A Knowledge-Based Approach to Language Production

By

Paul Schafran Jacobs

A.B. (Harvard University) 1981
S.M. (Harvard University) 1981

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

GRADUATE DIVISION

OF THE

UNIVERSITY OF CALIFORNIA, BERKELEY

Approved: [illegible] 9/8/85
Chairman Date
[illegible] 8/9/85
Charles J. Fillmore 8 Aug. 1985

to lisa

Acknowledgements

The "acknowledgements" section of a dissertation may be the most pleasurable to write. It is certainly the most free-form, demanding the least formality and technicality. But definitely not the easiest.

First, I must thank the generous sources who funded a starving graduate student, paying for stipends, computer cycles, and conference trips. This research was sponsored in part by the Office of Naval Research under contract N00014-80-C-0732, the National Science Foundation under grants IST-8007045 and IST-8208602, and the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3041, Monitored by the Naval Electronic Systems Command under contract N00039-82-C-0235.

The hard-won signature and much of the credit for this work belong to Robert Wilensky, who served as my adviser and verbal sparring partner throughout the preparation of this thesis. Much to my astonishment, he really *read* the numerous drafts of numerous papers and chapters that I dropped into his in-box for almost four years. His attention and insight have made an indescribable contribution to this work.

The other members of my committee, Lotfi Zadeh and Charles Fillmore, also provided instrumental comments and advice. George Lakoff, who avoided reading the thesis by fleeing to Bali to study linguistics, made his contribution to many of the linguistic ideas in this work by providing education, inspiration and innovation. Also indispensable in the preparation of this thesis were contributions from discussions with the members of the Berkeley Artificial Intelligence Research group, including (in order of height) Lisa Rau, Nigel Ward, David Chin, Margaret Butler, Michael Braverman, Yigal Arens, Joe Faletti, Jim Martin, Jim Mayfield, Rick Alterman, Charley Cox, Marc Luria, and Peter Norvig.

A number of other individuals contributed to the thesis by providing feedback on various aspects, including Stuart Shieber, Doug Appelt, Johanna Moore, and Norm Sondheimer. Norm deserves a special thanks for helping to work out many of the kinks in the earlier versions.

Now for the hard part: I can't possibly acknowledge all the people who didn't contribute to this thesis in the least, but instead contributed to making life a lot more fun and interesting. These include the poker crowd, the East Bay dinner group, the hump-night crew, the folks at San Ramon, the Palo Alto people, and the friends and family back East.

The hardest part: When I started on this work it might have been the most important thing to me. As I finish, it lags behind the bonds that I have formed with people such as those mentioned above. Fortunately, the most important one to me, Lisa Rau, who's also the best one at reminding me what's really important, will be with me for long after I've forgotten what I've written here. It's hard to separate one's personal life from one's thesis, and Lisa has been a partner, mentor, and supporter in both.

Table of Contents

A Knowledge-Based Approach to Language Production

Paul Schafran Jacobs

ABSTRACT

The development of natural language interfaces to Artificial Intelligence systems is dependent on the representation of knowledge. A major impediment to building such systems has been the difficulty in adding sufficient linguistic and conceptual knowledge to extend and adapt their capabilities. This difficulty has been apparent in systems which perform the task of language production, *i. e.* the generation of natural language output to satisfy the communicative requirements of a system.

The problem of extending and adapting linguistic capabilities is rooted in the problem of integrating abstract and specialized knowledge and applying this knowledge to the language processing task. Three aspects of a knowledge representation system are highlighted by this problem: *hierarchy*, or the ability to represent relationships between abstract and specific knowledge structures; *explicit referential knowledge*, or knowledge about relationships among concepts used in referring to concepts; and *uniformity*, the use of a common framework for linguistic and conceptual knowledge. The *knowledge-based approach* to language production addresses the language generation task from within the broader context of the representation and application of conceptual and linguistic knowledge.

This knowledge-based approach has led to the design and implementation of a knowledge representation framework, called *Ace*, geared towards facilitating the interaction of linguistic and conceptual knowledge in language processing. Ace is a *uniform, hierarchical* representation system, which facilitates the use of abstractions in the encoding of specialized knowledge and the representation of the referential and metaphorical relationships among concepts.

A general-purpose natural language generator, KING (Knowledge INtensive Generator), has been implemented to apply knowledge in the Ace form. The generator is designed for *knowledge-intensivity* and *incrementality*, to exploit the power of the Ace knowledge in generation. The generator works by applying structured associations, or mappings, from conceptual to linguistic structures, and combining these structures into grammatical utterances. This has proven to be a simple but powerful mechanism, easy to adapt and extend, and has provided strong support for the role of conceptual organization in language generation.

