the proposition it was used in and its argument position, the words used for each category remain the same across every situation of use. Knowledge about the user and the previous discourse is not used (or available) to select different lexical items. Use of a dictionary component, however, allows for easier extension in this area if this information were made available.

4.5.1 Design -

The flow of control in the dictionary is initiated by an input proposition. The entry for the proposition predicate is accessed. That entry selects the verb and calls dictionary entries for each of the instantiated arguments of the predicate. Each of those entries may, in turn, call other entries if needed. After entries for each token in the proposition have been accessed, the complete deep structure is returned and the tactical component invoked.

4,5.1.1 Entry Structure -

Each entry in the dictionary is actually a function. The function name is the entry key. Since the number of arguments for an entry may vary from one call to the next, the first step in every entry is the declaration and assignment of the entry variables. Tests are then

performed on the entry variable values and an appropriate structure, including lexical items, returned. Note that, to fill in all lexical items in the structure, another dictionary entry may have to be accessed.

A very simple dictionary entry for the entity GUIDED is shown below. It takes no arguments (therefore no setting and assignment of variables is made) since the translation for the entity is always the same. Note that the lexical items are not simply returned as a string; each lexical item is assigned to its syntactic category and thus, a portion of the entire underlying structure of the sentence is returned.

```
[GUIDED (lexpr (x)
          (prog (nil)
               (return '((adj === guided)
                         (n == projectile]
```

A slightly more complicated entry is shown below. The entry for the distinguishing descriptive attribute (DDA) value SURFACE is shown. It takes a single argument, a marker from the calling entry which indicates what type of syntactic translation is called for.* The entry tests the value of the marker to determine what kind of translation to provide. Nil is returned if the value can not be translated using the given form.

*The type of syntactic category required depends upon the argument's function in the sentence. One function may call for a noun phrase and another, an adjective.

```
[SURFACE
 (lexpr (x)
        (prog (marker)
              (setq marker (arg 1))
              (return
               (cond ((equal marker 'adj)
                      '((adj === surface)))
                     ((equal marker 'mod)
                      '((adj === surface-going)))
                     ((equal marker 'n)
                      '((n === surface)))
                     ((equal marker 'pp)
                      '((pp ((cat pp)
                             (prep === on)
                             (n === surface)
                             (article === def)))))
                     ((equal marker 'verb) nil))
                     (t nil]
```

A single calling function, entry-for, is used to access entries in the dictionary. Entry-for searches the dictionary, an associated list, for its first argument. It then applies the function associated with its first argument (the entry key) to its remaining arguments and returns the result. This function calls entry-for to access other entries in the dictionary when a variable in the translation exists. Thus, the entry for the predicate attributive is a function which selects the verb have (for certain uses of attributive) and calls other entries using entry-for for each of the predicate's arguments since these arguments do not remain constant for each use of the attributive predicate.

4.5.1.2 General Entries -

Entries common to all instances of a single data-type are used wherever possible. These entries can be used when the same decisions have to be made for each instance of a data-type. A single entry is written, encoding these decisions, which calls another entry to fill in the word, or words, that differ.

The data-type entity, for example, always translates as a noun phrase. A certain number of decisions are common to the translation of any entity: Does it have any modifiers? Should those modifiers be translated as relative clauses or as adjectives?* Is this a case of a list of entities, in which case conjunction is required, or is a single entity being translated? Does this use of the entity require indefinite or definite reference? Etc. Rather than repeat these decisions under the entry for each entity, a single entity entry is used which calls an entry for the particular entity being translated. The separate entity entries, therefore, encode no decisions, and simply return the lexical entry for the noun constituent of the entity (in some cases, translation of an entity also calls for a modifier, as was the case in the example given above).

General entries are also used for the translation of database

*In the TEXT system, a restrictive modifier (the modifier restricts the class of items the entity refers to) calls for the use of a relative clause. A non-restrictive modifier (the modifier describes all instances of the class that the entity denotes) results in the selection of an adjective.

attributes. Database attributes often appear in a list since two or more attributes are usually discussed at a time. One test common to all translations of database attributes, therefore, is on the necessity of conjunction. All translations of database attributes must also test whether an attribute is a topic or a leave in the topic hierarchy (see Section 4.3) since a different structure will be used for the two cases. If several, but not all, attributes under a topic (the topic is termed duplicate attribute in this situation) are described, a parenthetical is used to list the sub-nodes after the topic. The message formalism, the dictionary output, and the eventual translation for this case is shown below.

Dictionary input for duplicate attribute:

```
(duplicate (FUEL (FUEL__CAPACITY FUEL__TYPE)))
```

Dictionary output for duplicate attribute:

```
[ (n === FUEL)
  (parenthetical ((conj === and)
                  (np ((n === FUEL__CAPACITY)))
                  (np ((n === FUEL_TYPE]
```

Eventual translation:

The ship has DB attributes DIMENSIONS, FUEL (FUEL CAPACITY and FUEL TYPE), ...

Currently in the TEXT system, the knowledge base token for each database attribute is used directly in the produced sentence, rather than translating each attribute separately. Translation of a set of database attributes, therefore, only uses the general entry in the

dictionary and separate entries for each attribute are not needed. This shortcut was taken since the knowledge base tokens for attributes are English-like (e.g. - SPEED_INDICES, MAXIMUM_SPEED, FUEL_CAPACITY). The extra time needed to add entries for each attribute in the topic hierarchy to the dictionary would result in some added fluency in the text but it should be noted that such fluency would result solely from a hand-encoded effort.

Database attribute/value pairs are also translated using a general entry in the dictionary. An example of the use of attribute/value pairs within a sentence is shown below. Each attribute/value pair is translated as "<attr> of <value>" (dictionary output for the given translation is also shown) and inserted within a conjunction if more than one pair is present. Database values are used as lexical items in the sentence as were database attributes. Again, a slightly smoother translation would result if dictionary entries provided translations for all character values in the database (e.g. consider "FLAG of BLBL" vs. "of US nationality" since BLBL stands for the country "US"). This would, however, be a large and tedious task.

Dictionary input:

```
(attributive db AIRCRAFT-CARRIER (FLAG BLBL) ....)
```

Dictionary output:

```
[(n === FLAG)
 (pp ((prep === of)
      (n === BLBL))]
```

Eventual translation:

All aircraft carriers in the ONR database have FLAG of BLBL, ...

4.5.2 An Example -

Tracing the translation of a proposition through the dictionary will clarify the process. Consider the following proposition, which attributes the DDA (data-type indicated by def) TARGET-LOCATION = UNDERWATER to the TORPEDOE (GUIDED is present in the proposition since it is the TORPEDOE's parent and the given attribute distinguishes the TORPEDOE from all other children of GUIDED. It is not used in the translation).

```
(attributive def TORPEDOE GUIDED TARGET-LOCATION SURFACE-AIR)
```

The translation process begins by accessing the entry under the predicate attributive, passing the proposition as argument. The attributive entry tests the data-type used to make the attribution and accesses a second entry based on that type (In this case, the type is distinguishing descriptive attribute. For a complete list of predicate data types see Chapter 2.6). During this stage, a check is also made

on whether the particular DBA can be translated as a verb. TARGET-LOCATION has no verb translation (FUNCTION is an example of a DDA that can be translated as a verb)* The entry accessed is attr-def and the discrimination net used to get there is shown in Figure 4.5.1.

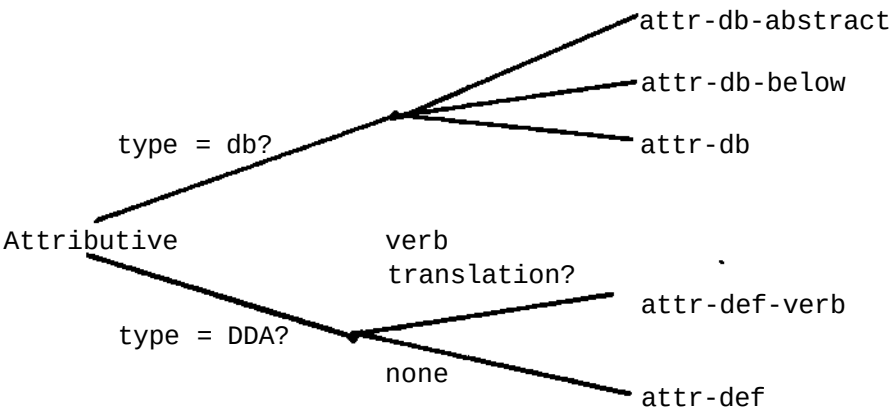


FIGURE 4.5.1 Discrimination net for Attributive entry

In the entry attr-def, the verb "have[11] is selected for the sentence. The semantic cases of "have" are filled by arguments in the proposition. Protagonist (prot) is filled by the entity TORPEDOE (since this is the item to which information is being attributed) and the goal is filled by the DDA name-value pair, TARGET-LOCATION = UNDERWATER. The deep structure for the sentence having been determined, the translation for the sentence cases are accessed by looking up the entries for the corresponding proposition arguments in the dictionary. The FD constructed at this stage is shown below. Lists in this FD headed by entry-for are slots that will be filled in by the value returned after applying the function located in the dictionary under the first argument of entry-for to the remaining

arguments.

```
[(verb ((v === have))
(prot (entry-for entity TORPEDOE))
(goal (entry-for TARGET-LOCATION UNDERWATER]
```

The value for the protagonist is obtained by applying the function associated with entity in the dictionary to the argument TORPEDOE. As discussed in Section 4.5.1.2, a general entry (entity) is used for translating all entities. In the entity entry, a test for conjunction is made and fails since only one entity, the TORPEDOE, is passed. No modifiers are passed and therefore no adjectives or relative clauses are added to the result. These decisions having been made, the entry for TORPEDOE is accessed and the lexical entry returned. The modified FD, with the value for protagonist filled in, is shown below.

```
[(verb ((v === have)))
(prot ((n === TORPEDOE)))
(goal (entry-for TARGET-LOCATION UNDERWATER n)]
```

The entry under TARGET-LOCATION is accessed next with DDA value UNDERWATER and marker n as arguments. Its translation depends upon a variety of factors. The first is its function in the sentence. A DDA can be used as a modifier of an entity or as an np itself. In this case, the DDA functions as the goal of the sentence, an np (the reason for passing marker n). Secondly, the DDA name can be translated without the value or with it. In this case, the value is given and therefore must be taken into account in the translation. Given these decisions, the DDA name is translated as the head noun and the DDA value as its modifier. The head noun is selected in this entry, but

since more than one value may be passed as argument (e.g. - the MISSILE has TARGET-LOCATION = SURFACE-AIR), the entry for the particular value passed is called to determine the type of modifier (i.e. adjective vs. prepositional phrase) as well as the lexical items used. The modified FD is shown below and the choices that were made to arrive at this FD are shown as a tree in Figure 4.5.2.

Since the DDA name is being used in goal position and has not been previously mentioned, the indefinite determiner is selected. Although focus information could be used to make this selection, no explicit checking of focus is made here. Instead, use of indefinite is always made for this particular construction, since it is assumed that this proposition would not be generated if target location had already been mentioned.

```
[(verb ((v === have)))
(prot ((n === torpedo)))
(goal ((entry-for UNDERWATER 'adj 'pp)))
        (article === indef)
      (n === location)
      (adj === target) ]
```

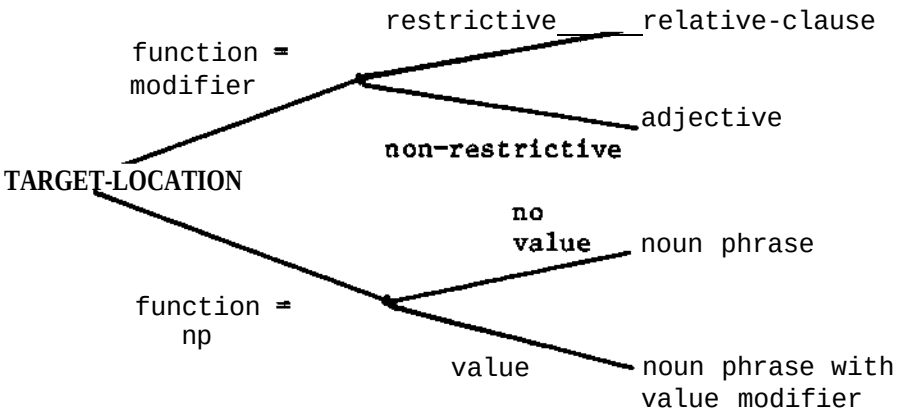


FIGURE 4.5.2 Discrimination net for entry TARGET-LOCATION

All DDA value entries translate their value according to its function in the sentence, as was the case for DDA name entries. In this case, the second and third arguments indicate that the value is to function as modifier, and more specifically, as an adjective if possible (the preference since it would entail less text) and a prepositional phrase if not. UNDERWATER translates as an adjective and this result is added to the FD to produce the final dictionary output shown below. The choices taken in the entry are shown in Figure 4.5.3.

Dictionary Outputs

```
[(verb  ((v «« have)))
 (prot  ((n --- torpedo)))
 (goal ((adj === underwater)
        (article === indef)
        (n === location)
        (adj -== target))) ]
```

Eventual Translation:

The torpedo has an underwater target location.

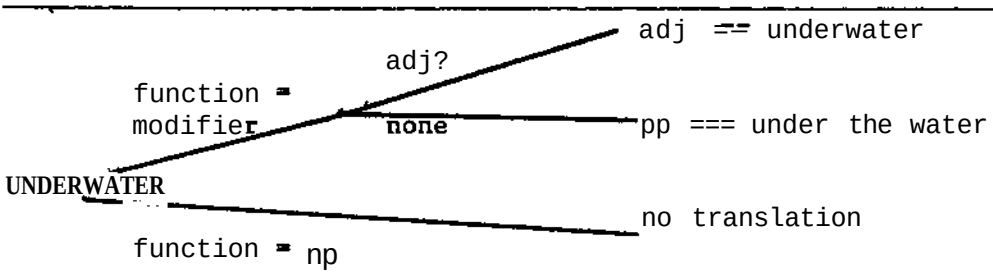


FIGURE 4.5.3 Discrimination net for UNDERWATER entry

4*5-3 Creating The Dictionary -

Creating the dictionary is a tedious, time-consuming task since i must be generated by hand. Furthermore, it acts as a bottleneck in th generation process since a message in internal representation cannot b generated in English until the English translations for the tokens i contains have been entered in the dictionary. The dictionary als limits the fluency of the generated text. If the translations o knowledge base tokens are not well thought out, the resulting text wil be awkward and unnatural. Thus, the larger the range of generated texi

(i.e. - the more messages that can be constructed), the more time will have to be invested in writing the dictionary. In order to use more sophisticated lexical items in the generated text, more thought must be invested in the creation process.

Planning the dictionary involves systematic analysis of each data-type in the knowledge base and the message formalism. The first step involves determining the unmarked semantic structure corresponding to each predicate used in the message formalism. For predicates that can be instantiated by more than one type of data in the knowledge base (see Chapter 2.6), a semantic structure must be determined for each instantiation type. Following this stage, each data-type in the knowledge base (see Chapter 4.3 for a description of the different data-types used in the TEXT system) must be analyzed to determine: 1) its possible functions in the sentence (which will be multiple since it may be used by different predicate translations in different ways), 2) what syntactic category should correspond to each of its functions, and 3) what lexical items should be used within the categories.

A set of interactive functions were developed for the TEXT system to aid in the creation of the dictionary. Each data-type was analyzed separately to determine the variety of sentential roles it could fill and then interactive functions were written to prompt for the conditions for each role and the lexical items to be used in each case. The functions use the responses given to construct the entry in proper format. Automating the process in this way is particularly helpful since the functions can scan the knowledge base for each instance of

the data-type and prompt the dictionary designer for each translation. This relieves the designer of the tedious task of scanning the knowledge base by hand and ensures that a translation for each instance will be included.

Interactive functions were written for the DDA names, DDA values, the supporting database attributes, entities, relations and for the predicates. Since database attributes and database attribute value pairs do not require lexical translation, a single general entry could be written to handle these cases. Each function prompts the user for the following parts of the entry: 1) its parameters, 2) the setting of its parameters, 3) conditions for translation (these may be omitted if the conditions are the same for each instance of a data-type), and 4) the lexical translations for each condition. The user is prompted for these values for each instance of the data-type in the knowledge base.

The interactive functions used for creating the entries for DDA name request translations for each sentential role the DDA could serve. As mentioned earlier, these are non-restrictive modifier, restrictive modifier, noun phrase with modifying DDA value, or noun phrase without modifying DDA value. Since it is assumed that the desired sentential role can be passed to the entry as argument (the predicate entry assigns instantiated arguments of the predicate to semantic arguments in the deep structure and therefore, can pass this information on when the DDA name entry is called), the DDA name entry builder need only prompt for translations and not conditions. The translations that are

needed are relative clause, adjective, and noun phrase* Note that any of these may depend in part or in whole (e.g. - adjective) on the translation of the value that is passed. In such cases, the designer can enter the function entry-for and its arguments instead of a syntactic category and lexical item.

The function first notifies the user of the DDA name currently being worked on. Separate functions are then called for each of the categories that the DDA name can be translated as. Each of these functions knows about the possible constituents of the respective category and prompts the user for lexical values for each of these. The noun-phrase building function, for example, is aware that a noun phrase can consist of a head, optional modifiers, determiner, and number. Complex constituents of the noun phrase, such as relative clause and prepositional phrase, are built by their own functions which prompt the user for their constituents. Note that for any constituent, the user can enter either a value or a function. If a function Is entered, it will be evaluated at the time the entry is accessed to produce the constituent value. Functions can be used either for calling other entries in the dictionary or for testing the arguments of an entry to determine the appropriate value.

A predicate entry consists of tests on its type and calls to separate entries which construct the deep structure for the sentence which corresponds to the particular predicate type. The entry is constructed by prompting the user for the tests on type as well as the entry-name and its arguments which' are needed for each type. As

discussed in Chapter 2.6, a predicate may be instantiated by more than one data-type in the database. The attributive predicate, for example, may be instantiated by attributing database attributes to an entity or by describing an entity's distinguishing descriptive attribute. Each of these types translates into a different deep structure for the sentence. A separate entry is written for each predicate type for clarity.

In constructing the predicate type entry, the user is prompted for the verb and the semantic cases of the sentence. The predicate type determines the verb of the sentence and thus, a lexical value is usually entered for this constituent. The protagonist and goal of the sentence are most often filled by the instantiated arguments of the predicate and thus, a call to the entry for the appropriate predicate argument is usually assigned to these slots in the deep structure. The user is also prompted for a sentential adverb. Textual connectives are associated with the underlying predicate of the sentence. Thus, the predicate particular illustration uses the sentential adverb for example while the inference predicate triggers the use of the connective therefore. Currently in the TEXT system, if a sentential adverb is associated with a predicate type in the translation, it is always used in the sentence. Some more sophisticated uses of textual connectives might involve testing focus information across sentence boundaries. A connective would be required when a sudden shift in focus was made and would not be used in other cases (see Chapter 3, Section 3.3.2 for a discussion of this phenomena).

4.5.4 Conclusions -

Hand-encoding of the dictionary requires a considerable amount of time and effort. The dictionary designer, moreover, can encode as complex (or as ad-hoc) translations as he likes. One of the problems along these lines in the TEXT system implementation was the separation of semantic and syntactic information about the sentence structure. Ideally, the dictionary should use only semantic terms in constructing the deep structure. Although Kay's formalism allows for the input to be specified in purely semantic form, the TEXT system grammar wasn't developed fully enough to handle this. To accomodate this lack, the dictionary had to specify some of the sentence structure in syntactic form (i.e. - the use of adjective and noun in dictionary output).

Some steps for automating the creation of the dictionary were taken in the TEXT system by writing interactive functions which prompt the designer for lexical values. Additional work in this area needs to be done if generators are to be made portable. Research on reasoning about referential choice is part of the process needed to increase the sophistication of this component (e.g. - [APPELT 81], [COHEN 81P]). If the dictionary has access to a user model and pragmatic information about lexical choice, the designer would have to do less work in selecting appropriate lexical items for the translation.

4.6 The Tactical Component

The tactical component takes as input an internal representation of what's to be said and uses a grammar to translate that representation into English. In the TEXT system, input to the tactical component is a deep structure representation of the message. The tactical component determines the surface ordering of the constituents and exactly which grammatical constructions are to be used. Since the tactical component was not the main emphasis in this dissertation, it does not include as complete a coverage of English grammar as might be desired. Full use of pronominalization, ellipsis, conjunction, and other sophisticated linguistic devices, for example, were not implemented. The tactical component is needed, however, to illustrate that the text planning devices used in the TEXT system are successful. Some isolated uses of more sophisticated linguistic devices were, therefore, implemented to show that the kind of information provided by the text planning components is sufficient for making decisions about their use. The tactical component uses a functional grammar based on Kay's formalism [KAY 79]. It was selected because of the ability to directly encode in the grammar tests on focus and theme for determining which syntactic construction to use, information which the strategic component supplies for this purpose.

4.6.1 The Grammar Formalism -

The basic unit of Kay's grammar is the attribute value pair. Attributes specify categories (syntactic, functional, or semantic) and values specify the legal fillers of those categories. Attributes are symbols and denote categories such as noun phrase, noun, protagonist, goal, subject, etc. Values may be either symbols or sub-grammars, which also consist of attribute value pairs. These basic building blocks are augmented by a discrete set of connecting devices which allow for the representation of complex syntactic structures.

A grammar is termed a functional description (FD). A single FD is used to encode the entire sentence grammar and it contains smaller FDs which describe the grammar of sentence constituents. For example, an FD for a sentence grammar might consist of three attribute value pairs: subject, verb, and object. FDs are represented diagrammatically by square brackets. Figure 4.6.1 shows a sample sentence grammar. Note that the sub-grammars are not specified and the grammar is not complete as it stands.

$$S = \begin{bmatrix} \text{SUBJ} = [\ \text{subject grammar}\ ] \\ \text{VERB} = [\ \text{verb grammar}\ ] \\ \text{OBJECT} = [\ \text{object grammar}\ ] \end{bmatrix}$$

FIGURE 4.6.1 Sample Sentence Grammar

FDs may contain alternatives which specify that a particular category may be formed in more than one way« Another version of the sentence grammar, shown in Figure 4.6.2, specifies that the object is optional and that a sentence may therefore contain either two or three constituents. Alternatives are represented by curly braces. Note the use of the special symbol NONE, which indicates that this alternative will only be taken if there is no object in the input and if taken no object occurs in the output.

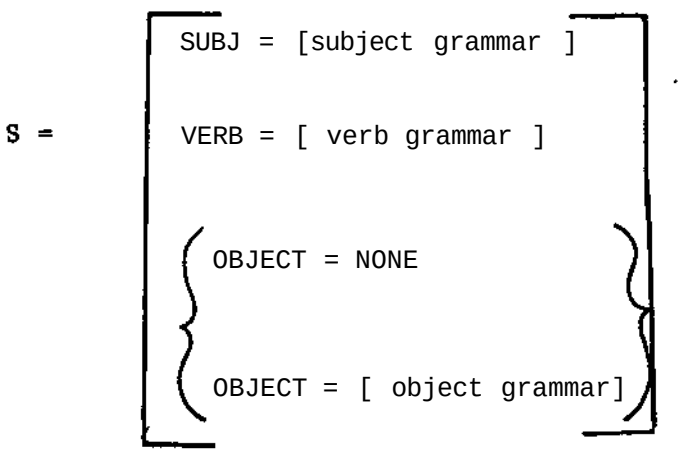


FIGURE 4,6.2 Sample Sentence Grammar II

Patterns are used within an FD to specify the surface order of the constituents in the resulting string. A pattern is a list of attribute names which specifies the left to right order of the constituents. Since the value of an attribute may be another FD, which may in turn contain a pattern, the process of linearizing an FD using a pattern is a recursive one. An entry in a pattern list can correspond to a single word in the resulting string or to a string of words specified by another pattern. Patterns may contain two special symbols in addition

to attribute names. Dots (...) are used to specify that 0 to n constituents from other patterns may appear between two attributes and pound-signs (#) specify that exactly one constituent from another pattern must occur between two attributes. These two symbols are used in handling alternatives which encode different orderings of the same constituent.

For example, dots are often used to allow for the inclusion or exclusion of optional constituents such as OBJECT in Figure 4,6.2 above. Consider Figure 4,6.3 below which is a modification of the grammar shown in Figure 4.6.2 to contain two patterns. Unification of the "patterns will result in (SUBJ VERB OBJECT) if an object exists in the input or the pattern (SUBJ VERB) if no object exists in the input.

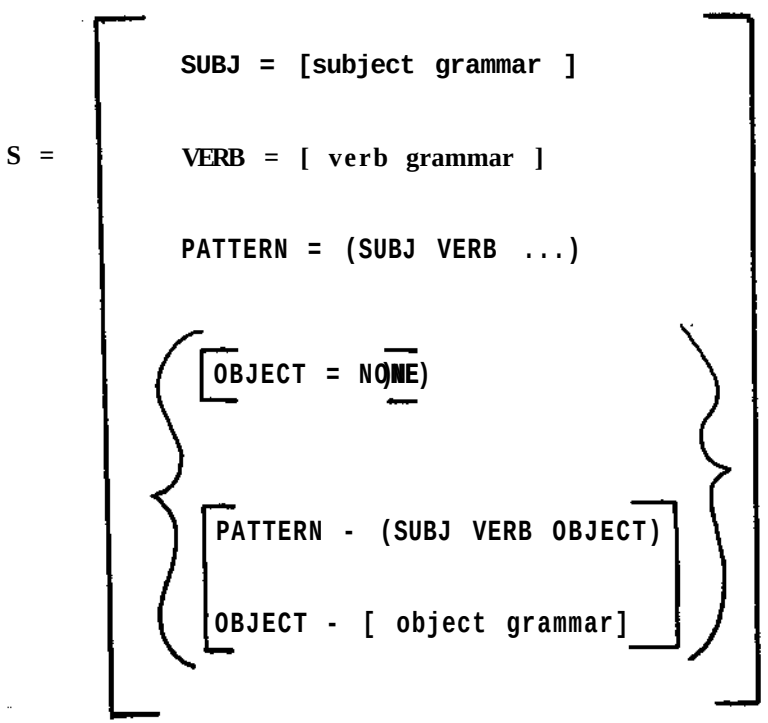


FIGURE 4.6.3 Sample Sentence Grammar III

Paths are used in Kay's formalism to refer to the value of one constituent from another constituent. A path is represented by angle brackets (<>) and specifies a list of attributes. The sample path <a1 a2 ... an-1 an> points to the value of attribute an in the value of attribute an-1, and so forth. Up-arrow (^) is a special symbol in a path that can be used to point to the FD containing the current FD (i.e. - upper-level FD). Paths can be used, among other things, for number agreement, person agreement, and traces. In Figure 4.6.4 below, the path is used to indicate that the number (NUMB) of the verb is equivalent to the number of the sentence subject (<^SUBJ NUMB>).

$$
S = \left[\begin{array}{l}
\text{SUBJ} = \left[\begin{array}{l}\text{CAT} = \text{NP} \\ \text{NUMB} = \text{ANY}\end{array}\right] \\
\text{VERB} = \left[\begin{array}{l}\text{CAT} = \text{VERB} \\ \text{V} = [\text{CAT V}] \\ \text{NUMB} = \langle \hat{} \ \text{SUBJ NUMB}\rangle\end{array}\right] \\
\text{OBJ} = [\text{CAT NP}]
\end{array}\right]
$$

FIGURE 4.6.4 The Use of a Path

Description of some special symbols used in the grammar will be helpful before describing how it is used. Attribute names include the following symbols, as well as traditional category names :

1. cat (category): the value indicates the syntactic category of the FD.
2. pattern: Its value is the list of elements determining the surface order.
3. lex (lexical entry): Its value specifies a particular lexical element or set of elements.

Two special symbols are used for values: any and none. Any is a

wild-card and indicates that the value of the attribute can be any non-null value. None specifies that the attribute must not occur in the output. A sample grammar can now be given for a noun phrase. The grammar allows for an optional adjective and an optional prepositional phrase:

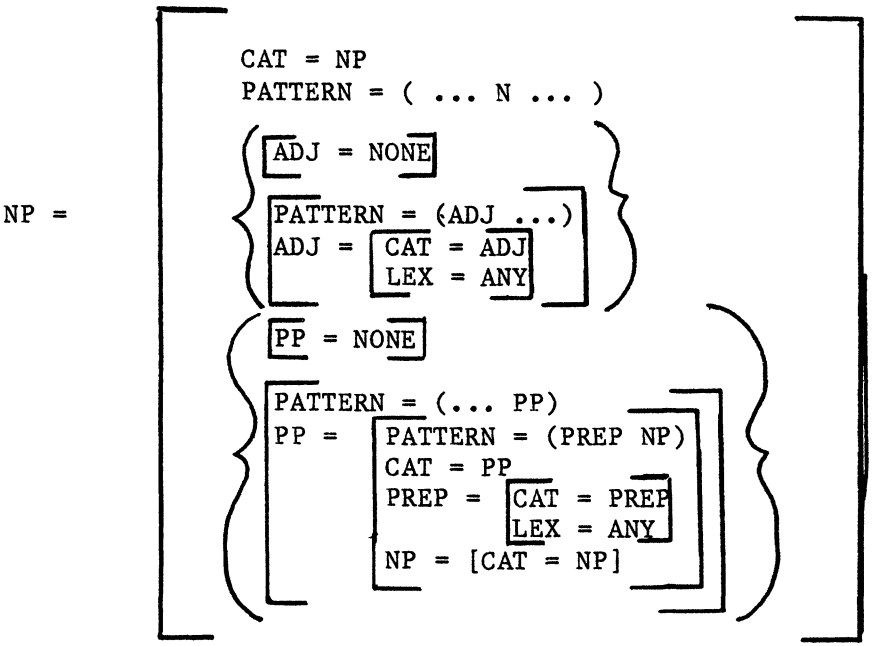
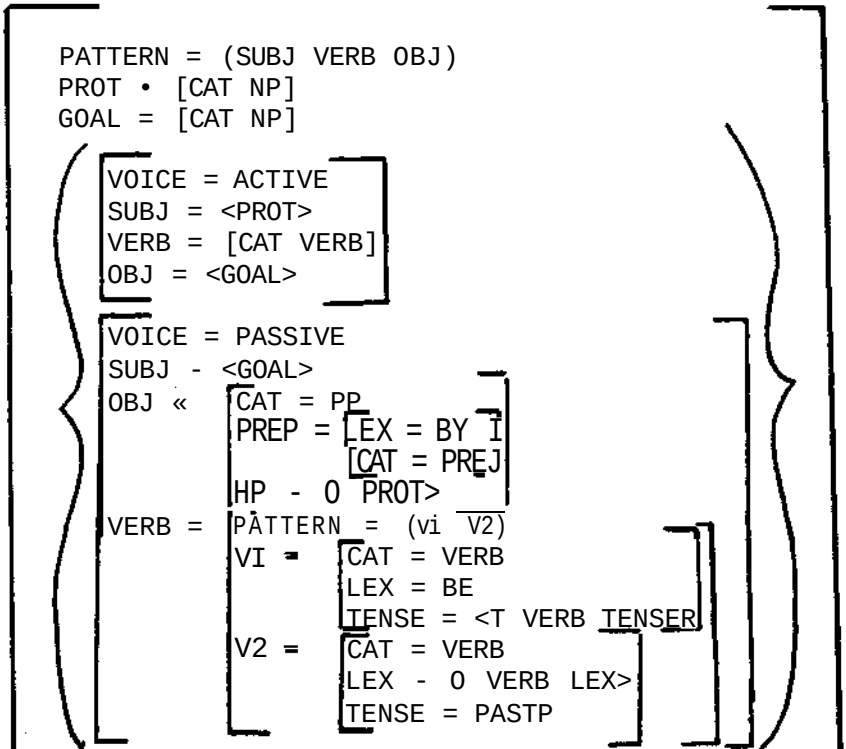


FIGURE 4.6.5 Sample NP grammar

4.6.2 A Functional Grammar -

Kay's grammar is termed "functional" because it assigns equal status to terms which describe the functional roles elements play in a sentence (e.g. - protagonist) and to terms which describe the syntactic category an element belongs to (e.g. - noun phrase). An FD may describe legal strings of the language using both these terminologies. In the TEXT system, Kay's functional categories

protagonist (PROT), sometimes called agent in case formalisms, and goal (GOAL), sometimes called object, were adopted. Beneficiary (BENEF) is used for verbs which take an indirect object. A sentence grammar is defined for passive and active sentences by stating that the subject of the sentence is the protagonist if the voice is active and the sentential subject is the goal if the voice is passive. An example of this simple grammar for actives and passives is shown below in Figure 4.6.6. Note that verbs in this example do not take an indirect object. This was done to simplify the example. Simplification was also achieved by . assuming that a protagonist is obligatory. Each category mentioned in the FD (e.g. VERB, PP) must be defined by a grammar elsewhere.



In this grammar, the value of voice is used as a test for determining the values of subject and object. Use of this functional notation simplifies the input considerably. Assuming that the sentence grammar above, the noun phrase grammar shown in Figure 4.6.5, and unspecified verb and prepositional phrase grammars constitute the system grammar, the input need only specify the values for protagonist and goal and indicate whether the sentence is to be in active or passive form. Unification of the input with the sample grammar would result in* an identification of subject and object, the construction of the prepositional by-object when needed, and indicate the surface order of the constituents. The sample input shown below in Figure 4.6.7 would result in the sentence "The old man was bitten by the dog.[11] when unified with the sample grammar.

```
PROT » [ADJ === OLD]
       [N === MAN]
VERB » [F=== BITE
       [TENSE = PAST]
GOAL =[N === DOG]
VOICE = PASSIVE
```

FIGURE 4*6.7 Sample Input

Kay's formalism also allows for the specification of concepts such as topic and comment directly in the grammar. This feature is particularly attractive since the assignment of values to these categories can be used to determine the order of constituents within the sentence. In such a case, the grammar would indicate that the subject of the sentence is the topic. This scheme means that the

process which creates the input need not be cognizant of the difference between syntactic concepts such as active and passive. Use of topic-comment articulation within the grammar itself means that the input can be specified in functional and semantic terms and that the grammar can use these functional roles in determining the values of the syntactic categories and their ordering within the sentence. Figure 4.6.8 shows how the input given in Figure 4.6.6 would have to be changed to specify the topic.

```
[PROT = [ADJ === OLD
         n === MAN]
 VERB = [ V === BITE
         TENSE = PAST]
 GOAL = [ N === DOG ]
 TOPIC = <PROT>]
```

FIGURE 4.6.8 Sample Input with Topic

This type of formalism is particularly appealing for the TEXT system since the output of the strategic component contains focus information and argument assignments for predicates, but embodies no syntactic information. Focus information can be used in the same way as topic/comment articulation to select between syntactic constructions. Currently, tests on focus information are made in the dictionary and the sentence voice selected at that point; input to the tactical component, therefore, looks like the sample input shown in Figure 4.6.7. Steven Bossie will be working on incorporating these tests into the grammar for his master's thesis. Chapter 3.3.2 describes exactly how focus information is used in the TEXT system.