1. Introduction

1.1. The Language Production Task

The use of natural language as an interface to computer systems requires a capacity for language *analysis,* or understanding, and language *production,* or generation. This thesis is concerned with the problem of natural language generation, which may be characterized as follows:

> *The task of language production is to generate natural language utterances to satisfy the communicative requirements of a system.*

One good reason to focus on this problem is that relatively little research has directly addressed the task of language generation, and many generation systems built thus far have been somewhat *ad hoc*. A plausible explanation for the paucity of generation research is historical. There has been little practical demand for sophisticated generation capabilities, as the programs using these capabilities have been able to express their output using "canned" or very constrained text. The need for powerful mechanisms for linguistic expression comes about only as the programs themselves become more powerful and have more to say.

A second reason to concentrate on language production is that it provides a useful means of evaluating the results of a system. The language produced is inextricably dependent, for example, on the knowledge used to produce it; thus a good way of evaluating the knowledge representation is by examining the natural language output. Even the skilled knowledge engineer is a better judge of natural language than of coded knowledge; language generation is therefore a facility for testing the power of a knowledge base.

Historically, the task of language generation has developed from the problem of constraining linguistic output (Yngve, 1962) to the problem of translating conceptual representation into natural language (Goldman, 1974), and more recently to the broader problem of "planning" or "deciding" what to say as well as how to say it (McDonald, 1980; Appelt, 1981; McKeown, 1982). Ongoing research in the field has centered more on the "what to say" or "strategic" aspects of generation and less on the "how to say it" or "tactical" components.

None of the problems described above has been adequately solved. This thesis is devoted to an approach to the diverse problems of generation through a core issue: the organization and utilization of knowledge about language. The work presented here includes a framework for linguistic knowledge representation, called *Ace*, and an implemented language generator called *KING* (Knowledge INtensive Generator), which makes use of this framework.

Building a substantial language generation system "from scratch" can be a formidable task, yet there are relatively few techniques and tools available to use as a foundation. While linguists and computer scientists have succeeded in implementing grammars which approximate natural languages, there are still a great many linguistic phenomena which fall outside of the coverage of these grammars. Tools have also been developed which facilitate the tactical component of generation, but the fundamental problems of building linguistic structures, selecting words, and tailoring linguistic output to a context all present difficulties for existing systems. Furthermore, progress in these technical areas is hampered by the fact that most systems seem inherently limited in their design; that is, they cannot be easily modified to attain a broader or more powerful capacity to produce language. One way of attacking the technical problems of generation, therefore, is to address the problem of building a system whose capacity may be increased. This problem is described in the next section.

1.2. The Extensibility and Adaptability Problem

Increasing the capacity of a natural language system may be achieved by two slightly different means: *extensibility*, which facilitates the addition of new knowledge to the system, and *adaptability*, which allows the existing knowledge of the system to be utilized in a new manner. The example which follows illustrates how these two attributes come into play.

The UNIX Consultant Example

The work described in this thesis has stemmed from the project of building adaptable natural language interfaces. The primary area of application of this work is as a component of the UNIX[†] Consultant (Wilensky, Arens, and Chin, 1984), an on-line natural language help facility equipped with knowledge about the UNIX operating system. The UNIX Consultant (UC) is designed to answer questions, especially from naive users, about the computer system which they are using. While fairly constrained in terms of the technical knowledge of the system, the UC application is extremely rich in terms of the linguistic and world knowledge used in communicating within the domain. This makes it a good proving ground for a general-purpose language generator. Consider the following simple exchange, handled by the current version of the UNIX Consultant:

User: I tried typing 'rm foo', but I got 'foo not removed'.
UC: **You need write permission on the parent directory.**

In addition to demonstrating some basic phenomena which provide serious tests for a language understanding mechanism, this exchange illustrates certain difficult aspects of the language generation task. The generator expresses its message using a combination of technical and non-technical language. In the technical knowledge of the system, this response refers to a relationship between the user and a particular bit which can be set in a data structure called an *inode*. In the linguistic output, this bit is referred to as "write permission on the parent directory" and the verb used to suggest changing the bit is "need". Much of the knowledge required to produce such a response may be particular to the UNIX domain or to protection in the UNIX domain. But certainly some of the knowledge about the use of "need" and "on", for example, is not. If the generator is to be called upon to produce a variety of similar responses, it should be able to take advantage of

[†] UNIX is a trademark of Bell Laboratories

this more abstract knowledge. The problem of extending and adapting the generator, then, becomes intertwined with the problem of exploiting generalizations or abstractions. This problem is further detailed below.

Extending and Adapting a Generation System

Selecting terms such as "write permission" and "parent directory" and building a linguistic structure with the verb "need" entail equipping the generator with knowledge pairing the linguistic structures with some representation of their meaning. Such knowledge links "write permission" and "parent directory" to the system's technical knowledge about protection and directories, and links its knowledge about "needing write permission" to its knowledge about preconditions for using the 'rm' command. The generation process can then use these pairings to select linguistic components from its technical representation, and can combine these components using basic grammatical knowledge. This is a rough but essentially accurate description of the way most generation systems go about producing this type of sentence.