4.6.3 The Unifier -

A sentence is produced in Kay's formalism by unifying the input, which is specified in the same formalism as the grammar, with the grammar. The input to the unifier is a deep structure representation of what is to be generated. The output is a surface syntactic representation of the sentence which is linearized using the patterns it contains.

During the unification process, variables (values of any) are replaced by values from the input and alternatives in the grammar are eliminated. The process involves unifying the value of each attribute in the grammar FD with the value of the attribute of the same name in the input FD. If the grammar value is an alternative, all options are unified and the first successful result taken. If either value is a symbol, then unification succeeds when the two values are equal, when one value is a wild card (any) and the other value is non-null, or when either of the values is nil. If both values are FDs then the two FDs are unified. If an attribute occurring in the grammar does not occur in the input, the attribute and its value are added to the result. In all other cases, unification results in failure. After the FD is unified with the grammar, each constituent of the FD is unified with the appropriate sub-grammar. Unification is a fully recursive non-deterministic process. For further clarification on Kay's unification process, see [KAY 79]. Unification in the TEXT system was modeled after Kay's design, but liberties were taken in solving problems peculiar to the TEXT system.

4.6.4 The TEXT System Unifier -

The unifier for the TEXT system was designed and partially implemented by Steven Bossie [BOSSIE 82]. Some of the special features of the TEXT system unifier which depart in concept from Kay's include the ability to handle unattached attributes in the input, the treatment of gapping, and the use and implementation of paths in the grammar. A brief description of each of these features is given followed by an example of the unification procedure* For more details on the TEXT system implementation of the unifier, see [BOSSIE 82].

The ability to handle unattached attributes in the input means that the input deep structure representation of the sentence can be less well-defined than would otherwise be required. The looser specification of exactly where constituents are attached means less work needs to be put into the dictionary (which is hand-encoded) where the translation from message formalism to deep structure is made. This feature is particularly useful for the description of noun phrases. The fact that noun phrases can take any number of adjectives is described recursively in the grammar. That portion of the noun phrase following the determiner is described by the rule: NNP -> ADJ NNP / NOUN. In functional notation:

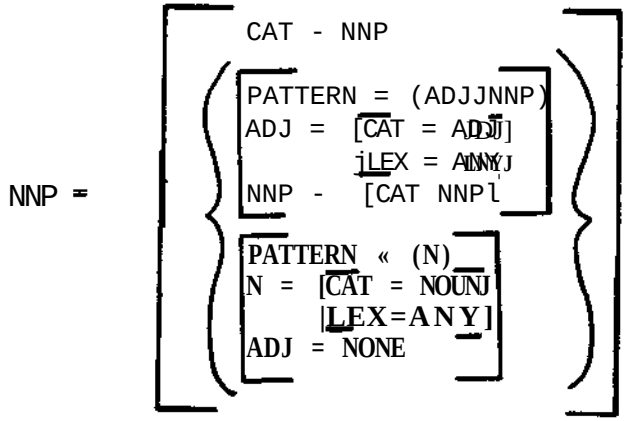


FIGURE 4.6.9 Encoding Multiple Adjectives

If more than one adjective is desired in the output, each adjective need not be attached to its NNP category in the input. Instead, adjectives can simply be listed as part of the containing noun phrase. During the process of unification, the nodes corresponding to NNP are added to the structure.

Recursion on the category NNP stops when all adjectives in the input have been used. Figure 4.6.10 shows some sample input along with the output that would be generated by the unifier if the given input were unified with the grammar shown in Figure 4.6.9. Note that == is used in the input to abbreviate the attribute value pair LEX = <value>.

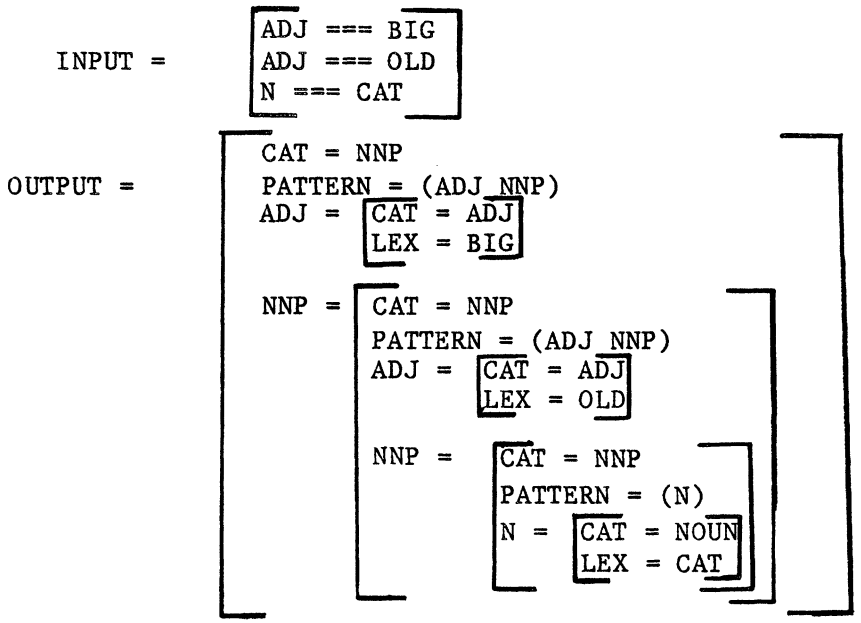


FIGURE 4.6.10 Input and Result of Unification

The implementation of gapping in the TEXT system departs from Kay's design (see [KAY 79]) for reasons of expediency. In the TEXT system, the feature "(gap +)" (an attribute value pair) is added to the FD of any constituent that corresponds to a gap in the final sentence. Since a gap is a hole that would have been filled by some other constituent in the sentence, a path is also added to the FD as the value of the attribute. The constituent that the path points to is used when resolving questions of number agreement. Although the attribute denoting the constituent that is gapped appears in the pattern, the linearizer checks for the presence of the gap feature before linearizing the value of any constituent in the pattern. If the gap is present, nothing is added to the string.

The use of paths, as specified by Kay, was found to be incompatible with the use of recursive structures (such as the NNP above) and the loose specification of attributes. Kay uses the up-arrow (^) to specify the FD in which the current FD is embedded. If a path is used in the input to specify the value for a gap occurring in a relative clause, it is unclear how many levels up (or down) in the recursive structure built by the unifier, the NP occurs. For example, in the sentence "The old man who walks by my house every day.", the gap acting as subject of the relative clause is co-referential with the noun phrase "the old man". The structure for the noun phrase could be built by the unifier in several ways. The unifier may group the head noun and relative clause under an nnp which is modified by an adjective. Or, the unifier may group the adjective and the head noun under an nnp which is modified by a relative clause. In one case three up-arrows are required to refer to the head noun and in the other case, only two. When designing the input, there is no way of knowing which of these structures will be built, and therefore how many up-arrows to include in the input.

To accommodate for this phenomena, a *up-arrow (*^) is used, which refers to the closest higher level FD which contains the attribute following *up-arrow. *Up-arrow (multiple upward path) indicates that the search is made upwards through the FD until the desired attribute is found. The same problem can occur when following a path downward through an FD. For this reason, a breadth-first search through the FD is done for the next attribute in a path. This means that if the next

attribute in the path occurs in the current level FD, processing of paths will proceed as Kay suggests* If it is not in the current level FD, a search for the attribute is made through the value of each attribute in the current level FD.

**6«5 Unifying A Sample Input With A Sample Grammar -

A subset of the grammar used for TEXT is shown in Figure 4.6*11. Note that it consists of a list of alternatives where each alternative is a different syntactic category. The lexicon is considered part of the grammar. A sample noun phrase input is shown in Figure 4.6.11. Pre-processing adds the attributes lex to the input to obtain 12, shown in Figure 4.6.11 to replace the abbreviation "===". Unification processing is controlled by the grammar. Processing begins by scheduling each alternative in the grammar for unification with the input.

The first success halts the unification of following alternatives. The first alternative in the sample grammar is the NP FD. Each attribute in the FD is unified with the input. The first attribute is cat. Its value in the input is retrieved and is found to match the grammar's value (np). The attribute and value are returned to the input and the second attribute checked. Pattern has a null value in the input and since nil and a given symbol are defined as success (Section 4.6.3), the attribute and the value, "(•• nnp)^{ff} are added to the input. An alternative is found next and each choice is scheduled for unification . The first FD ([article - NONE]) fails since the

symbol NONE only matches against nil or NONE and the value for article in the input is [lex = def]. The second alternative is then attempted.

```
I =
[cat = np
 article === def
 n === man]

I2 =
[cat = np
 article = [lex = def]
 n = [lex = man]]

Grammar =
{[cat = np
  pattern = (... nnp)
  {[article = none]
   [pattern = (article ...)
    article = [cat = article
               lex = any]]}
  nnp = [cat = nnp]]
 [cat = nnp
  pattern = (... n)
  n = [cat = noun]
  {[adj = NONE]
   [adj = [cat = adj
           lex = any]
    nnp = [cat = nnp]
    pattern = (adj nnp)]}]
 [cat = adj
  {lex = old
   lex = big}]
 [cat = noun
  {lex = cat
   lex = man}]
 [cat = article
  {lex = indef
   lex = def}]}
```

FIGURE 4.6.11 Input and Grammar

The value of pattern in the grammar "(article ...)" is matched against the value of pattern in the input "(... nnp)". Although the two patterns can be unified in any of the following ways, the first success is selected and used as the value for the input:

```
pattern = (article nnp)
          (article ... nnp)
```

The result of unifying the value of article in the grammar with the value of article in the input results in the value of lex in the grammar ("any") being replaced by the value of lex in the input ("def").

The final attribute in the NP FD is then matched against the input. Again, nnp has a null value in the input and therefore, the grammar's attribute and value are added to the input. The FD that results from this stage of unification is shown in Figure 4.6.12.

```
[cat = np
pattern = (article nnp)
n = [lex = man]
nnp = [cat nnp]]
```

FIGURE 4.6.12 Intermediate FD

Processing proceeds at this point by unifying the value of every constituent in the input FD's pattern against the grammar.* If the FD

*Attributes name both constituents and features of the grammar (e.g. n is a constituent, but numb is a feature). See [BOSSIE 82] for a description of the use of constituents.

had no pattern, unification would halt successfully with the FD shown in Figure 4*6.11 without its pattern. The value of the first constituent, article, is unified with the grammar and successfully results in the same value (it matches against the fifth alternative in the grammar and returns successfully since [lex = def] is one of the alternatives).

The value of the second attribute in the pattern is then unified with the grammar. Note that the second attribute in the pattern is nnp and its value in the Input is [cat =^ nnp]. The first alternative in the grammar fails since "cat » np" is in the grammar and "cat = nnp" is in the input. The second alternative in the grammar succeeds. After unifying the categories (cat), the first alternative of the nnp FD in the grammar is matched against the input. Since there is no adj in the input, the first attribute value pair of the alternative (adj * none) succeeds and is added to the value of the input. Pattern also succeeds and is added. The value for n. is then searched for. Although it is not a member of this particular FD, it is an unattached attribute, since it is not a constituent of the FD it is contained in, and is therefore available for use (see [BOSSIE 82] for a discussion of the treatment of unattached attributes). Thus, the value of the attribute ri in the grammar is unified with the value of *n_* in the input and results in success. The value of nnp after this stage in the unification process is shown in Figure 4.6.13.

```
nnp = [ cat = nnp
        pattern = (n)
        adj = NONE
        n = [ cat = noun
              lex = man ] ]
```

FIGURE 4.6.13 Unified nnp

The value of the attributes in this FD's pattern are then unified with the grammar (not described here). The result of the nnp unification is then added to the resulting FD (unattached attributes were removed when used) and the final FD is shown in Figure 4.6.14. This FD is linearized to produce the string "the man". For further details on the unification process see [BOSSIE 82].

```
output = [ cat = np
           pattern = (article nnp)
           article = [ cat = article
                       lex = def ]
           nnp = [ pattern = (n)
                   adj = NONE
                   n = [ cat = noun
                         lex = man ] ] ]
```

FIGURE 4.6.14 Final FD

4.6.6 Grammar Implementation -

In this section, the capabilities of the TEXT system grammar are described and example output from the system given which illustrates the use of each syntactic construction. Sentence constructions include the simple active, simple passive, and there-insertion. Sentences 1,2, and 3 are sentences generated by the TEXT system which use these three constructions:

1. The torpedo has an underwater target location. (active)
2. Torpedoes are carried by water-going vehicles. (passive)
3. There are two types of entities in the ONR database: destructive devices and vehicles. (there-insertion)

Verbs in the sentence grammar either take no object or a direct object (the grammar was implemented to handle indirect objects as well, but no use was made of this type of verb in the TEXT system.). Output from the TEXT system showing the use of verbs with no object and a single object are given in sentences 1 and 2 below (the portion of the sentence illustrating the example is underlined). Modifiers of a verb phrase can include adverbs and prepositional phrases. The use of verb complements was also implemented. Sample output for these constructions are shown in sentences 3-5, with the relevant portion underlined.

1. An aircraft travels by flying. (no object)
2. The entity has DB attributes REMARKS. (direct object)
3. A submarine is a water-going vehicle that travels underwater. (adverb)
4. A ship is a water-going vehicle that travels on the surface. (verb prepositional phrase)
5. A submarine is classified as a whisky if its CLASS is WHISKY. (verb complement)

A sentence may take a sentential-adverb, as in sentence 1 below* The use of "for example[11] is triggered by the predicate particular-illustration and is selected in the dictionary. Amplification is another predicate which can trigger the use of a sentential adverb (e.g. - also). A sentence may be followed by a sub-list as in sentence 2 below, which exemplifies what was stated in the sentence. The use of a sub-list is triggered by the constituency predicate. When the sublist is selected, a colon is attached to the preceding part of the sentence.

1. Mine warfare ships, for example, have a displacement of 320 and a LENGTH of 144« (sentential adverb)

2. There are 2 types of guided projectiles in the ONR database: torpedoes and missiles. (sentence sublist)

Noun phrases are perhaps the most complex syntactic category in the grammar. Adjectives, relative clauses, and prepositional phrases are all modifiers of a noun that can appear any number of times within a single noun phrase and were therefore implemented recursively. Relative clauses are fairly complex since they make use of paths and the gap feature. Examples of these three constructions are shown in sentences 1-3 below. A noun phrase also has several optional constituents which can appear only once, if at all. These include determiners, parentheticals, and the use of sub-lists. The input need only specify whether a determiner is definite or indefinite and the appropriate lexical item will be selected. A default definite determiner is assumed if none is specified in the input.

1. The torpedo and the missile are self-propelled guided projectiles. (adjectives)

2. All aircraft carriers in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, ... (prepositional phrases on nouns)

3. The vehicle has DB attributes that provide information on SPEED_INDICES and TRAVEL MEANS. (relative clause)

Parentheticals are used to provide further specification of a noun phrase for two reasons: providing examples and specifying members of a class. Examples are used when a noun phrase specification may not be sufficient for a user to understand. In most cases a single example is presented because the complete set is too large to list. Parentheticals are also used to list the database attributes that an entity has under a single topic in the topic hierarchy. A noun phrase is used for the topic and the specific attributes are parenthesized. The use of parentheticals for examples is shown in Sentence 1 below where only two of the sub-types of the aircraft are mentioned. Sentence 2 shows the use of a parenthetical to list the attributes of FUEL indices which the aircraft possesses.

1. Aircraft are categorized by PROPULSION (for example, jet) and FLAG (for example, US aircraft).

2. Other DB attributes of the aircraft include FUEL (FUEL_CAPACITY and FUEL_TYPE) and FLAG.

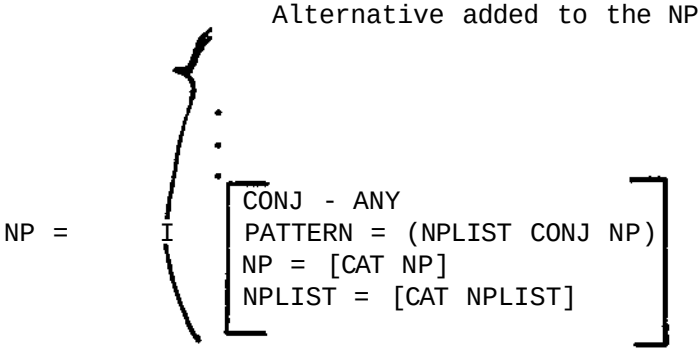
Noun phrase sub-lists are used to specify elements of a category. The category is described using a generic description, but only some members of the generic class actually participate in the relation.