The process described above is weak with respect to both extensibility and adaptability. For example, as the system is extended, the generator is called upon to produce output such as the following:

(1) 'Chmod' can be used to *give* you write permission.
(2) You don't *have* write permission on the directory.
(3) You can't *get* write permission on the directory.
(4) You *need* ethernet access.
(5) You don't *have* ethernet access.

The verbs used above are consistently those which refer to possession and changes of possession. To treat the knowledge of each use of each verb as independent of all other knowledge seems wasteful. Each new piece of knowledge about language will be as difficult to add as the previous bit of knowledge, although it seems that new knowledge should be able to take advantage of existing knowledge. But each usage above still appears to have a specialized interpretation in the UNIX world. Sentences 1-3 above, for example, all involve the status of a UNIX inode. Sentences 4 and 5 may refer to the physical configuration of machines. It would be an overgeneralization to suggest that sentences 2 and 5, for example, invoke exactly the same interpretation of the verb "have". But it also seems inappropriate to treat the two interpretations as completely independent. A prerequisite for extensibility, then, is to facilitate the addition of new knowledge to the system by exploiting consistencies in the way language is used without overgeneralizing.

The question of adaptability becomes important when the generator is to be used for another application. Suppose that UC's generator is to be connected to a system which explains proprietary technical material. It is then called upon to generate, "You need security clearance", or "You don't have access to that information". If all its knowledge about "need" and "have" is specific to the UNIX domain, then it would be necessary to give it new knowledge about the components of these sentences and their related meanings. The system should have knowledge about "need" and "have" which applies to this new domain as well. Just as a novice UNIX user fits new specific linguistic knowledge to what is already known, it is desirable for an adaptable language interface to utilize general linguistic knowledge across domains.

It seems that computationally practical and cognitively realistic answers to both the adaptability and extensibility questions above hinge on the representation of knowledge in a system. It is necessary also to have a flexible mechanism for applying the knowledge so that this adaptation or extension is not limited by the way in which the program is written. The practical questions presented here are thus based in the theoretical issues of the organization and application of knowledge. The next two sections describe the foundations of a framework for language generation which promotes extensibility and adaptability.

1.3. Representing Knowledge about Language

The focus on extensibility and adaptability suggests certain features of a theoretical framework for language generation. Principally, it suggests that the representation of knowledge about language must be evaluated with respect to ease of application, ease of acquiring new knowledge, and ease of adaptation. These considerations highlight the following aspects of a knowledge representation framework:

Hierarchy

The use of language is the product of the application of knowledge at varying levels of specificity. Many examples, such as the use of the terms "write permission" and "parent directory" in the dialogue here, illustrate the use of specialized constructions and suggest that such constructions are the product of the application of specialized associations between linguistic terms and conceptual structures. Treating knowledge about these constructs entirely as specialized knowledge, however, ignores the relationships among such terms as "write permission", "read permission", and "dialup access". A hierarchical representation takes advantage of common knowledge about such terms, thereby achieving a more parsimonious representation of knowledge and also facilitating the encoding of additional related linguistic knowledge.

Explicit Referential and Metaphorical Knowledge

The knowledge used in the production and analysis of language diverges from the classical framework in which knowledge is considered as basically factual, and pieces of knowledge are assertions about the world. We may view the task of language generation as the instantiation of new knowledge structures which are related to existing structures in some referential capacity. Specifically, conceptual structures may be only indirectly associated with linguistic structures; in these cases it is difficult to treat the association as a factual relationship. In these instances, it is useful to have a knowledge representation which explicitly encodes relationships among knowledge structures which may be used in analyzing or producing language. For example, in the dialogue above, "You *need* write permission" is used to describe a UNIX relationship between the user and an inode structure. This relationship is referred to using verbs of possession; thus "You must *have* write permission" and "You must be *given* write permission" may refer to the same relationship. Factually it is difficult to assert anything about whether one can actually *have* write permission. However, there may be links in a system which specifically relate knowledge structures to other structures which are used in referring to them. These links may represent metaphorical relationships among concepts as well as relationships between linguistic and conceptual structures.

Uniformity

It is often convenient to represent the knowledge of a system in a variety of specialized forms. But knowledge of these various forms is still *knowledge*, and may be related to other knowledge of different types and encoded into a uniform hierarchical framework in spite of its distinguishing characteristics. A uniform framework allows for the same principles to be applied to the representation and manipulation of a variety of knowledge forms.

The discussion which follows further describes the role of each of these elements in facilitating extensibility and adaptability.

The Importance of Hierarchies

Hierarchical representations have been consistently exploited in Artificial Intelligence systems, but underexploited in the representation of linguistic knowledge. This has been due partially to the difficulty of defining abstract linguistic entities in the upper levels of a hierarchy and partially due to the continuing reliance on linguistic notations which do not lend themselves to hierarchical encoding. The ability to utilize abstract knowledge in the acquisition, retrieval, and application of specialized knowledge seems fundamental to the way human beings use language. In practice, however, natural language systems fail to exploit abstract knowledge, particularly linguistic knowledge. A system which makes use of a hierarchical organization of both conceptual and linguistic knowledge ultimately facilitates a more parsimonious knowledge representation as well as the addition of new knowledge.

Chapters 4, 5, and 6 of this thesis show how linguistic and conceptual knowledge can be organized into a hierarchy and how linguistic abstractions are encoded in this hierarchy.

The Importance of Explicit Referential and Metaphorical Knowledge

The framework proposed here suggests explicitly encoding knowledge about referential relationships between language and meaning. This includes knowledge about relationships between words and concepts, between linguistic structures and conceptual structures, and relationships among conceptual structures which are used in reference. Such relationships between language and meaning may be used for both language analysis and production, and these relationships may be triggered by the instantiation of structures in the knowledge base. We may treat "An A *refers* to a B" as a declarative piece of knowledge just as "An A *is* a B" is. It will be shown that this explicit representation of referential relationships facilitates the interaction of abstract and generalized knowledge and thereby promotes extensibility.

In practice, virtually all generation systems have utilized knowledge about language primarily either by matching templates attached to linguistic structures or by applying queries attached to these structures. Such implementations have also embodied a "direct translation" relationship between meaning and language, often applying knowledge which pairs linguistic structure with denotation. A representation in which relationships between language and meaning are explicitly encoded permits the application of knowledge without special matching or queries.

One aspect of the way natural language is often used is that a concept may be expressed by referring to other concepts which are related to it. Such phenomena are often classified as metaphor or metonymy. Like referential relationships between language and meaning, these conceptual relationships are not so much factual knowledge as they are knowledge about the way language is used. Chapter 4 describes the way in which these relationships, called *views*, can be encoded. The use of views in generation achieves extensibility by permitting a piece of linguistic knowledge to be used in referring to a variety of related concepts.

The Importance of Uniformity

Arguments both for and against uniformity of representation have been frequently voiced in the Artificial Intelligence community. On the one hand, uniformity may be an asset in that it may help to explain the apparent versatility of human knowledge. On the other hand, different types of tasks seem to require different organizations and distinct types of knowledge structures. In theory, there are strong cases on both sides, depending on how uniformity is defined. In practice, it is difficult to enforce strict uniformity, as the understanding of the cognitive processes is not sufficiently complete to allow for a completely uniform mechanism to perform them. Uniformity in this text is used to suggest having a common framework for the representation of knowledge irrespective of the domain of the knowledge. This type of uniformity can be of help in constructing a natural language system, as it allows for the development of general mechanisms which aid in the application of knowledge in many forms.

The use of a uniform representation system for linguistic and conceptual knowledge is described in Chapters 4-6.

The theoretical knowledge representation framework to be presented in Chapter 3 embodies the three main characteristics described here. This framework allows the encoding of both linguistic and conceptual knowledge in hierarchies, using a general knowledge structure called a *structured association* to represent relationships among entities in the hierarchies. Special structured associations are used to represent metaphorical or *view* relationships among concepts, as well as explicit referential relationships between linguistic and conceptual structures. Because these structured associations are used specifically to join conceptual structures to other knowledge structures which may be used in referring, knowledge within this system explicitly encodes the intricate relationships between language and meaning which seem essential to the generation task. The same basic structured associations may be used for both the linguistic and conceptual hierarchies, thus providing for uniformity of representation. The description of the means by which knowledge is encoded and used within this framework constitutes the body of this thesis.

The aspects of knowledge representation presented here are important for extending and adapting linguistic capacity. The next section discusses a framework for applying this knowledge to the generation task.

1.4. Characteristics of the Generation Process

As a rule, it is far more difficult to modify a program than it is to modify the knowledge which the program manipulates. This suggests that a processing framework for generation should exploit the power of a knowledge representation using as simple a mechanism as possible. The following are the two principal characteristics of this mechanism:

Knowledge-intensivity

Language generation is knowledge-intensive because knowledge about language, about the speaker's intentions, about the hearer, and about context all influence the language produced. The process of generation is thus one in which knowledge at different levels and of different types may interact, which may be characterized as a knowledge-intensive process. The result of the generation process is heavily dependent on the knowledge being applied. This has the practical effect of making it possible to change substantially the results of the generation program by adapting its knowledge base, without changing the program itself.

Incrementality

In generation, lexical and structural knowledge is incrementally refined until an utterance is produced. For example, the choice of a verb and its object may be refined into a complete surface structure, rather than choosing the structure and filling out its constituents. In a system in which knowledge may be drawn from a variety of sources, incrementality is the result of combining knowledge structures derived from these sources. The synthesis of a well-formed utterance is produced by incrementally combining linguistic structures.