These are specified by using a list construction. In the noun phrase "Jane's friends, John, Sue, and Mary,", "Jane's friends" constitutes the generic description while the list "John, Sue and Mary" specify the relevant elements. Sentence 1 below shows an example of noun phrase sublist construction from the TEXT system output:

1. Its surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. (noun sublist)

The tactical component is currently capable of using conjunction around sentences, noun phrases, and prepositional phrases. Conjunction is represented in the grammar through the use of recursive structures. Each category that allows for conjunction must be modified to contain an alternative which tests for the presence of a conjoining term. Its presence triggers the construction of two categories, np-list and np for a noun phrase, separated by the conjoining term. Np-list recurses on itself until no more nps are found in the input. The alternative added to the noun phrase FD and the additional syntactic category added to the grammar in order to produce sentences with conjunction around noun phrases are shown below in FIGURE 4.6.15

Alternative added to the NP FD



Additional category (NPLIST) added to the grammar

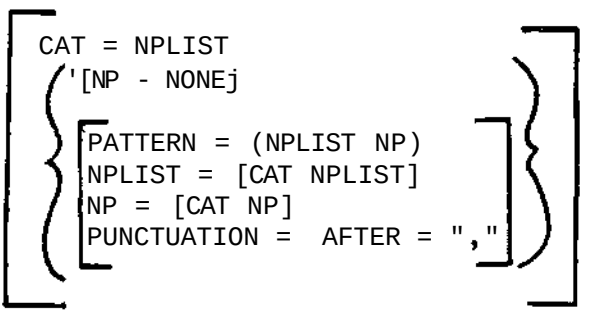


FIGURE 4.6.15 Conjunction Additions

It should be noted that the lexical value of the conjoining term must be specified by the input. "And" and "or" are the two most common types of lexical conjunction, but at the sentence level other types of conjunction may be necessary (e.g. "but", "although", etc.). Although the TEXT strategic component currently does not make any selection of these types of conjunction, the grammar could produce these types of sentences if they were specified. The use of a sublist within an NP or a sentence calls for the standard use of conjunction. Some examples of this have already been seen. Two additional examples are provided in Sentences 1 and 2 below, the first using and and the second using or.

1. All aircraft carriers in the ONR database have REMARKS of 0, FUEL TYPE of BNKR, FLAG of BLBL, BEAM of 252, ENDURANCE RANGE of 4000, ECONOMIC SPEED of 12, ENDURANCE SPEED of 30, and PROPULSION of STMTURGRD. (conjunction around nps using and)

2. The missile has a target location in the air or on the earth's surface. (conjunction around pps using or)

Future work on conjunction in the tactical component should include a more general treatment of conjunction. Currently, a test for conjunction must be added to each syntactic category for which it is needed. Although the test is fairly simple, a more general method would be to let the presence of the conjoining term trigger the use of conjunction around whatever category was present in the FD of the input.

The tactical component is also capable of producing possessives. A possessive is a constituent of the noun phrase in the TEXT grammar and is itself a noun phrase (although a limited one in the TEXT grammar since the full range of modifiers on possessives were not accounted for). Possessives affect the immediately succeeding noun phrase since it can no longer take a determiner. Sentence 1 below contains a noun phrase with possessive.

1. The missile's target location is indicated by the DB attribute DESCRIPTION ... (possessive)

Punctuation is also handled within the grammar. Sentence final periods are the only exception to this rule. If a particular syntactic construction requires the use of punctuation (i.e. - commas for lists, colons for sub-lists following sentences, parentheses for parentheticals), a punctuation feature is added to the constituent specifying the type of punctuation to be used and whether it should occur before or after the constituent. The punctuation element is concatenated to the appropriate word (the last word of the constituent if "after" is specified; the last word of the preceding constituent if "before" is specified) during the process of linearization. In Figure 5.4.14, the use of a comma was specified by the punctuation feature.

4.6.7 Morphology And Linearizaiton -

Morphological suffixes to words are added during the linearization process. A list of attributes specifying the linear order of the constituents is obtained from an FD by retrieving its pattern. The value of the first attribute of the list is then accessed. If the value is an FD, its pattern is accessed. If it is a lexical entry, the morphological routines are called, the lexical entry processed, and the resulting word added to the sentence string.

Certain word categories, such as adjectives and adverbs, need no morphological processing. Nouns, on the other hand require processing. They must be pluralized if they contain the feature "NUMB = PLUR". This requires adding "s" or "es" to the root noun provided in the input ("es is added if the noun ends in "s"). For nouns taking an irregular

plural, the plural is provided explicitly under the word's entry in the lexicon.

Verbs are conjugated as dictated by the features "TENSE" and "NUMB" in the verb FD. Present tense triggers the concatenation of "s" for third person singular only, to the root verb. Past tense triggers the concatenation of "ed" for all persons. Because of the database application, a default tense of "present", person of "third", and number of "singular" is assumed if any of these features are missing. The conjugations for irregular verbs, such as "be" and "have", are provided explicitly in the lexicon under their entries.

4.6.8 Disadvantages -

The major disadvantages of the functional grammar and unifier are implementation issues. Since the unifier is non-deterministic, the production of a single sentence form the given deep structure is incredibly time-consuming. The unifier is, *at best*, ten times slower than the strategic component. A representative processor time for the unifier to produce an answer 43845 CPU seconds, while the strategic component uses 3290 CPU seconds. Average elapsed time for the tactical component is approximately 15 to 20 minutes, while the strategic component is about 2 minutes. Code for the tactical component, furthermore, was compiled while the strategic component code was not.

Some improvements in the design of the TEXT tactical component unifier, however, could result in significant speed-ups. Using the syntactic category provided in the input* to directly retrieve the appropriate alternative from the grammar, rather than testing each alternative individually as is currently done, would be one such improvement. The same approach could be used when unifying a lexical entry against the lexicon.

A second problem with the unifier also results from the fact that it is non-deterministic. Debugging the grammar during its design stages is made more difficult by the fact that an error doesn't show up until far away from the place where it first caused the problem. This situation is common to many non-deterministic processes and could be improved through the incorporation of testing diagnostics in the unifier.

4.6.9 Advantages -

Use of Kay's formalism in the TEXT tactical component was successful on two accounts. The first, mentioned earlier, is that the grammar allows for equal treatment of functional and syntactic categories. This means that the input to the tactical component can be formulated by a process having little information about syntax. The protagonist and goal of the predicate are easily specified by the

*The syntactic category of only the top-level FD needs to be provided in the input (cat = s). The syntactic category of each of the sentence's constituents is deduced during the process of unification.

dictionary, but the tactical component determines which of these function as surface subject and object and that their syntactic category is noun-phrase. It should be noted that the TEXT system grammar was not developed fully enough to allow for complete semantic specification by the input. For example, a modifier of a noun must be specified by adj, a syntactic classification. Further simplification of input specification was achieved by Bossie by allowing for the presence of unattached constituents.

Augmenting the grammar also turned out to be a fairly easy task in Kay's formalism. When a new type of syntactic category is needed, another alternative can be added to the grammar specifying the structure. Only the alternatives which encode the syntactic categories of which the new category is a constituent, need to be modified. All other alternatives remain unchanged. The grammar is based on a clearly modular design as a result of the use of alternatives.

Future incorporation of the use of focus information into the grammar is another favorable aspect of the grammar. Although not currently implemented in the TEXT tactical component, the grammar allows for easy incorporation of tests on focus to select an appropriate syntactic construction.

The advantages cited in this section far outweigh the disadvantages described in the previous section, which could, in fact, be taken care of through an increased implementation effort.

4.7 Practical Considerations

Although the TEXT system was implemented to test the generation principles developed in this dissertation, it addresses a real need in natural language database systems. It is just this fact that raises the question of how practical the system is for use in a real-world application. Faster response time is obviously one problem that must be solved before the TEXT system can be considered practical. Practicality, however, involves more than just questions of speed. In this section, the following two questions are discussed: 1) Does the system appropriately address the needs of the user for the questions covered? and 2) Does the system cover all the questions that a user might want to ask about database structure?

4.7.1 User Needs -

By providing a facility that can respond to questions the user has been shown to have about database structure [MALHOTRA 75], the TEXT system clearly addresses the needs of the user in ways that previous systems did not attempt. By providing a natural language response that describes concepts as they are viewed by the database system and at the same time incorporating real-world knowledge about the concepts that the user may be familiar with into the response, the system provides a comprehensible text for the naive and casual user of the database system. Some questions that need to be addressed before the system can

generated.

Currently the system may repeat information that it has just provided in a previous response. If the user is constantly presented with information he has already seen, he will become bored and frustrated with the system. Although the problem of when to omit information because of its presence in previous discourse and when to repeat it, is not a simple question (i.e. it may be affected by how much time has passed since the information was first presented), it is an issue that must be addressed before the system can be called truly practical (some issues involved in this problem are discussed in detail in Chapter 5).

One of the main issues that this dissertation addresses is the generation of multi-sentential text, a problem that was only superficially addressed by previous research in natural language generation. Now that the generation of longer text is feasible, a serious examination of when length is needed must be done. Shorter answers do not suffice in situations where the user's question does not clearly delimit the conditions for determining an appropriate answer. For example, when providing definitions, there may be more than one piece of information which can be used to identify a term, while in answering a question like "Who is the president of the programming division?", a name alone will suffice. Consistently long and wordy answers, however, may tax a user's patience.

In order to make decisions about how much information is sufficient for answering a particular user, an explicit model of the user's beliefs is necessary. In cases where a user is clearly unfamiliar with the terms involved, a more detailed explanation is necessary. If more detail is not provided, the user will be unhappy with the system performance when he can not understand the response. When it is clear that the user understands the concepts involved, longer text is unnecessary and cumbersome. An example of this situation was discussed in Chapter 2, where the definition of a hobie cat was given. If the user was ascertained to be unfamiliar with the world of sailing, a detailed explanation of catamarans was provided. If the user was knowledgeable about sailing, however, the hobie-cat could be simply identified as a brand of catamaran.

4.7.2 Question Coverage -

The TEXT system addresses three classes of questions that were discovered by Malhotra during experimentation with user interaction with natural language database systems. These three question types were questions that Malhotra found users frequently asked to familiarize themselves with the database before asking questions about the data itself. Other classes of questions which Malhotra identified in his experimentation were not covered in this system because the typical knowledge base used in natural language database systems does not encode the kind of information needed to answer such questions. These include questions about the system's capabilities, such as "Can the system

handle ellipsis?", questions about system processes such as "How is manufacturer's cost determined?", and questions involving counterfactuals, such as "If John Brown were promoted to systems analyst, what would his salary be?". Mays [MAYS 82] is currently working on a knowledge representation that would provide the information necessary to handle questions about time and processes. Discovery of other classes of questions that should be handled and the knowledge needed to answer those questions requires more experimentation with users of natural language database systems.

4.7.3 Conclusions -

The TEXT system was developed in part to increase the practicality of natural language database systems by allowing users to ask questions to familiarize themselves with the database system. The system as it stands, however, is not ready to be used in a real life situation. In order to make the system fully practical for everyday use, it must be augmented so that it can avoid repetition of information within a single session, so that it can provide shorter or longer answers depending on the user's needs, and so that it can answer other classes of questions as well as requests for definitions, requests about available information, and questions about the differences between entities. In providing the ability to answer questions of these types, the TEXT system has opened up possibilities for areas of future research which are needed to create a truly practical system.

5.0 DISCOURSE HISTORY

Tracking the discourse history involves remembering what has been generated in a single session with a user and using that information when generating additional responses. The discourse history can be used simply to avoid repetition within a single session or it can be used to provide analogies that contrast previous answers. Although the maintenance of a discourse history record was not implemented in the TEXT system, an analysis of the effects such a history could have on succeeding questions as well as the information that needs to be recorded in order to achieve those effects was made. In the following sections some examples from each class of questions that the system handles are examined to show how they would be affected by the various kinds of discourse history records that could be maintained.

5*1 What Needs To Be Recorded?

Varying quantities of information could be maintained in a discourse history record. The system could simply note that a particular question was asked and an answer provided, in which case a list of question types and their arguments would be maintained. On the other hand, the system could note both the question asked, the structure used in the response, and the particular information provided. In such cases, the question, its arguments, and the fully instantiated schema must be recorded. In the following sections, it is

shown, through example, that by simply maintaining a list of the questions asked in the session the capabilities of the system can be substantially improved by avoiding repetition. In some cases, access to the exact information that was provided earlier can be used to create a text that contrasts or parallels the earlier ones. The differences between the effects that can be achieved with each type of history record are described below. It should be noted that the discourse history could also be extended to include the actual words and syntactic structure used in the response. The affect of this type of information on the generation of responses won't be considered here as it falls under the jurisdiction of the tactical component.

5.2 Questions About The Difference Between Entities

Questions about the difference between entities are greatly affected by the previous discourse. If one of the entities in question was recently defined, then information about that entity does not have to be presented in the current answer. Consider the question "What is the difference between a destroyer and a bomb?". The answer currently provided by the system, which does not take the previous discourse into account, is shown below in (1). If the question "What is a bomb?" had been recently answered as shown in (2), then the question about the difference between entities would be more appropriately answered as shown in (3). (Note that this is a hypothetical answer, not one generated by the system.) In order to answer the question this way, the system only needs to know that it recently provided the definition of a

bomb. Since only generic class information is provided about the bomb for this question, the system can assume that information would have been provided in the definition and can thereby avoid repetition. Knowing which structure was used for the previous response does not improve the current response since its structure is not affected.

1) What is the difference between a destroyer and a bomb?

A destroyer is a surface ship with a DRAFT between 15 and 222. A ship is a vehicle. A bomb is a free falling projectile that has a surface target location. A free falling projectile is a lethal destructive device. The bomb and the destroyer, therefore, are very different kinds of entities.

(2) What is a bomb?

A bomb is a free falling projectile that has a surface target location. The bomb's target capabilities are provided by the DB attribute LETHAL_RADIUS_&_UNITS. Other DB attributes of the bomb include NAME and WEIGHT_&_UNITS. The MK-84, for example, has a LETHAL_RADIUS of 200 FT and a WEIGHT of 2000 LBS.

(3) What is the difference between a bomb and a destroyer?

A destroyer is a surface ship with a DRAFT between 15 and 222. A ship is a vehicle. Since the bomb is a destructive device, it is a very different kind of entity than the destroyer.

In responding to questions about entities that are very similar, the previous discourse information can be even more useful. Consider the question "What is the difference between an ocean-escort and a cruiser?". The answer currently produced by the TEXT system is shown below in (4). If the answer shown in (5) had been recently generated in response to the question "What is an ocean-escort?", then the answer

shown in (6) would be a more appropriate response to the question about the difference between the two entities.

(4) What is the difference between an ocean-escort and a cruiser?

The cruiser and the ocean escort are surface ships. The ship's surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. The ship has DB attributes MAXIMUM_SPEED, PROPULSION, FUEL (FUEL_CAPACITY and FUEL_TYPE), DIMENSIONS, SPEED_DEPENDENT_RANGE and OFFICIAL_NAME. Ocean escorts have a DISPLACEMENT between 3400 and 4100. All ocean escorts in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, MAST_HEIGHT of 85 and PROPULSION of STMTURGRD. Ocean escorts carry between 2 and 22 torpedoes, 16 missiles, and between 1 and 2 guns. A ship is classified as an ocean escort if the characters 1 through 2 of its HULL_NO are DE. Cruisers have a PROPULSION of STMTURGRD and a LENGTH between 510 and 673. All cruisers in the ONR database have REMARKS of 0 and FUEL_TYPE of BNKR. Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles, and between 1 and 4 guns. A ship is classified as a cruiser if the characters 1 through 2 of its HULL_NO are CL or the characters 1 through 2 of its HULL_NO are CG. The ocean escort, therefore, has a smaller LENGTH and a smaller DISPLACEMENT than the cruiser.

(5) What is an ocean-escort?

An ocean escort is a surface ship with a DISPLACEMENT between 3400 and 4100. A ship is classified as an ocean escort if the characters 1 through 2 of its HULL_NO are DE. All ocean escorts in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, MAST_HEIGHT of 85 and PROPULSION of STMTURGRD.

(6) What is the difference between an ocean-escort and a cruiser?

A cruiser is a surface ship with PROPULSION of STMTURGRD and a LENGTH between 510 and 673. A ship is classified as a cruiser if the characters 1 through 2 of its HULL_NO are CL or the characters 1 through 2 of its HULL_NO are CG. All cruisers in the ONR database have REMARKS of 0, DISPLACEMENT between 4600 and 5200, and FUEL_TYPE of BNKR. The cruiser, therefore has a larger DISPLACEMENT and a larger LENGTH than the ocean escort.

In order to provide a response like that shown in (6), the system would have to have access to a fully instantiated schema of the previous question. It could use that previous message to provide directly contrasting information about the cruiser. In other words, in places where the previous answer provided characteristic features of the ocean-escort, the new answer would provide characteristic features of the cruiser. The value of attributes whose values were provided for the ocean-escort would also be provided for the cruiser. In this case, the predicates selected for the previous answer would be the predicates which would be instantiated for the current answer. The new answer could be followed by an inference made about the two entities.

If the system knew that the previous question had been answered, but didn't have access to the instantiated schema, it could eliminate repetition about the ocean escort, but it could not necessarily select appropriate information about the cruiser when constrasting it against the ocean escort. Note that this is the case since the two entities are very similar and therefore have many common features which can be contrasted. This was not the case for the destoyer and the bomb. Thus, an improvement over the current state of the system could be achieved by recording the question only, but no guarantee that the most appropriate information for comparision would be selected could be made.

The question "What is the difference between an ocean escort and a cruiser?" will also be affected if discussion of ships occurred in the previous discourse. If a request for available information or for a definition of the ship occurred recently*, then there is no need to provide much detail on the common features of the ocean escort and the cruiser, since they will have been provided in the previous answer about ships. In such a case, the two objects in question need only be identified as ships, before discussing their differences. The modified answer is shown in (7).

(7) What is the difference between an ocean-escort and a cruiser?

The cruiser and the ocean escort are surface ships. Ocean escorts have a DISPLACEMENT between 3400 and 4100. All ocean escorts in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, MAST_HEIGHT of 85 and PROPULSION of STMTURGRD. Ocean escorts carry between 2 and 22 torpedoes, 16 missiles, and between 1 and 2 guns. A ship is classified as an ocean escort if the characters 1 through 2 of its HULL_NO are DE. Cruisers have a PROPULSION of STMTURGRD and a LENGTH between 510 and 673. All cruisers in the ONR database have REMARKS of 0 and FUEL_TYPE of BNKR. Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles, and between 1 and 4 guns. A ship is classified as a cruiser if the characters 1 through 2 of its HULL_NO are CL or the characters 1 through 2 of its HULL_NO are CG. The ocean escort, therefore, has a smaller LENGTH and a smaller DISPLACEMENT than the cruiser.

For questions such as "What is the difference between the whisky and the kitty hawk?" which involves entities which are neither very close nor very dissimilar in concept, fewer facts have to be eliminated

*The system must also be able to discriminate between those questions which occurred so long ago in the previous discourse that they would not affect the current answer and those that should.

from the answer which is currently generated by the system (shown in (8)). If a request for the definition of the whisky occurred recently in the session, the two entities in question still have to be identified as water-going vehicles and the classes of water-going vehicles discussed, but explicit facts about the whisky should be omitted from the answer. Again, if the system only had a record of the question asked, the best it could do would be to omit information about the whisky without changing anything else in the original answer, an improvement, nonetheless, over the original answer. The hypothetical answer that would be generated in such a case, is shown in (9).

The instantiated schema for the definition of the whisky could potentially be used to provide a description about the kitty hawk that parallels the earlier answer, as was done for the question about the ocean escort and the cruiser. Since the type of information that is available for one entity like these is not necessarily available for the other (e.g. ships have no information on OPERATING_DEPTH, while submarines do), the answer will not always be improved by having access to the fully instatntiated schema. In the rare case where the same type of information happens to be available, however, an improvement can be achieved.

8) What is the difference between a whisky and a kitty hawk?

The whisky and the kitty hawk are water-going vehicles. The water-going vehicle's water-going capabilities are provided by the DB attributes under WATER_GOING_OPERATION (for example, DRAFT). The water-going vehicle has DB attributes FUEL(FUEL_CAPACITY and FUEL_TYPE) and OFFICIAL_NAME. Its transporting capabilities are provided by the DB attributes under TRAVEL_MEANS(for example, FUSE_TYPE). There are 2

types of water-going vehicles in the ONR database: submarines and ships. A kitty hawk is a surface ship with a OFFICIALJJAME of KITTY HAWK, a HULLJNO of CV-63, a IRCS of BL13, a FUEL_CAPACITY of 7060, a ECONOMICJRANGE of 10000, a DRAFT of 36 and a MASTJflEIGHT of 195. A whisky is an underwater submarine with a PROPULSIONJTYPE of DIESEL and a FLAG of RDOR. A ship is classified as a kitty hawk if its CLASS is KITTY HAWK. A submarine is classified as a whisky if its CLASS is WHISKY. The whisky and the kitty hawk, therefore, are 2 different kinds of entities.

9) What is a whisky?

A whisky is an underwater submarine with a PROPULSIONJTYPE of DIESEL and a FLAG of RDOR. A submarine is classified as a whisky if its CLASS is WHISKY. All whiskies in the ONR database have REMARKS of 0, FUEL_CAPACITY of 200, FUELJTYPE of DIESEL, IRCS of 0, MAXIMUM_0PERATINGJ3EPTH of 700, NORMAL_OPERATING_DEPTH of 100 and MAXIMUM SUBMERGED_SPEED of 15.

10) What is the difference between a whisky and a kitty hawk?

The whisky and the kitty hawk are water-going vehicles. The water-going vehicle's water-going capabilities are provided by the DB attributes under WATER_GOING_OPERATION (for example, DRAFT). The water-going vehicle has DB attributes FUEL(FUEL_CAPACITY and FUELJTYPE) and OFFICIALJNAME. Its transporting capabilities are provided by the DB attributes under TRAVELJffiANS(for example, FUSEJTYPE). There are 2 types of water-going vehicles in the ONR database: submarines and ships. A kitty hawk is a surface ship. A whisky is an underwater submarine. All kitty hawks in the ONR database have a OFFICIALJNAME of KITTY HAWK, a HULLJIO of CV-63, a IRCS of BL13, a FUEL_CAPACITY of 7060, a EC0N0MIC_RANGE of 10000, a DRAFT of 36 and a MAST_HEIGHT of 195. A ship is classified as a kitty hawk if its CLASS is KITTY HAWK. The whisky and the kitty hawk, therefore, are 2 different kinds of entities.

The previous discourse could also be used to establish a context igainst which a request for the differences between objects could more ippropriately be evaluated. Context can be used to determine which set >f differences is more relevant ' to the user. For example, in

responding to the question "What is the difference between British Airways and TWA?"* it is more appropriate to include the difference in cost when the user has indicated he wants to travel from the US to London while it is more appropriate to note the difference in intermediate stops when the user has previously noted he wants to fly to Bangladesh. In such a case, the previous context indicates what the user's goals are and therefore what set of differences is relevant for the response.

5.3 Requests For Definitions

A response to a request for a definition requires less explicit modifications on the basis of the preceding discourse. Analogies can be made, however, to a similar object recently discussed in the user session. Similarity can be determined on the basis of the generalization hierarchy. If the entity currently being questioned is a sibling in the hierarchy of an entity recently discussed, then differences and similarities between the entities can be discussed. This would require use of the one-sided contrastive schema described in Chapter 2. Reference to the earlier discussed entity is made first, detailed discussion of the entity in question follows, parallelling information presented earlier, and finally, a direct comparison between the two entities made.

---

*Example is due to Peter Buneman.

Consider the question [12]"What is an aircraft-carrier?[11]. If the question "What is a destroyer?[11] had been answered recently as shown in (10), the response to the request for the definition of an aircraft carrier would be affected. The answer currently generated by the TEXT system is shown in (11). If the answer were contrasted against the definition of the destroyer, it might hypothetically look like that shown in (12).

10) What is a destroyer?

A destroyer is a surface ship with a DRAFT between 15 and 222. A ship is classified as a destroyer if the characters 1 through 2 of its HULLJNO are DD. All destroyers in the ONR database have REMARKS of 0 and FUELJTYPE of BNKR.

11) What is an aircraft carrier?

An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships. Mine warfare ships, for example, have a DISPLACEMENT of 320 and a LENGTH of 144. All aircraft carriers in the ONR database have REMARKS of 0, FUELJTYPE of BNKR, FLAG of BLBL, BEAM of 252, ENDURANCEJRANGE of 4000, ECONOMIC SPEED of 12, ENDURANCEJSPEED of 30 and PROPULSION of STMTURGRD. A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL_N0 are CV.

12) What is an aircraft carrier?

An aircraft carrier, like a destroyer, is a surface ship. It has a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULLJflO are CV. All aircraft carriers in the ONR database have REMARKS of 0, FUELJTYPE of BNKR, DRAFT between 36 and 37, FLAG of BLBL, BEAM of 252, ENDURANCE JRANGE of 4000, ECONOMIC SPEED of 12, ENDURANCEJSPEED of 30 and PROPULSION of STMTURGRD. An aircraft carrier has a larger DRAFT and a larger DISPLACEMENT than the destroyer. It has a greater LENGTH than all other ships.

5.4 Requests For Information

Responses to requests for information are affected if a question was recently asked about an ancestor of the entity in question. In such cases, all information available about the ancestor is also available for the entity, but it doesn't need to be repeated. Additional information which is particular only to the entity currently in question and restrictions on general information provided for the entity's ancestor can be discussed. Since the content of the reponse is affected and not the structure, a record that the question about the ancestor had been asked is sufficient for modifying the response. All information inherited from the ancestor, which is not further restricted for the entity, can be omitted from the response. In this case, therefore, having access to the instantiated schema for the previous response does not provide any advantage over having access to the question alone. An example of this situation is shown in (13) - (15) below. The response to the question "What do you know about aircraft?" is shown as it is currently generated by the TEXT system in (13). If the answer to the question "What do you know about vehicles?" (shown in (14)) was generated previously, the answer could be modified as shown in (15).

13) What do you know about aircraft?

The aircraft has DB attributes that provide information on FLIGHT_RADIUS, CEILING, FUEL and ROLE. Other DB attributes of the aircraft include CRUISE_SPEED, MAXIMUM_SPEED, PROPULSION and NAME. An aircraft's ROLE includes PRIMARY_ROLE, DESCRIPTION and REMARKS, its FLIGHT_RADIUS includes CRUISE_RADIUS and COMBAT_RADIUS, its CEILING includes COMBAT_CEILING and MAXIMUM_CEILING and its FUEL includes

REFUEL_CAPABILITY, FUEL_CAPACITY and FUEL_TYPE. Aircraft are categorized by PROPULSION (for example, jet) and FLAG (for example, US aircraft). Aircraft are carried by ships. An aircraft travels by flying.

14) What do you know about vehicles?

The vehicle has DB attributes that provide information on SPEED_INDICES and TRAVEL_MEANS. There are 2 types of vehicles in the ONR database: aircraft and water-going vehicles. The water-going vehicle has DB attributes that provide information on TRAVEL_MEANS and WATER_GOING_OPERATION. The aircraft has DB attributes that provide information on TRAVEL_MEANS, FLIGHT_RADIUS, CEILING and ROLE. Other DB attributes of the vehicle include FUEL (FUEL_CAPACITY and FUEL_TYPE) and FLAG.

15) What do you know about aircraft?

An aircraft's ROLE includes PRIMARY_ROLE, DESCRIPTION and REMARKS, its FLIGHT_RADIUS includes CRUISE_RADIUS and COMBAT_RADIUS, its CEILING includes COMBAT_CEILING and MAXIMUM_CEILING and its FUEL includes REFUEL_CAPABILITY, FUEL_CAPACITY and FUEL_TYPE. Other DB attributes of the aircraft include CRUISE_SPEED, MAXIMUM_SPEED, PROPULSION and NAME. Aircraft are categorized by PROPULSION (for example, jet) and FLAG (for example, US aircraft). Aircraft are carried by ships. An aircraft travels by flying.

5.5 Conclusions

Some ways in which the responses the system currently generated might be affected by the previous discourse were demonstrated in this chapter. By simply maintaining a list of the questions already asked in the session, the system could achieve a significant improvement in the responses provided in various contexts. This list would allow the system to avoid unnecessary repetition in generated responses.

In some cases, additional improvement in the responses provided could be achieved if a record were maintained of the instantiated schemas that were generated. It was shown that, although repetition could be avoided by consulting the list of questions asked, no guarantee can be made that appropriate comparative information could be provided if the instantiated schemas were not available. This is particularly important for questions about the difference between entities. Although maintaining a record of questions asked would significantly improve the system's responses in view of the previous discourse, a record of instantiated schemas would allow for more sophisticated responses in certain cases, which would parallel previous answers. These would be especially appropriate in cases where comparisons are required.

6.0 RELATED RESEARCH IN NATURAL LANGUAGE GENERATION

Interest in the generation of natural language is beginning to grow as more systems are developed which require the capability for sophisticated communication with their users* This chapter provides an overview of the development of research in natural language generation.* The earliest generation systems relied on the use of stored text and templates to communicate with the user. While these techniques are useful for situations in which a very limited range of generation is required (see, for example, Section 6.3), it is a fairly ad-hoc technique and cannot be extended in any significant way. These systems are only discussed here in relation to their use in database systems. Later research in natural language generation can be divided into the following areas of research: tactical components, planning and generation, knowledge needed for generation, and text generation. Research in each of these areas is overviewed in the following sections. The use of generation in natural language database systems is also briefly discussed.

*Other areas of research from which this work draws are overviewed in chapters relevant to that area. For example, linguistic research on discourse structure is discussed in Chapter 2.

6.1 Tactical Components (early Systems)

Some of the earliest work done in generation which used a grammar of English was by Simmons and Slocum [SIMMONS and SLOCUM 72]. Their grammar was encoded in a formalism similar to an ATN [WOODS 70] and was used to generate English sentences from semantic networks. Their system generated single sentences, whose content had already been specified in the semantic network formalism, and not texts. While their system produces different sentences for the same semantic net, they did little work on the reasons for using one of those sentences as opposed to any of the others.

Goldman [GOLDMAN 75] developed a system (MARGIE) which generates English from conceptual dependency networks. Goldman also used an ATN formalism to handle syntactic procedures, but he concentrated his research effort on developing a process to select particular words and idioms for a sentence. Goldman's generator can operate in any of three modes: question-answer, inference, or paraphrase. In paraphrase mode, Goldman's generator can output all possible ways it knows of for expressing a particular conceptual dependency net. Like Simmons and Slocum's system, however, each sentence is generated in isolation of the others and little work was done on reasons for generating one paraphrase over another.

Davey's generation system [DAVEY 79] was an early system that was capable of more sophisticated output than some of the others. His system was implemented within the context of a tic-tac-toe game and

provided a running commentary of the game. The capabilities of his system span, in some sense, both the strategic and tactical component. Although what needs to be said is given by the moves made by game participants, Davey's system maps those moves into concepts similar to the predicates used by the TEXT system (e.g. "counter-attack", "foiled threat", etc.) and uses those concepts, though limited to the tic-tac-toe context, to select connectives such as "however" and "but". His grammar is based on a functional systemic grammar derived from Hudson [HUDSON 71]. An important aspect of Davey's program is the capability for omitting details from the text which an "audience model" revealed was not necessary to provide.

6.2 Tactical Components (later Works)

More recently, McDonald ([MCDONALD 80]) has done a considerable amount of work in natural language generation. McDonald's generator (called MUMBLE), given a "message", translates that message into English. His system can also be classified as a tactical component since it does none of the initial planning of the content or organization of the message. He describes the generation process as consisting of an "expert", which knows about the domain, the "speaker", which decides what to say about the domain, and the "linguistic component", which decides what words and syntactic structures to use. McDonald's work addresses problems in the linguistic component.

McDonald's system departs significantly from earlier work because of its broad coverage of English syntax and because of the decision making process it uses to arrive at the lexical and syntactic choices made. The linguistic component consists of three major modules: the dictionary, the grammar, and the controller. The input message is expanded as an annotated tree which will eventually represent the complete surface structure of the sentence. Initially, however, it represents the structure dictated by the selected verb of the sentence and may contain untranslated message tokens. The controller makes a depth-first traversal of the tree, consulting the dictionary when message tokens are discovered which uses a discrimination net to select a structure and lexical items for the tokens. The grammar is also invoked during the traversal of the tree to make decisions about surface syntactic structure. Both the dictionary and the grammar take into account decisions already made in the tree traversal, the previous discourse, and knowledge about the audience to make their decisions.

The main differences between McDonald's system and other systems include 1) the modeling of spoken, and not written, English. This motivated the use of an indelible left-to-right decision-making procedure that increases the program's efficiency, 2) production is driven by the message to be produced and not by the grammar, again increasing efficiency, and 3) the linguistic structure of text already produced is represented and can be used as the basis of later decisions about syntactic or lexical choices.

Mathiesson [MATHIESSON 81] is working on the development of a systemic grammar [HALLIDAY 76] for generation as part of the PENMAN project at ISI [MANN and MOORE 80]. This research group is concerned with the development of a linguistically justifiable grammar since this component places an ultimate limitation on the kind of English that can be produced. They are particularly interested in the systemic grammar, which models a system of choices, as a viable alternative, and are investigating the kinds of demands such a grammar would make of a knowledge representation or text planning system.

^•^ Generation In Database Systems

Generation of natural language in database systems has been used for providing responses to questions asked and for paraphrasing users' questions as a means of verification. These generation systems have been primarily concerned with problems that arise in the tactical component since the content of the response is usually determined by the database system to which the natural language front end is interfaced and the content of the paraphrase is determined by the user's question.

Paraphrasing systems have relied on fairly simplistic generation techniques such as canned text and the use of templates. The PLANES system [WALTZ 78] selected a template in order to paraphrase a formal query and filled in the slots with arguments from the query. The Rendezvous system [CODD 78] also used templates, although it allowed for the combination of various patterns to construct a single

paraphrase. An exception to this type of generation can be found in the CO-OP paraphraser [MCKEOWN 79] which used a transformational grammar to generate paraphrases and a distinction between given and new information in the user's question to generate a paraphrase that differed syntactically from the user's.

Response generation in database systems has also been handled through the use of fairly simple techniques. Very often, information has been presented in tabular form. In some cases the user's question is inverted to produce a sentence which can be used to introduce the information retrieved from the database. An exception to this is Grishman's system [GRISHMAN 79] which used an ATN formalism to generate the response and addressed the problem of unambiguously presenting answers containing conjunction or quantifiers.

6.4 Planning And Generation

Cohen [COHEN 78] was interested in the interaction between planning and generation. He addressed the problem of planning speech acts in response to a user's question. Part of the problem involved determining which speech act (e.g. inform, request, etc.) was most appropriate for the response. The implemented system, OSCAR, was capable of selecting speech acts, specifying the agents involved, and the propositional content of the act, but it did not produce actual English output. It should be noted that Cohen did not address problems resulting when a speech act (e.g. inform) requires the presentation of a quantity of material and how this might be achieved. These problems

are, however, addressed by the TEXT system. In recent work, Cohen [COHEN 81P] proposes the use of the planning formalism for deciding upon appropriate referential descriptions.

Appelt [APPELT 81] extended Cohen's ideas by examining the interaction between planning and generation at all stages of the generation process. He showed that the planning formalism could be used not only for planning speech acts, but for determining the syntactic structure and lexical items of the text as well. Appelt was particularly interested in the use of language to satisfy multiple goals (for example, the ability to inform, request, and flatter simultaneously). He hypothesized that planning for speech is no different than the kind of planning that is done for physical actions and that, in fact, communication often requires the use of physical actions as well (e.g. pointing). He claims, therefore, that a speaker's behavior is controlled by a goal satisfaction process, whereby a speaker may construct a plan for satisfying one or more goals from available actions and these plans may involve interactions between physical and linguistic actions.