This section presents a general description of the implications of knowledge-intensivity and incrementality for a natural language generation system. The details of this processing framework are considered in Chapter 7, and the implementation of a system which realizes this framework is discussed in Chapter 8.

The Importance of Knowledge-Intensivity

The distinction between knowledge-intensive mechanisms and process-intensive mechanisms can be a subtle one, as procedures are themselves a form of knowledge. The problem with procedural knowledge representations for generation is that it is often difficult to adapt these procedures, thus often a new procedure is required to perform a task similar to that performed by another. The practical goal of knowledge-intensivity here is to minimize actual program size, with the effect of having the workings of the program be especially sensitive to the nature of its knowledge. This is intended to facilitate the use of the knowledge by different programs, for example in a knowledge base shared by analyzer and generator, and to allow for the extension and adaptation of a program by adding to its knowledge base.

The Importance of Incrementality

Incrementality is an important processing characteristic which is difficult to achieve. Time efficiency and simplicity often favor the use of knowledge units rich in information rather than the *synthesis* of such units. In generation, the common approach has been to select substantial chunks of linguistic structure based on sets of linguistic and/or conceptual attributes. For example, a generator might select a subject-verb-object linguistic structure and build a sentence by instantiating this structure. Suppose that knowledge about the relationship between subject and verb agreement is attached to this sentence structure. The relationship between verb and object, including the case of the object and the underlying conceptual role of the object, may be specified also as particular to this sentence, thus the invocation of the subject-verb-object structure will carry with it the knowledge necessary to select and use the complete structure. This tends to highlight the top-down aspects of the generation process: Once a linguistic structure is chosen, most the work of the generator becomes oriented towards filling out the components of the structure.

The alternative proposed here is to minimize the effect of the selection process and emphasize the use of more simple structural relationships to synthesize a completed structure. The results of applying these structural relationships may then be incrementally combined to produce an utterance. Consider the example of the subject-verb-object sentence: The incremental method is to utilize linguistic relationships independently of the linguistic pattern: Subject-verb agreement, case, and the conceptual role of an object are not dependent on surface structure. An incremental mechanism can derive these relationships from the concept to be expressed and the constraints to be satisfied, attaching minimal knowledge to the individual linguistic patterns. Complete linguistic structures can be built from the derived relations. Incrementality in this case is thus an effect of applying more abstract knowledge about linguistic structure. The result is more complexity in the interaction of structural relationships, but less redundancy and greater versatility of knowledge.

The processing framework for generation which will be discussed in Chapter 7 proposes a simple mechanism to allow for both knowledge-intensivity and incrementality. The bulk of the work performed by this mechanism results from the process of *mapping*, or utilizing referential and metaphorical relationships to produce new linguistic structures. These linguistic structures are then incrementally combined to form utterances. Simple rules for applying structured associations and for selecting the grammatical structures used for synthesizing utterances make the nature of the generation process in this model largely dependent on the nature of the knowledge used.

The theoretical framework sketched thus far has been implemented in a knowledge representation system and a real-time generator, which will be described in the next section.

1.5. The System

The output from the current version of the UNIX Consultant system described earlier in this chapter is produced by a generator called KING (Knowledge INtensive Generator). The theory of linguistic knowledge representation outlined here is the basis for an implementation of a knowledge representation framework called *Ace* (Jacobs and Rau, 1984), which KING utilizes in producing utterances.

The development of Ace and KING stemmed largely from results of the PHRED (PHRasal English Diction) system (Jacobs, 1983, 1985) used in an earlier version of UC. Written as a general-purpose generator to share a knowledge base with the PHRAN (PHRasal ANalyzer) language analyzer (Wilensky and Arens, 1980), PHRED was easily adapted to produce UC output in English and Spanish. Like other similar programs, however, PHRED was subject to the limitations of the knowledge representation from which it operated. While it was not difficult to extend and adapt PHRED's knowledge to new problems and domains, it was not possible to do so and still take advantage of much of the knowledge that the system already had. Ace and KING alleviate this problem by facilitating the representation and utilization of abstractions in language generation.

Ace incorporates an implementation of a hierarchical knowledge representation language called KODIAK (Wilensky, 1984), and includes features designed to facilitate the association of language and meaning. The particulars of this representation will be discussed in Chapter 4. The following sentences produced by KING illustrate the basic capabilities of the generator:

John sold Mary the book.
Mary gave John a hug.
John was given a kiss on the cheek by Mary.
John's being kissed on the cheek bothered Mary.
With which enemy did John bury the hatchet?
Who from Mary's home town does John think she will marry?

Applied to the domain of the UNIX Consultant, KING produces responses such as the following:

You need write permission on the parent directory.
'Chmod' can be used to get write permission on a directory.
You can give others in your group write permission on a file by
    typing 'chmod g+w <filename>'.
To deny messages, type 'mesg n'.
To take back a job that has been sent to the line printer, use 'lprm'.