Appelt's program, KAMP, is a hierarchical planner. At the highest level, illocutionary acts, such as inform or request, are decided upon. At the next level, the surface speech act, an abstract representation of what's to be said, is determined. The next stage, concept activation, involves selection of explicit descriptions. At the lowest level, the utterance act is specified and this requires determining the actual words and syntactic structures to be used in the generated text.

Although KAMP sometimes found it necessary to generate two or three sentences at a time in order to satisfy multiple goals, it did not embody the kind of planning necessary to deal with the generation of multi-sentential text in general.

One of the major departures of Appelt's work is its refutation of the "conduit metaphor[11]. While other generation systems have assumed a separation between the processes of deciding what to say and how to say it, Appelt's work is based on the hypothesis that decisions made in the lowest level of the language generation process can influence decisions about what to say. This was implemented in the KAMP system through the use of a backtracking mechanism which can retract decisions across all levels of the planner.

It should be noted that although research for the TEXT system has focused on the problems of deciding what to say and how to organize it effectively, the approaches taken towards these problems would not be affected by the use of a control strategy that allows for backtracking across the boundaries of the strategic and tactical component. In order to allow for such a control strategy, a minor change would be needed in the TEXT system flow of control. Instead of waiting until the text message is completely constructed before invoking the tactical component, the tactical component would be invoked for each proposition as it is added to the message. The use of backtracking in the ATN mechanism, which controls the construction of the message, could then

the tactical component*

6.5 Knowledge Needed For Generation

Other approaches to research in language generation have emphasized the kind of knowledge needed in order to generate appropriate descriptions. Swartout [SWARTOUT 81] examined this problem in the context of a medical consultation system. He showed that although knowledge may be conveniently represented in one way in order to efficiently arrive at a medical diagnosis, that representation may not allow for the generation of understandable explanations about reasoning the system uses to arrive at its diagnosis. He developed a representation appropriate for explaining the expert system's reasoning which was used for the generation of explanations. His main concern, however, was with the knowledge representation and not with the generation processes.

6.6 Text Generation

Early research done in text generation includes research by Meehan [MEEHAN 77] on story generation. Meehan's system was capable of producing simple short stories about persons making plans to achieve goals and their frustrations in achieving those goals. Meehan was most concerned with the planning aspects of the program, although his system could produce multi-sentence descriptions of the characters and their actions.

Mann and Moore [MANN and MOORE 81] were more interested in the problems that arise in the generation of multi-sentential strings. They developed a system called Knowledge Delivery System (KDS) which could produce a paragraph providing instructions about what to do in case of a fire alarm. Their system relies on hill-climbing techniques to produce optimal text and does not use knowledge about discourse structure. Another drawback to their system is the fact that it operates in the very limited domain of the fire-alarm system.

More recently, researchers have begun to look at some of the strategies needed to provide explanations. Stevens and Steinberg [STEVENS and STEINBERG 81] have made an analysis of texts used by the Navy for instruction about propulsion plants. They identify 9 types of explanations that were used which include such strategies as describing the flow of information through the process, describing the process components, etc. Although not part of a generation system, this is exactly the type of information needed to extend the capabilities of the TEXT system to other generation tasks. Forbus [FORBUS 81], in fact, has proposed a system that would use qualitative simulation of processes to provide explanations of this sort.

Jensen [JENSEN 81] proposes the development of a system capable of generating standard business letters. She is particularly interested in the generation of coherent text and suggests using predicates such as CAUSE, EFFECT, etc. to aid in the selection of appropriate textual connectives. She assumes, however, that the content of the letters and the assignment of predicates to propositions has already been

determined.

7.0 SUMMARY AND CONCLUSIONS

This research has shown how principles of discourse structure, discourse coherency, and relevancy criterion can be used for the task of generating natural language text. It is important that each of these areas be taken into consideration since the generation of text and not simply the generation of single sentences was studied. In this section, the ways in which these principles were developed and incorporated into the text generation method are reviewed. Some limitations of the generation method are then discussed and finally, some possibilities for future research are presented.

7.1 Discourse Structure

A central thesis of this research is based on the observation that descriptions of the same information may vary from one telling to the next. This indicates that information need not be described in the same way in which it is stored. For the generation process, this means that production of text does not simply trace the knowledge representation. Instead, standard principles for communication are used to organize a text. These rhetorical techniques are used to guide the generation process in the TEXT system. Since different rhetorical techniques are associated with different discourse purposes, it was shown that different descriptions can be produced depending upon the discourse situation. By incorporating commonly used techniques for

communication into its answers, the system is able to more effectively convey information.

7.2 Relevancy Criterion

It was pointed out that when speaking, people are able to ignore large bodies of knowledge and focus on information that is relevant to the current discourse purpose. By constraining focus of attention to relevant information, a generation system is able to more efficiently determine what should be said next. By computing such constraints early in the generation process, the system doesn't need to consider everything it knows about when deciding what to include in the text. This partitioning of the knowledge base to a subset of relevant information was shown to be the equivalent of global focus since its contents remain focussed throughout the course of an answer. Some techniques for determining relevancy were also presented.

7.3 Discourse Coherency

It was shown that each utterance further constrains the possibilities for what can be said next. Thus, as the discourse is constructed it can be used to narrow down the set of choices for what information is selected. These constraints were implemented through the use of a mechanism which tracks immediate focus. Although immediate focus has been used effectively for the interpretation of discourse, extensions were needed if it was to be used for generation. These extensions required the specification of reasons for

discriminating between the legal focus moves a speaker can make at any given point. One major result of this research, therefore, is an ordering on these legal focus moves. Use of immediate focus constraints in the generation process means that the system will choose to say next that which most closely ties in with the previous discourse and in this way discourse coherency is ensured.

7.4 Generality Of Generation Principles

Although the generation principles developed here were used for the tasks of providing definitions, describing information, and comparing entities, these principles are applicable in all generation tasks. That is, in any situation where generation is required, the best results can be achieved if the system is capable of reasoning about the communicative strategies most appropriate for the generation task. Responding to the classes of questions handled here is not unique in requiring consideration of communicative techniques. In fact, the analysis of texts (Chapter 2) shows that the use of various techniques occurs across a wide spectrum of text types.

Moreover, any generation system which does not adopt this approach is forced into describing information in exactly the same way every time a description is required, regardless of the situation for which the description is required. In such a case, special care is required when designing the knowledge base to ensure that information is

The use of focusing, as described in this work, is clearly applicable to all types of generation. The preferences developed for shifting and maintaining focus are in no way dependent upon the type of generation required or the domain. The use of focusing provides computational constraints on the generation process that are useful in all domains. Furthermore, any multi-sentential text that is produced must be coherent and focusing is one way of ensuring that.

Although the TEXT system was designed to generate answers to questions about database structure (a feature lacking in most natural language database systems), the same techniques and principles can be used in other application areas where generation of language is needed. Computer assisted instruction systems provide a good example of where generation of language could be enhanced by taking into account the best techniques for presenting information. The generation of explanations in expert systems is another area where communicative techniques could be used to improve the quality of text output.

7.5 Limitations Of^ The Implemented System

One limitation of the TEXT system is the lack of specific information about the particular users of the system. This information could be used to tailor responses for different individuals. Currently, the system assumes a static casual and naive user and its responses are geared for that type of person. Inclusion of a user-model would allow for improvement in the quality of text produced.

Another limitation of the TEXT system is the lack of an inferencing capability. This means that the TEXT system is only capable of talking about information which is explicitly encoded in the knowledge base.* The inclusion of an inferencing capability would allow the system to generate additional information from what is known which might be appropriate in different situations.

Despite these limitations, a significant improvement in the quality of computer generated multi-sentential text was achieved by the TEXT system through the use of text structuring techniques and an account of focusing. By limiting the scope of the project, this research could focus on issues concerning the content and organization of the generated text. These two issues are complex ones that have not been appropriately handled by previous work in natural language generation. Thus, by developing a computational solution to these two questions, this work constitutes a major contribution to the field of natural language generation.

7.6 Future Directions

Although the TEXT system embodies a thorough treatment of principles of discourse structure, coherency, and relevancy, other uses and development of these principles are possible. In this section, possibilities for future research in each of these three areas is

*A few exceptions to this rule exist. For example, the system is capable of making numerical comparisons between entities.

considered.

7.6.1 Discourse Structure -

Examination of the recursive nature of the schemas and the hierarchical text structure that would result from recursion is one possibility for future work. This would require determining when a single sentence is sufficient for explanation and in what situations more detail is required. Thus, an examination of when recursion is necessary as well as an examination of how recursion is achieved are needed. Decisions about the necessity for detail rest in part on an assessment of the user's knowledge and therefore, the development of a user model for generation would be required. Although the machinery for recursion has been implemented as part of the text system, schemas for each predicate, which would be used to dictate how detail can be provided, must be developed.

Segmentation of the discourse is another possibility for future research. Segmentation involves decisions about where sentence and paragraph boundaries should occur. The delineation of paragraphs within a text corresponds in part to the amount of information presented about a given topic (a text of single sentence paragraphs, for example, would not be appropriate) and in part on semantic boundaries. Paragraphing, therefore, seems to be closely related to the recursive use of schemas. Where a predicate has been expanded as a schema, a sufficient amount of related information is presented to warrant the use of a paragraph. The compare and contrast schema, which

entails the use of other schemas, illustrates how this use of paragraphing would work. In example 3.1 below, a paragraph is formed in each place where a different schema has been invoked.

Identification Schema

The cruiser and the ocean escort are surface ships. The ship's surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. The ship has DB attributes MAXIMUMJSPEED, PROPULSION, FUEL (FUEL_CAPACITY and FUELJTYPE), DIMENSIONS, SPEEDJDEPENDENTJtANGE and OFFICIAL_NAME.

Attributive Schema

Ocean escorts have a DISPLACEMENT between 3400 and 4100. All ocean escorts in the ONR database have REMARKS of 0, FUELJTYPE of BNKR, FLAG of BLBL, MASTJHEIGHT of 85 and PROPULSION of STMTURGRD. Ocean escorts carry between 2 and 22 torpedoes, 16 missiles, and between 1 and 2 guns. A ship is classified as an ocean escort if the characters 1 through 2 of its HULLJJO are DE.

Attributive Schema

Cruisers have a PROPULSION of STMTURGRD and a LENGTH between 510 and 673. All cruisers in the ONR database have REMARKS of 0 and FUELJTYPE of BNKR, Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles, and between 1 and 4 guns. A ship is classified as a cruiser if the characters 1 through 2 of its HULL_N0 are CL or the characters 1 through 2 of its HULLJJO are CG,

Return to Compare and Contrast Schema

The ocean escort, therefore, has a smaller LENGTH and a smaller DISPLACEMENT than the cruiser.

Other research on discourse structure could involve the development of different strategies or structures for use in describing other kinds of knowledge. Descriptions of processes, cause and effects, and temporal or spatial relations might require the use of

different communicative techniques as well as different combinations of these techniques. This type of research is particularly important in the development of interaction with dynamic databases. While the questions handled by the TEXT system are appropriate for static databases, additional question-answering capabilities are necessary for dynamic natural language database systems that encode knowledge about changes that can occur. Answering these types of questions requires the ability to describe processes, cause and effects, and temporal sequences.

7.6.2 Relevancy -

In the TEXT system as currently implemented, global focus remains unchanged throughout the course of an answer. For longer sequences of text, global focus may shift. This is also related to the use of schema recursion (see Chapter 3). Where more detail is required, focus shifts to the details presented. The implementation of shifting focus would require capabilities for expansion and stacking of focus.

More sophisticated techniques for determining relevancy could also be examined. Any complex analysis of relevancy must take into account the particular user and thus, a user model would be required. Other techniques for determining relevancy might rely on a theoretical account of saliency (e.g. [MCDONALD 82]).

7.6.3 Coherency -

An open question for the use of focusing concerns the nature of the structures for maintaining and shifting focus. More complex structures may be needed for some situations. For example, a speaker may introduce an item into conversation, but specify that he will continue to talk about it at a later point. It is unclear whether the use of a simple stack is sufficient for modeling these types of explicitly orchestrated expectations (see [REICHMAN 81] for a discussion of speaker expectations). Another case where different structures may be necessary is illustrated by Joshi and Weinstein [JOSHI WEINSTEIN 81] who point out that certain syntactic structures are used to turn off past foci as candidates for future foci. They also show that shifts in focus to items that are functionally dependent on past foci may have a similar effect.

As mentioned in Chapter 3, no account of the use of foci implicitly associated with focus was made since this requires the use of general world knowledge and an inferencing capability. How and when focusing on items associated with previous foci fits in with the scheme developed here remains a topic for future research.

7.6.4 User Model -

The addition of a user model to the generation system is another direction for future research. It was shown that a user model is needed for extensions in both the use of discourse structure and

relevancy for generation. A user model is necessary for the use of full schema recursion, both for determining when more detail is necessary and for expanding the relevant knowledge pool to include that detail. A user model could also be used in determining which information to include in the relevant knowledge pool initially. In order to incorporate information about the users into the generation system, research on exactly what information about the users can be determined from the discourse and on how that information can be deduced needs to be done.

7.7 Conclusion

The TEXT system successfully incorporates generation principles into a method for generating coherent, effective English text of paragraph length. This thesis has illustrated that knowledge about discourse structure can be used to guide the generation process and that focus constraints can be used to ensure discourse coherency. It was shown, furthermore, that an interaction between these structural and semantic processes is required for the production of text. By addressing issues such as these in the generation of multi-sentential text, this work has opened up a number of possible avenues for future research.

8.0 APPENDIX A -- INTRODUCTION TO WORKING

The paragraphs from the first topic group in the Introduction to Working [TERKEL 72] are reproduced below.

INTRODUCTION

This book, being about work, is, by its very nature, about violence — to the spirit as well as to the body. It is about ulcers as well as accidents, about shouting matches as well as fistfights, about nervous breakdowns as well as kicking the dog around. It is, above all (or beneath all), about daily humiliations. To survive the day is triumph enough for the walking wounded among the great many of us.

The scars, psychic as well as physical, brought home to the supper table and the TV set, may have touched, malignantly, the soul of our society. More or less. ("More or less," that most ambiguous of phrases, pervades many of the conversations that comprise this book, reflecting, perhaps, an ambiguity of attitude toward The Job. Something more than Orwellian acceptance, something less than Luddite sabotage. Often the two impulses are fused in the same person.)

It is about a search, too, for daily meaning as well as daily bread, for recognition as well as cash, for astonishment rather than torpor; in short, for a sort of life rather than a Monday through Friday sort of dying. Perhaps immortality, too, is part of the quest. To be remembered was the wish, spoken and unspoken, of the heroes and heroines of this book.

There are, of course, the happy few who find a savor in their daily job: the Indiana stonemason, who looks upon his work and sees that it is good; the Chicago piano tuner, who seeks and finds the sound that delights; the bookbinder who saves a piece of history; the Brooklyn fireman, who saves a piece of life ... But don't these satisfactions, like Jude's hunger for knowledge tell us more about the person than about his task? Perhaps. Nonetheless, there is a common attribute here: a meaning to their work well over and beyond the reward of the paycheck.

For the many, there is a hardly concealed discontent. The blue-collar blues is no more bitterly sung than the white-collar moan. "I'm a machine," says the spot-welder. "I'm caged," says the bank teller, and echoes the hotel clerk. "I'm a mule," says the steelworker. "A monkey can do what I do," says the receptionist. "I'm

nan a farm implement," says the migrant worker. "I'm an object," he high-fashion model. Blue collar and white call upon the cal phrase: "I'm a robot." "There is nothing to talk about," the accountant despairingly enunciates. It was some time ago that Henry sang "A man ain't nothin' but a man." The hard, unromantic s: he died with his hammer in his hand, while the machine pumped onetheless, he found immortality. He is remembered.

9.0 APPENDIX B — SAMPLE OUTPUT OF THE TEXT SYSTEM

```
$ lisp
Franz Lisp, Opus 34
->  64400 bytes read into 2c00 to 66fff
loading init.lsp
* changed def


l.(load-TEXT)
kb access files dskined
initial files of system dskined
TEXT strategic component loaded
Dictionary loaded
[fasl [kathy.tact]newu.o]
[fasl [kathy.tact]ucon.o]
Unifier functions loaded
Grammar loaded
[fasl [kathy•tact]linear.o]
Linearizer loaded
Unifier ready to use
TEXT tactical component loaded


fql loaded_____system ready to use
nil
2.(TEXT)
                        Welcome to TEXT
              A system which answers questions about
                        database structure

-------------------------------------------------------

TEXT allows you to familiarize yourself with a database
before asking questions of the database itself.


You may ask questions of the following type about the database
structure:


     1. What do you know about <e>?
     2. What is a <e>?
     3. What is the difference between <el> and <e2>?


by using the corresponding functions:
```

1. (information <e>)
2. (definition <e>)
3. (differense <e1> <e2>)

where <e> represents an entity-class in the database.

As each answer is constructed, you have the option of seeing
various intermediary information printed.
Would you like to see tracing information? (y/n) n
Would you like to see the potential focus list? (y/n) n
Would you like to see the focus-stack? (y/n) n

To exit from TEXT, type EXIT.
To execute regular LISP functions type x <lisp function>.

You may begin asking questions.

(information)

Schema selected: constituency

proposition selected:
(attributive db OBJECT (name REMARKS))

focus: OBJECT

proposition selected:
(constituency OBJECT (VEHICLE DESTRUCTIVE-DEVICE))

focus: OBJECT

proposition selected:
(attributive db VEHICLE (based-dbs (SOME-TYPE-OF TRAVEL_MEANS)
(SOME-TYPE-OF SPEED_INDICES)))

focus: VEHICLE

proposition selected:
(attributive db DESTRUCTIVE-DEVICE (based-dbs (SOME-TYPE-OF
LETHAL_INDICES)))

focus: DESTRUCTIVE-DEVICE

Message through dictionary. Entering tactical component

Processor time used: 977 seconds
Time used for garbage collection: 500 seconds

All entities have DB attributes REMARKS. There are 2 types of entities in the ONR database: destructive devices and vehicles. The vehicle has DB attributes that provide information on SPEED_INDICES and TRAVEL_MEANS. The destructive device has DB attributes that provide information on LETHAL_INDICESs.

Processor time used: 17314 seconds
Time used for garbage collection: 8819 seconds

#(information ECHO-II-SUBMARINE)

Schema selected: attributive

proposition selected:
(attributive def ECHO-II-SUBMARINE ((FLAG) (PROPULSION_TYPE)) (((FLAG (RDRD))) ((PROPULSION_TYPE (NUCL)))))

focus: ECHO-II-SUBMARINE

proposition selected:
(amplification db ECHO-II-SUBMARINE (((FLAG) (PROPULSION_TYPE)) ((FLAG (RDRD))) ((PROPULSION_TYPE (NUCL)))))) (MAXIMUM_SUBMERGED_SPEED 20) (NORMAL_OPERATING_DEPTH 100) (MAXIMUM_OPERATING_DEPTH 700) (IRCS 0) (FUEL_TYPE NUCL) (REMARKS 0))

focus: (((FLAG) (PROPULSION_TYPE)) (((FLAG (RDRD))) ((PROPULSION_TYPE (NUCL))))

proposition selected:
(classification ECHO-II-SUBMARINE 0 sub-classes)

focus: ECHO-II-SUBMARINE

proposition selected:
(analogy rels ECHO-II-SUBMARINE ON (GUN 0) (MISSILE (16 99)) (TORPEDO (16)))

ocus: ECHO-II-SUBMARINE

roposition selected:
explanation based-db ECHO-II-SUBMARINE (((FLAG) (PROPULSION_TYPE)) ((FLAG (RDRD))) ((PROPULSION_TYPE (NUCL))))) (CLASS ECHO II))

ocus: ECHO-II-SUBMARINE

essage through dictionary. Entering tactical component

Echo IIs have a PROPULSION_TYPE of NUCL and a FLAG of RDRD. All cho IIs in the ONR database have REMARKS of 0, FUEL_TYPE of NUCL, IRCS f 0, MAXIMUM_OPERATING_DEPTH of 700 and NORMAL_OPERATING_DEPTH of 100. here are no sub-classes of echo II in the ONR database. Echo IIs arry 16 torpedoes, between 16 and 99 missiles and 0 guns. A submarine s classified as an echo II if its CLASS is ECHO II.

(definition GUIDED)

chema selected: constituency

roposition selected:
identification GUIDED PROJECTILE (restrictive TRAVEL-MEANS ELF-PROPELLED) (non-restrictive ROLE PROJECTED-OBJECT))

ocus: GUIDED

roposition selected:
constituency GUIDED (MISSILE TORPEDOE))

ocus: GUIDED

roposition selected:
attributive def MISSILE GUIDED TARGET-LOCATION SURFACE-AIR)

ocus: MISSILE

roposition selected:
attributive def TORPEDOE GUIDED TARGET-LOCATION UNDERWATER)

ocus: TORPEDOE

proposition selected:
(evidence based-db MISSILE (TARGET-LOCATION SURFACE-AIR) (INDICATED-BY DESCRIPTION) (HAVE ALTITUDE))

focus: MISSILE

proposition selected:
(evidence based-db TORPEDOE (TARGET-LOCATION UNDERWATER) (SOME-TYPE-OF DEPTH))

focus: TORPEDOE

proposition selected:
(attributive db GUIDED (name NAME) (Topics HORZJtANGE_&_UNITS TIMEJTOJCARGET_&JJNITS))

focus: db-attribute

Message through dictionary. Entering tactical component

Processor time used: 758 seconds
Time used for garbage collection: 0 seconds

A guided projectile is a projectile that is self-propelled. There are 2 types of guided projectiles in the ONR databse: torpodoes and missiles. The missile has a target location in the air or on the earth's surface. The torpedo has an underwater target location. The missile's target location is indicated by the DB attribute DESCRIPTION and the missile's flight capabilities are provided by the DB attribute ALTITUDE. The torpedo's underwater capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUMJ3PERATINGJ)EPTH). The guided projectile has DB attributes TIME_TO_TARGET__& JJNITS, HORZ_RANGE_&_UNITS and NAME.

Processor time used: 26229 seconds
Time used for garbage collection: 11531 seconds

//(definition SHIP)

Schema selected: identification

proposition selected:
(identification SHIP WATER-VEHICLE (restrictive TRAVEL-MODE SURFACE)

(non-restrictive TRAVEL-MEDIUM WATER) (non-restrictive FUNCTION TRANSPORTATION))

focus: SHIP

proposition selected:
(evidence based-db SHIP (TRAVEL-MODE SURFACE) (HAVE DRAFT) (HAVE DISPLACEMENT))

focus: (TRAVEL-MODE SURFACE)

proposition selected:
(attributive db SHIP (name OFFICIAL JNAME) (topics SPEED_DEPENDENT RANGE DIMENSIONS) (duplicates (FUEL FUELJFYPE FUEL_CAPACITY)) Tattrs PROPULSION MAXIMUM_SPEED))

focus: db-attribute

proposition selected:
(particular-illustration SHIP ((name OFFICIALJNAME) (topics SPEEDJDEPENDENT_RANGE DIMENSIONS) (duplicates (FUEL FUELJTYPE FUEL_CAPACITY)) (attrs PROPULSION MAXIMUM_SPEED)) (OFFICIALJNAME DOWNES) (ENDURANCEJRANGE 2200) (ECONOMICJtANGE 4200) (LENGTH 438) (BEAM 46) (DRAFT 25) (FUELJTYPE BNKR) (FUEL_CAPACITY 810) (PROPULSION STMTURGRD) (MAXIMUM_SPEED 29))

focus: ((name OFFICIALJNAME) (topics SPEEDJ)EPENDENT_RANGE DIMENSIONS) (duplicates (FUEL FUEL_TYPE FUEL_CAPACITY)) (attrs PROPULSION MAXIMUM_SPEED))

Message through dictionary. Entering tactical component

Processor time used: 3184 seconds
Time used for garbage collection: 858 seconds

A ship is a water-going vehicle that travels on the surface. Its surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. Other DB attributes of the ship include MAXIMUMJSPEED, PROPULSION, FUEL (FUEL_CAPACITY and FUELJTYPE), DIMENSIONS, SPEED DEPENDENT RANGE and OFFICIAL NAME. The DOWNES, for example, has MAXIHUM_SPEED~~of 29, PROPULSION 3"f STMTURGRD, FUEL of 810 (FUEL_CAPACITY) and BNKR (FUELJTYPE), DIMENSIONS of 25 (DRAFT), 46 (BEAM), and 438 (LENGTH) and SPEED_DEPENDENT_RANGE of 4200 (ECONOMIC_RANGE) and 2200 (ENDURANCE_RANGE).

```
Processor time used:  31150 seconds
Time used for garbage collection:  11050 seconds


#(definition AIRCRAFT-CARRIER)


Schema selected:  identification


proposition selected:
(identification  AIRCRAFT-CARRIER  SHIP  (restrictive  ((LENGTH)
(DISPLACEMENT))  (((  LENGTH  (1039  1063)))  ((DISPLACEMENT  (78000
80800))))) (non-restrictive TRAVEL-MODE SURFACE))

focus:  AIRCRAFT-CARRIER


proposition selected:
(analogy ranges AIRCRAFT-CARRIER (((LENGTH (1039 1063))) ((DISPLACEMENT
(78000 80800)))) (larger-than-all LENGTH (1039 1063)) (larger-than-most
DISPLACEMENT (78000 80800)))

focus:  (((LENGTH (1039 1063))) ((DISPLACEMENT (78000 80800))))


proposition selected:
(particular-illustration MINE-WARFARE-SHIP (((LENGTH  (1039  1063)))
((DISPLACEMENT (78000 80800)))) (LENGTH (144)) (DISPLACEMENT (320)))

focus:  (((LENGTH (1039 1063))) ((DISPLACEMENT (78000 80800))))


proposition selected:
(amplification  db  AIRCRAFT-CARRIER  (((LENGTH  (1039  1063)))
((DISPLACEMENT  (78000  80800))))  (PROPULSION  STMTURGRD  )
(ENDURANCE_SPEED 30) (ECO NOMIC_SPEED 12) (ENDURANCE_RANGE 4000) (BEAM
252) (FLAG_BLBL

     ) (FUEL_TYPE BNKR ) (REMARKS 0 ))

focus:  (((LENGTH (1039 1063))) ((DISPLACEMENT (78000 80800))))


proposition selected:
(evidence  based-db  AIRCRAFT-CARRIER  (((LENGTH)  (DISPLACEMENT))
(((LENGTH  (1039 1063))) ((DISPLACEMENT (78000 80800))))) (HULL_NO (1 2
CV)))

focus:  (((LENGTH)  (DISPLACEMENT))  (((LENGTH  (1039  1063)))
((DISPLACEMENT (78000 80800)))))
```

Message through dictionary. Entering tactical component

Processor time used: 3290 seconds
Time used for garbage collection: 0 seconds

An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships. Mine warfare ships, for example, have a DISPLACEMENT of 320 and a LENGTH of 144. All aircraft carriers in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, BEAM of 252, ENDURANCE_RANGE of 4000, ECONOMIC_SPEED of 12 and ENDURANCE_SPEED of 30. A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL_NO are CV.

Processor time used: 43845 seconds
Time used for garbage collection: 18256 seconds

#(differense OCEAN-ESCORT CRUISER)

Schema selected: c&c-attributive

proposition selected:
(identification (OCEAN-ESCORT CRUISER) SHIP (non-restrictive TRAVEL-MODE SURFACE))

focus: (OCEAN-ESCORT CRUISER)

proposition selected:
(evidence based-db SHIP (TRAVEL-MODE SURFACE) (HAVE DRAFT) (HAVE DISPLACEMENT))

focus: (TRAVEL-MODE SURFACE)

proposition selected:
(attributive db SHIP (name OFFICIAL_NAME) (topics SPEED_DEPENDENT_RANGE DIMENSIONS) (duplicates (FUEL FUEL_TYPE FUEL_CAPACITY)) (attrs PROPULSION MAXIMUM_SPEED))

focus: SHIP

```
(attributive def OCEAN-ESCORT SHIP ((DISPLACEMENT)) (((DISPLACEMENT
(3400 4100)) )))

focus: OCEAN-ESCORT


proposition selected:
(amplification db OCEAN-ESCORT (((DISPLACEMENT)) (((DISPLACEMENT (3400
4100))))) (PROPULSION STMTURGRD ) (MAST_HEIGHT 85) (FLAG BLBL )
(FUEL_TYPE BNKR ) (REMARKS 0 ))

focus: (((DISPLACEMENT)) (((DISPLACEMENT (3400 4100)))))


proposition selected:
(analogy rels OCEAN-ESCORT ON (GUN (1 2)) (MISSILE (16)) (TORPEDOE (2
22)))

focus: OCEAN-ESCORT


proposition selected:
(explanation based-db OCEAN-ESCORT (((DISPLACEMENT)) (((DISPLACEMENT
(3400 4100) )))) (HULL_NO (1 2 DE)))

focus: OCEAN-ESCORT


proposition selected:
{(attributive def CRUISER SHIP ((LENGTH PROPULSION)) (((LENGTH (510
673)) (PROPUL SION (STMTURGRD )))))

focus: CRUISER


proposition selected:
(amplification db CRUISER (((LENGTH PROPULSION)) (((LENGTH (510 673))
(PROPULSION (STMTURGRD ))))) (FUEL_TYPE BNKR ) (REMARKS 0 ))

focus: (((LENGTH PROPULSION)) (((LENGTH (510 673)) (PROPULSION
(STMTURGRD

     )))))


proposition selected:
(analogy rels CRUISER ON (GUN (1 4)) (MISSILE (4 98)) (TORPEDOE (8
42)))

focus: CRUISER
```

proposition selected:
(explanation based-db CRUISER (((LENGTH PROPULSION)) (((LENGTH (510 673)) (PROPU LSION (STMTURGRD))))) (HULL NO (1 2 CA) (1 2 CG) (1 2 CL)))

focus: CRUISER

proposition selected:
(inference OCEAN-ESCORT CRUISER (HULLJfO (1 2 DE) (1 2 CA)) (smaller DISPLACEMENT) (smaller LENGTH) (PROPULSION (STMTURGRD) (STMTURGRD)))

focus: (OCEAN-ESCORT CRUISER)

Message through dictionary. Entering tactical component

Processor time used: 5949 seconds
Time used for garbage collection: 0 seconds

The cruiser and the ocean escort are surface ships. The ship's surface- going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. The ship has DB attributes MAXIMUMJSPEED, PROPULSION, FUEL(FUEL_CAPACITY and FUELJTYPE), DIMENSIONS, SPEEDJ}EPENDENT_RANGE and OFFICIALJNAME. Ocean escorts have a DISPLACEMENT between 3400 and 4100. All ocean escorts in the ONR database have REMARKS of 0, FUELJTYPE of BNKR, FLAG of BLBL and MASTJ1EIGHT of 85. Ocean escorts carry between 2 and 22 torpedoes, 16 missiles and between 1 and 2 guns. A ship is classified as an ocean escort if the characters 1 through 2 of its HULL_N0 are DE, Cruisers have a PROPULSION of STMTURGRD and a LENGTH between 510 and 673. All cruisers in the ONR database have REMARKSs of 0. Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles and between 1 and 4 guns. A ship is classified as a cruiser if the characters 1 through 2 of its HULLJJO are CL or the characters 1 through 2 of its HULLJIO are CG, The ocean escort, therefore, has a smaller LENGTH and a smaller DISPLACEMENT than the cruiser.

Processor time used: 80671 seconds

Time used for garbage collection: 32890 seconds

//(definition WHISKY-SUBMARINE)

Schema selected: identification

proposition selected:
(identification WHISKY-SUBMARINE SUBMARINE (restrictive ((FLAG) (PROPULSIONJTYPE)) (((FLAG (RDOR))) ((PROPULSIONJTYPE (DIESEL))))) (non-restrictive TRAVEL-MODE UNDERWATER))

focus: WHISKY-SUBMARINE

proposition selected:
(evidence based-db WHISKY-SUBMARINE (((FLAG) (PROPULSIONJTYPE)) (((FLAG (RDOR))) ((PROPULSIONJTYPE (DIESEL))))) (CLASS WHISKY))

focus: (((FLAG) (PROPULSIONJTYPE)) (((FLAG (RDOR))) ((PROPULSION_TYPE (DIESEL)))))

proposition selected:
(attributive db WHISKY-SUBMARINE (MAXIMUMJSUBMERGEDJ3PEED 15) (NORMAL_OPERATING__ DEPTH 100) (MAXIMUMJDPERATINGJDEPTH 700) (IRCS 0) (FUELJTYPE DIESEL) (Fu*EL_CAPACITY 200) (REMARKS 0))

focus: db-attribute

Message through dictionary• Entering tactical component

Processor time used: 1631 seconds
Time used for garbage collection: 0 seconds

A whisky is an underwater submarine with a PROPULSIONJTYPE of DIESEL and a FLAG of RDOR. A submarine is classified as a whisky if its CLASS is WHISKY. All whiskies in the ONR database have REMARKS of 0, FUELJCAPACITY of 200, FUELJTYPE of DIESEL, IRCS of 0, MAXIMUMJ3PERATINGJDEPTH of 700, NORMALJ3PERATINGJDEPTH of 100 and MAXIMUM_SUBMERGED_SPEED of 15.