The Ace representation framework allows KING to make use of general grammatical knowledge, knowledge about verbs such as "need", "give", and "take", and knowledge about linguistic structures used with these verbs. This knowledge can be applied to the generation of specialized constructs such as "giving a hug" and to technical references such as "giving write permission". Extending KING is thus a problem of relating new knowledge to what the generator already knows, rather than of specifying completely the information necessary to communicate about a particular topic or domain.

1.6. A KING Example

This section presents a modified trace of the KING generator producing the sentence: "John was given a kiss on the cheek by Mary." A more detailed explanation of this example is presented in the discussion of the implementation of the generator in Chapter 8. The left column below contains the diagnostic output of KING, showing the processes applied and new knowledge structures produced during each phase of the generation process. The right column gives a brief explanation of each step in the production of the complete sentence:

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated* ----------------------------------	Explanation -----------------

Input to KING:
(kissing
  (kisser mary1)
  (kissee john1)
  (surface cheek1))
	Input concept expressed in Lisp form. The manner in which this type of conceptual knowledge is related to other knowledge in the Ace hierarchy is discussed in Chapters 4-6.

Applying related views	

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
New concept produced: (kiss-transfer (source mary1) (recipient john1) (given-kiss (kissing (surface cheek1))))	The *kissing* concept is related to the *kiss-transfer* concept by a *VIEW*, as will be described in Chapter 4. The *kiss-transfer* concept represents a metaphorical view of *kissing* as a transfer. The choice and application of this VIEW is part of the *mapping* phase of generation described in Chapter 7. This phase allows new concepts to be used in generating input concepts.
Applying related views	
New concept produced: (kiss-giving (kiss-giver mary1) (recipient john1) (given-kiss (kissing (surface cheek1))))	The *kiss-giving* concept is related to the *kiss-transfer* concept by a VIEW. This VIEW is the same as that which relates *transfer-event*s to *giving* actions in general. *Transfer-event*s are concepts usually referred to using the verbs "give" or "take", depending on whether the event is viewed as *giving* or *taking*. The distinction between events and actions, as well as the justification for this type of VIEW, are presented in Chapters 4 and 6.
Applying related views	
New concept produced: (pred (c-subj john1) (c-pred (recipient (of (kiss-giving ...)))))	The application of another VIEW, triggered by the knowledge that sentences are formed using *predications*, results in the instantiation of this structure. The role of predications is further discussed in Chapter 6.
No related views found. *Applying referential relationships* *Selecting linguistic pattern: S*	

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
Pattern selected: S -> NP VP	The basic sentence pattern is typically chosen for expressing *predications*. The representation of this type of linguistic knowledge, presented as a grammar rule here but incorporating other knowledge in Ace as well, is discussed in Chapter 4, and the pattern selection process is covered in Chapters 7 and 8.
Filling out selected pattern	
Now generating from: john1 (case nominative)	*John1* is the concept which corresponds to the noun phrase of the basic sentence. The mechanism which allows the generator to relate *john1* to the NP is called *restriction* and is described in Chapter 7. This mechanism also applies knowledge about the basic sentence to add the constraint that the subject must have nominative case. *Restriction* thus "fills out" the selected pattern, and starts the generator working on the first part of that pattern.
Expanding token	
Now generating from: (person (name John))	*John1* is a *token* in the system, representing a unique object. Expansion of this token, or retrieval of the knowledge about the object, is a simple part of the generation process.
No related views found. *Applying referential relationships* *Selecting linguistic pattern: NP*	

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
Pattern selected: NP -> NM	This pattern is selected using the knowledge that names refer to people, and that they constitute NPs.
Filling out selected pattern	
Finding word--word found: "John"	The name "John" fills the role of the NM part of the pattern. KING has the knowledge that this part may be a single lexeme, and thus completes its first word.
Now generating from: (recipient (of (kiss-giving ...)))	The program is now generating the predicate part of the NP-VP pattern. As in most generation programs, KING walks through each pattern and generates from each constituent of that pattern until the sentence is completed. The control of this process is described in Chapter 8.
No related views found. *Applying referential relationships*	
Constraint produced: [passive voice]	KING uses the knowledge that the *recipient* role is expressed using a passive VP. This type of knowledge, which relates concepts to linguistic structures or constraints, is represented as explicit referential knowledge using an association called *REF*, described in Chapter 6. REFs are applied during the mapping process.
Now generating from: (kiss-giving ...)	KING is now generating the rest of the predicate.

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
No related views found. *Applying referential relationships* *Selecting linguistic pattern: VP*	
Pattern selected: VP -> VP ADJUNCT	This pattern is chosen to express the actor of an action referred to in the passive voice. The VP ADJUNCT pattern, used here to build "by Mary" onto the passive verb phrase, is discussed in Chapter 5, and the method of selecting this pattern is considered in Chapter 8.
Selecting linguistic pattern: VP	
Pattern selected: VP -> V-P NP	This pattern is chosen to express the direct object, which refers to *kissing*. Like the previous pattern, this verb phrase form is chosen as a means of generating a linguistic relation derived from the original concept. These relations will be described in Chapter 5. The reason KING selects another pattern here without further mapping is that the concept expressed by the VP, the *kiss-giving* concept, has already been mapped from.
Selecting linguistic pattern: V-P	
Constraints are: [lex_give] [voice passive] [person third] [number singular]	Agreement constraints, lexical categories, and derived constraints are all applied each time KING starts to produce a new pattern element. These are the constraints on the verb part of the sentence.

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
Pattern selected: V-P -> HELPER V-P	Knowledge that passive verb parts require helping verbs is used to to select this compound verb pattern. The organization of compound verbs is developed in Chapter 4.
Selecting linguistic pattern: HELPER	
Constraints are: [lex_be] [person third] [number singular]	The knowledge that the passive helping verb is "be" is attached to knowledge about the passive voice; the constraint that the helping part of the verb is the part which must agree is knowledge about the HELPER V-P pattern.
Finding word--word found: "was"	There is now sufficient knowledge to complete the next lexeme, using the past tense as a default.
Selecting linguistic pattern: V-P	
Constraints are: [lex_give] [form perfective]	Knowledge that the passive verb part is in perfective form is also attached to the passive voice constraint. That the lexical category obtained from the concept to be expressed is the lexical category of this part of the complete verb is knowledge about compound verbs in general.
Pattern selected: V-P -> V	Since there are no more "helping" relations (such as the "be" helper above) to express, the generator now selects this simple pattern to complete the verb.

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] + *Process Initiated*	Explanation
Finding word--word found: "given"	
Now generating from: (kissing (surface cheek1))	In walking through the verb phrase, KING is now producing the NP part of the VP, which refers to *kissing*.
No related views found. *Applying referential relationships* *Selecting linguistic pattern: NP*	
Pattern selected: NP -> NP PMOD	This pattern, a noun phrase followed by a modifier, is chosen to express the *surface* relation. The preposition "on" is also selected at this point, but is not used until the modifier PMOD is generated.
Selecting linguistic pattern: NP	
Pattern selected: NP -> DET NP*	This basic NP pattern, an article-noun construct, is selected by default to refer to *kissing*.
Selecting linguistic pattern: DET	
Constraints are: [ref indef]	Default for referring to "kiss" given The means of deriving such defaults, which depend on the type of concept being expressed, are discussed in Chapters 6 and 7.
Finding word--word found: "a"	
*Selecting linguistic pattern: NP**	
Pattern selected: NP* -> N	This pattern is used for "kiss" because of the lack of further modifiers.

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] + *Process Initiated*	Explanation
Selecting linguistic pattern: N	
Constraints are: [lex_kiss]	Mapping from the *kissing* concept earlier using explicit referential knowledge resulted in the instantiation of the lexical category *lex_kiss*. This category is then applied to the noun part of the noun phrase using general knowledge about noun phrases.
Finding word--word found: "kiss"	
Now generating from: cheek1 [prep_on]	Walking through NP PMOD pattern, KING generates the postmodifier part.
Expanding token	
Now generating from: (cheek (part-of john1))	The expansion of the token *cheek1* results in this concept.
No related views found. *Applying referential relationships* *Selecting linguistic pattern: PMOD*	
Pattern selected: PMOD -> PP	This prepositional phrase pattern is used for the linguistic relation between the preposition "on" and its object, used here to refer to the *surface* relation.
Selecting linguistic pattern: PP	
Pattern selected: PP -> PREP NP	The only prep phrase pattern is used.
Selecting linguistic pattern: PREP	

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
Constraints are: [lex_on]	This lexical category was derived from the *surface* relation.
Finding word--word found: "on"	
NP -> DET NP*	The basic NP pattern is the default for referring to objects, such as "the cheek".
Finding word--word found: "the"	
*Selecting linguistic pattern: NP**	
Constraints are: [lex_cheek]	This lexical category results from the *cheek* concept.
Pattern selected: NP* -> N	The basic NP*, used for the complete noun part of the NP, is chosen because because all other relations involving *cheek1* have been expressed.
Finding word--word found: "cheek"	
Now generating from: mary1	The generator is now producing the "by" adjunct phrase to express the actor of the *kissing*.
Selecting linguistic pattern: ADJUNCT	
Pattern selected: ADJUNCT -> PP	"By" adjuncts are expressed using prepositional phrase patterns.
Selecting linguistic pattern: PP	