Processor time used: 23357 seconds
Time used for garbage collection: 8108 seconds

//(differense MISSILE TORPEDOE)

Schema selected: c&c-attributive

proposition selected:
(identification (MISSILE TORPEDOE) GUIDED (non-restrictive TRAVEL-MEANS

SELF-PROPELLED))

focus: (MISSILE TORPEDOE)

proposition selected:
(evidence based-db GUIDED (TRAVEL-MEANS SELF-PROPELLED) (HAVE FUSE_TYPE (EXCEPTION MISSILE)) (SOME-TYPE-OF SPEED_INDICES (EXCEPTION TORPEDOE)))

focus; (TRAVEL-MEANS SELF-PROPELLED)

proposition selected:
(attributive db GUIDED (name NAME) (topics HORZ_RANGE_&_UNITS TIME_TO_TARGET_&_UNITS))

focus: GUIDED

proposition selected:
(attributive def MISSILE GUIDED TARGET-LOCATION SURFACE-AIR)

focus: MISSILE

proposition selected:
(amplification evidence based-db MISSILE (TARGET-LOCATION SURfACE-AIR) (INDICATED-BY DESCRIPTION) (HAVE ALTITUDE))

focus: (TARGET-LOCATION SURFACE-AIR)

proposition selected:
(attributive db MISSILE (name) (topics TIME_TO_TARGET_&_UNITS LETHAL_RADIUS_&_UNITS ALTITUDE) (attrs SPEED PROBABILITY_OF_KILL))

focus: db

proposition selected:
(analogy rels MISSILE ON AIR-VEHICLE WATER-VEHICLE)

focus: MISSILE

proposition selected:
(attributive def TORPEDOE GUIDED TARGET-LOCATION UNDERWATER)

focus: TORPEDOE

(amplification evidence based-db TORPEDOE (TARGET-LOCATION UNDERWATER) (SOME-TYPE-OF DEPTH))

focus: (TAREGET-LOCATION UNDERWATER)

proposition selected:
(attributive db TORPEDOE (name)(topics TIME_TO_TARGET_&_UNITS HORZ_RANGE_&_UNITS ACCURACY_&_UNITS) (attrs MAXIMUM_DEPTH FUSE_TYPE))

focus: db-attribute

proposition selected:
(analogy rels TORPEDOE ON WATER-VEHICLE)

focus: TORPEDOE

proposition selected:
(inference MISSILE TORPEDOE (same TRAVEL-MEANS) (different TARGET-LOCATION) ((MISSILE (INDICATED-BY DESCRIPTION) (HAVE ALTITUDE)) (TORPEDOE (SOME-TYPE-OF DEPTH))))

focus: (MISSILE TORPEDOE)

Message through dictionary. Entering tactical component

Processor time used: 4163 seconds
Time used for garbage collection: 441 seconds

The torpedo and the missile are self-propelled guided projectiles. The guided projectile's propulsion capabilities are provided by the DB attributes under SPEED_INDICES (for example, MAXIMUM_SPEED) and FUSE_TYPE. The guided projectile has DB attributes TIME_TO_TARGET_&_UNITS, HORZ_&_UNITS and NAME. The missile has a target location in the air or on the earth's surface. The missile's target location is indicated by the DB attribute DESCRIPTION and its flight capabilities are provided by the DB attribute ALTITUDE. Other DB attributes of the missile include PROBABILITY_OF_KILL, SPEED, ALTITUDE, LETHAL_RADIUS_&_UNITS and TIME_TO_TARGET_&_UNITS. Missiles are carried by water-going vehicles and aircraft. The torpedo has an underwater target location. Its underwater capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUM_OPERATING_DEPTH). Other DB attributes of the torpedo include FUSE_TYPE, MAXIMUM_DEPTH, ACCURACY_&_UNITS, HORZ_RANGE_&_UNITS, and

database by the torpedo's attribute DEPTH and the missile's attributes ALTITUDE and DESCRIPTION.

Processor time used: 44786 seconds
Time used for garbage collection: 15608 seconds

(differense DESTROYER BOMB)

Schema selected: c&c-identification

proposition selected:
(identification DESTROYER SHIP (restrictive ((DRAFT)) (((DRAFT (15 222))))) (non-restrictive TRAVEL-MODE SURFACE))

focus: DESTROYER

proposition selected:
(identification SHIP VEHICLE (non-restrictive FUNCTION TRANSPORTATION))

focus: SHIP

proposition selected:
(identification BOMB FREE-FALLING (restrictive TARGET-LOCATION SURFACE) (non-restrictive TRAVEL-MEANS GRAVITY-PULL))

focus: BOMB

proposition selected:
(identification FREE-FALLING DESTRUCTIVE-DEVICE (non-restrictive FUNCTION LETHAL-KILL))

focus: FREE-FALLING

proposition selected:
(inference DESTROYER BOMB very-different-entities)

focus: (DESTROYER BOMB)

Message through dictionary. Entering tactical component

Processor time used: 1916 seconds
Time used for garbage collection: 497 seconds

A destroyer is a surface ship with a DRAFT between 15 and 222, ship is a vehicle^ A bomb is a free falling projectile that has surface target location. A free falling projectile is a lett" destructive device* The bomb and the destroyer, therefore, are ve different kinds of entities•

Processor time used: 18672 seconds
Time used for garbage collection: 8901 seconds

1.0 BIBLIOGRAPHY

[AKMAJIAN 73]. Akmajian, A., "The role of focus in the interpretation of anaphoric expressions." in Anderson and Kiparsky (eds.), Festschriff for Morris Halle, Holt, Rinehart and Winston, New York, N.Y., 1973.

[APPELT 81]. Appelt, D. E., Planning Natural Language Utterances to Satisfy Multiple Goals, Ph.D. dissertation, Stanford University, Stanford, Ca., 1981.

[BARTH 79]. Barth, J. The Floating Opera. 1979.

[BOSSIE 82]. Bossie, S., forthcoming M. S. thesis, University of Pennsylvania, Philadelphia, Pa., 1982.

[BRACHMAN 79]. Brachman, R., "On the epistemological status of semantic networks." in N. Findler (ed.) Associative Networks: Representation and Use of Knowledge by Computer, Academic Press, N. Y., 1979.

[CHAFE 76]. Chafe, W. L., "Givenness, contrastiveness, definiteness, subjects, topics, and points of view." in C. N. Li (ed.), Subject and Topic, Academic Press, N. Y., 1977.

[CHAFE 77]. Chafe, W. L., "Creativity and verbalization and its implications for the nature of stored knowledge." in R. O. Freedle (ed.), Discourse Production and Comprehension, Vol. 1, Ablex Publishing Co., N. J., 1977, pp. 41-55.

[CHAFE 79]. Chafe, W. L., "The flow of thought and the flow of language." in T. Givon (ed.) Syntax and Semantics, Vol. 12 of Discourse and Syntax, Academic Press, 1979.

[CHEN 76]. Chen, P. P. S., "The entity-relationship model - towards a unified view of data." ACM Transactions on Database Systems, Vol. 1, No. 1. (1976).

[CHOMSKY 71]. Chomsky, N. "Deep structure, surface structure and semantic interpretation." in Steinberg and Jakobovitz (eds.) Semantics: An Interdisciplinary Reader in Philosophy, Linguistics and Psychology, Cambridge University Press, London., 1971.

[CODD 78]. Codd, E. F., et. al., Rendezvous Version 1: An Experimental English-Language Query Formulation System for Casual Users of Relational Databases, IBM Research Laboratory, San Jose, Ca., Technical Report RJ2144(29407), 1978.

[COHEN 78]. Cohen, P., On Knowing What to Say: Planning Speech Acts, Technical Report No. 118, University of Toronto, Toronto, 1978.

[COHEN 81P]. Cohen, P., "The need for identification as a planned action." Proceedings of the 7th Annual Joint Conference on Artificial Intelligence, 1981.

[COHEN 81R]. Cohen, R., "Investigation of processing strategies for the structural analysis of arguments." in Proceedings of the 19th Annual Meeting of the Association for Computation Linguistics, Stanford, Ca., June 1981, pp. 71-6.

[COLLINS 74]. Collins, A., Warnock, and Passafiume, J., Analysis and synthesis of tutorial dialogues. BBN Report #2789, March 1974.

[DAVEY 79]. Davey, A., Discourse Production, Edinburgh University Press, Edinburgh, 1979.

[ENCYCLOPEDIA 76]. Encyclopedia Americana, Americana Corporation, New York, N. Y., 1976.

[FIRBAS 66]. Firbas, J., "On defining the theme in functional sentence analysis." Travaux Linguistiques de Prague 1, Univ. of Alabama Press, 1966.

[FIRBAS 74]. Firbas, J.," Some aspects of the Czechoslovak approach to problems of functional sentence perspective," Papers on Functional Sentence Perspective, Academia, Prague, 1974.

[FORBUS 81]. Forbus, K. and A. Stevens, Using Qualitative Simulation to Generate Explanations, Bolt Beranek and Newman, Inc., Technical Report 4490, March 1981. Also appeared in Cognitive Science3, 1981.

[GOLDMAN 75]. Goldman,N. M., "Conceptual generation," in R. C. Schank (ed.), Conceptual Information Processing, North-Holland, Amsterdam, 1975.

[GRICE 75]. Grice, H. P., "Logic and conversation," in P. Cole and J. L. Morgan (eds.) Syntax and Semantics: Speech Acts, Vol. 3, Academic Press, N.Y., 1975.

[GRIMES 75]. Grimes, J. E. The Thread of Discourse. Mouton, The Hague, Paris. (1975).

[GRISHMAN 79]. Grishman, R., Response generation in question-answering systems. Proceedings of the 17th Annual Meeting of the ACL, La Jolla, Ca., August 1979, pp. 99-102.

[HALL 73]. Hall, R.L., "Toxicants occurring naturally in spices and flavors." in Toxicants Occurring Naturally in Foods, National Academy of Sciences, Washington D. C., 1973.

[HALLIDAY 67]. Halliday, M. A. K., "Notes on transitivity and theme in English." Journal of Linguistics 3, 1967.

[HALLIDAY 76]. Halliday, M. A. K., System and Function in Language, Oxford University Press, London, 1976.

[HENDRIX 79]. Hendrix, G., "Encoding knowledge in partitioned networks." in N. V. Findler (ed.), Associative Netowrks: Representation and Use of Knowledge by Computer, Academic Press, N. Y., 1979.

[HOBBS 78]. Hobbs, J., Coherence and Corefernce. SRI Technical Note 168, SRI International, Menlo Park, Ca., 1978.

[HUDSON 71]. Hudson, R. A., North-Holland Linguistic Series. Volume 4: English Complex Sentences, North-Holland, London and Amsterdam, 1971.

[JENSEN 81]. Jensen, K., R. Ambrosio, R. Granville, M .Kluger, and A. Zwarico, "Computer Generation of Topic Paragraphs: Structure and Style," in Proceedings of the ACL, New York, N.Y., December 1981.

[JOSHI and WEINSTEIN 81]. Joshi, A. and S. Weinstein, "Control of inference: role of some aspects of discourse structure - centering", in Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, Canada, August 1982.

[LEE and GERRITSEN 78]. Lee, R. M. and R. Gerritsen, "Extended semantics for generalization hierarchies", in Proceedings of the 1978 ACM-SIGMOD International Conference on Management of Data, Austin, Tex., 1978.

[KAPLAN 79]. Kaplan, S. J., Cooperative responses from a portable natural language database query system. Ph. D. dissertation, Univ. of Pennsylvania,Philadelphia, Pa., 1979.

[KAY 79]. Kay, M. "Functional grammar." in Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society. (1979).

[KEENAN 71]. Keenan, E. L., "Two kinds of presupposition in natural language." in Fillmore and Langendoen (eds.), Studies in Linguistic Semantics, Holt, Rinehart, and Winston, New York, N. Y., 1971.

[LEE and GERRITSEN 78]. Lee, R. M. and R. Gerritsen, "Extended semantics for generalization hierarchies", in Proceedings of the 1978 ACM-SIGMOD International Conference on Management of Data, Austin, Tex., 1978.

[LUTTWAK 79]. Luttwak, E. N., "The American Style of Warfare and the Military Balance," Survival, Vol. XXI, #2, March-April 1979, pp. 57-60.

[LYONS 68]. Lyons, J. Introduction t£ Theoretical Linguistics, Cambridge University Press, London, 1968.

[MALHOTRA 75]. Malhotra, A. Design criteria for a knowledge-based English language system for management: an experimental analysis. MAC TR-146, MIT, Cambridge, Mass. (1975).

[MANN and MOORE 80]. Mann, W. C, and J. A. Moore, Computer as Author — Results and Prospects, USC/Information Sciences Institute, RR-79-82, 1980.

[MANN and MOORE 81]. Mann, W. C, and J. A. Moore, "Computer generation of multiparagraph English text," American Journal of Computational Linguistics Vol. 7, No. 1, January-March 1981.

[MARTIN 73]. Martin, L., "Tactical nuclear weapons." in Arms and Strategy: the World Power Structure Today, David McKay Company Inc., 1973.

[MATHIESSON 81]. Mathiesson, C.M.I.M., "A grammar and a lexicon for a text-production system," in Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, Stanford, Ca., 1981, pp. 49-56.

[MAYS 80]. Mays, E., "Correcting Misconceptions about data base structure." Proceedings 3rd CSCSI Biennial Meeting, Victoria, B.C., May 1980.

[MAYS 82]. Mays, E., Disseration Proposal, University of Pennsylvania, Philadelphia, Pa. 1981.

[MCCOY 82]. McCoy, K. F., Automatic enhancement of a data base knowledge representation used for natural language generation, forthcoming M. S. thesis, University of Pennsylvania, Philadelphia, Pa., 1982.

[MCDONALD 80]. McDonald, D. D., Natural language production as a process of decision making under constraint. Ph.D dissertation, draft version, MIT, Cambridge, Mass. (1980).

[MCDONALD 82]. McDonald, D. D., personal communication. 1982

[MCKEOWN 79]. McKeown, K. R., "Paraphrasing using given and new information in a question-answer system," Proceedings of the 17th Annual Meeting o£ the Association for Computational Linguistics, August 1979, pp. 67-72.

[MCKEOWN 80A]. McKeown, K. R., Generating descriptions and explanations: applications to questions about database structure. Dissertation proposal, Technical Report #MS-CIS-80-9, Univ. of Pennsylvania, Philadelphia, Pa. (1980).

[MCKEOWN 80B]. McKeown, K. R., "Generating relevant explanations: natural language responses to questions about database structure.[11] in Proceedings of AAAI, Stanford Univ., Stanford, Ca. (1980). pp. 306-9.

[MEEHAN 77]. Meehan, J.R., ffTALE-SPIN, an interactive program that writes stories," in Proceedings o£ the 5th International Joint Conference on Artificial Intelligence, August 1977, pp. 91-8.

[MOORE 81]. Moore, R. C, "Problems in logical form." in Proceedings of the 19th Annual Meeting o£ the Association for Computation Linguistics, June 1981, pp. 117-24.

[NELSON 77]. Nelson, J., The Poorperson's Guide to Great Cheap Wines, McGraw-Hill Book Co., New York, N. Y., 1977.

[PALMER 81]. Palmer, M., "A case for rule-driven semantic processing." Proceedings <o£ the 19th Annual Meeting o£ the Association for Computational Linguistics, June 1981, pp. 125-32.

[PATERSON 80]. Paterson, J., The Hamlyn Pocket Dictionary of^ Wines, Hamlyn, New York, N. Y., 1980.

[PRINCE 79]. Prince, E., "On the given/new distinction." CLS 15, 1979.

[QUICK 73]. Quick, J., Dcitionary o£ Weapons and Military Terms, McGraw-Hill, New York, N. Y., 1973.

[QUIRK and GREENBAUM 73]. Quirk, R. and S. Greenbaum, A Concise Grammar ££ Contempory English, Harcourt, Brace and Jovanovich Inc., New York, N.Y7, 1973.

[REICHMAN 81]. Reichman, R., Plain Speaking: A Theory and Grammar of Spontaneous Discourse. Ph.D. Thesis, Harvard University, Cambridge, Ma., 1981.

[REINHART 81J. Reinhart, T., "Pragmatics and linguistics: an analysis of sentence topics." in Philosophica, special issue on Pragmatic Theory, 1981.

[ROSEN 76]. Rosen, S., Future Facts, A Touchstone Book, Simon and Schuster, New York, N. Y., 1976.

[SCHUBERT et. al. 79]. Schuber, L. K., Goebel, R., G., and Cercone, N. J., "The structure and organization of a semantic network for comprehension and inference." in N. V. Findler (ed.), Associative

Networks: Representation and Use of Knowledge by Computer, Academic Press, N. Y., 1979.

[SGALL et. al. 73]. Sgall, P., E. Hajicova, and E. Benesova, Topic, Focus and Generative Semantics, Scriptor Verlag, Democratic Republic of Germany, 1973.

[SHEPHERD 26]. Shepherd, H. R., The Fine Art of Writing, The Macmillan Co., New York, N.Y., 1926.

[SHIPLEY 80]. Transcripts of mother-child dialogues. Unpublished manuscript, Univ. of Pennsylvania, Philadelphia, Pa., 1980.

[SIDNER 79]. Sidner, C. L., Towards a computational theory of definite anaphora comprehension in English discourse. Ph.D. dissertation, MIT, Cambridge, Mass. (1979).

[SIMMONS and SLOCUM 72]. Simmons, R., and J. Slocum, "Generating English discourse from semantic networks," Communications of the ACM Vol. 15, No. 10, October 1972, pp. 891-905.

[SMITH and SMITH 77]. Smith, J. M. and Smith, D. C. P., "Database abstractions: aggregation and generalizaiton." University of Utah, ACM Transactions on Database Systems, Vol. 2, #2, June 1977, pp. 105-33.

[STEVENS and STEINBERG 81]. Stevens, A., and C. Steinber, A Typology of Explanations and its Application to Intelligent Computer Aided Instruction, Bolt Beranek and Newman, Inc., Technical Report 4626, March 1981.

[SWARTOUT 81]. Swartout, W. R., Producing Explanations and Justifications of Expert Consulting Programs, Massachusetts Institute of Technology, Technical Report MIT/LCS/TR-251, January 1981.

[TENNANT 79]. Tennant, H., Experience with the evaluation of natural language question answerers. Working paper #18, Univ. of Illinois, Urbana-Champaign, Ill. (1979).

[TERKEL 72]. Terkel, S., Working, Avon Books, New York, N. Y., 1972.

[THOMPSON 77]. Thompson, H., "Strategy and tactics: a model for language production." Papers from the 13th Regional Meeting, Chicago Linguistic Society. (1977).

[WEISCHEDEL 75]. Weischedel, R. M., Computation of a Unique Class of Inferences: Presupposition and Entailment, Ph.D. dissertation, University of Pennsylvania, Philadelphia, Pa., 1975.

[WALTZ 78]. Waltz, D. L., "An English language question answering system for a large relational database," Communications of the ACM, Vol. 21, No. 7, July 1978.

WINSTON 79]. Winston, P. H., "Learning by Understanding Analogies," lemo 520, MIT Artificial Intelligence Laboratory, 1979.

WOODS 70]. Woods, W. A., "Transition network grammars for natural anguage analysis." Communications of the ACM, Vol. 13, No. 10, ctober, 1970, pp. 591-606.

10.0 INDEX