Concept Considered *or* Pattern Considered [+ special constraints] [+ extra info.] *+ Process Initiated*	Explanation
Constraints are: [prep_by]	Further knowledge about "by" phrases is that they involve a particular preposition. Like many of the other constraints here, this stipulation is derived by the generator in a manner described in Chapter 7.
Selecting linguistic pattern: PP	
Pattern selected: PP -> PREP NP	The single prepositional phrase pattern is again selected for the final prep. phrase.
Selecting linguistic pattern: PP	
Constraints are: [lex_by]	This lexical constraint is attached to the [prep_by] constraint.
Finding word--word found: "by"	
Now generating from: mary1	KING is now generating the final constituent, the object of "by".
Expanding token	
Now generating from: (person (name Mary))	Expansion of the token *mary1*
Selecting linguistic pattern: PP	
Pattern selected: NP -> NM	Knowledge that name NPs are used for for people is again applied to refer to Mary.
Finding word--word found: "Mary"	

```
*****************************************************
                     OUTPUT
*****************************************************

John was given a kiss on the cheek by Mary.

= = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

1.7. The Thesis

The claim of this thesis is that particular features of a knowledge representation framework are essential to the task of building practical, adaptable, and extensible generators. Specifically, the traits of uniformity, hierarchy, and the explicit representation of referential and metaphorical relationships among concepts are necessary for the exploitation of abstractions in the representation of knowledge about language. A knowledge representation system with these features empowers the development of a knowledge-intensive mechanism with a strong language capability. The support of this thesis lies in the nature of the representation and the successes of the implementation.

Chapter 2 presents an overview of research on language generation and related areas, focusing on the theory and implementation of existing systems and highlighting some of the practical problems with these systems. The discussion is meant to provide an update of the state-of-the-art in language production as well as the motivation for the research presented here.

Chapter 3 discusses the theoretical problem of exploiting generalizations in language generation, and presents the fundamentals of the Ace approach to this problem.

Chapters 4, 5, and 6 cover in detail the Ace knowledge representation framework, a set of tools for the representation of linguistic and conceptual knowledge, and provide examples of the knowledge encoded, contrasting it with other representations. These chapters are addressed primarily to the technical reader interested in the relation of language to knowledge representation. Chapter 4 presents the fundamentals of the Ace framework, Chapter 5 shows how this framework is applied to linguistic representation, and Chapter 6 focuses on the association of linguistic and conceptual knowledge.

Chapter 7 discusses the implications of the Ace representation on language processing, and outlines the overall processing strategies and design of the KING generator.

Chapter 8 describes the implementation of the KING generator, a knowledge-driven mechanism for performing language generation using the Ace representation. The implementation details are mainly provided for the reader who is specifically interested in the technical aspects of the generation task. A more thorough analysis of the trace given here is presented also in Chapter 8.

Chapter 9 analyzes the contribution of this work and suggests areas for future research.

Two appendices are provided specifically as an aid for readers who are interested in building natural language programs. Appendix A provides representative linguistic knowledge used by KING, and Appendix B supplements the examples of knowledge representation in the text.

2. Survey of Related Research

The development of natural language generation systems has been the focus of only a small body of research. The generation task, however, has evolved under the constant influence of advances and trends in artificial intelligence, linguistics, philosophy, and psychology. This discussion will trace the evolution, focusing on the current research issues in language generation as they relate to other relevant topics. These topics include the study of plans, intentions and communicative acts; the representation of linguistic knowledge; the production of connected texts, stories, and explanations; and the analysis of coherence and focus constraints in dialogues.

2.1. The Shaping of the Generation Task

The language generation problem has its roots in the goal of finding a sufficient specification of the constraints governing grammatical sentences (Yngve, 1962; Friedman, 1969). While this objective is central to the generative linguistic paradigm, work within this paradigm failed to address an issue of critical importance to the construction of language generators in Artificial Intelligence systems. As such systems were conceived, the need arose for the description not only of the constraints on grammatical output, but also of the process by which a *choice* could be made among possible outputs. The problem of language generation as a communication tool for computer systems became dependent on three major considerations: (1) *linguistic representation* -- the knowledge which describes the language that the system is generating *into*, (2) *knowledge representation* -- the internal structures which the system is generating *from*, and (3) *the choice problem* -- how the intricate relationship between the underlying knowledge and linguistic knowledge determines the appropriate output.

The choice problem in many AI natural language systems was greatly simplified by the limited domains within which the systems operated. The limited scope restricted both the nature of the knowledge to be expressed and the minimal quantity of linguistic knowledge required to satisfy the communicative requirements of the programs. While the goal of minimizing restrictions on the input made robustness an important requirement for the input analyzers, robustness of output had no such motivation. Systems such as Winograd's (1972) made use of relatively sophisticated models for language understanding, but employed a minimal set of simple rules to guide language production.

Some of the early work on language generation took advantage of research addressed to language analysis, which had led to the development of representations of linguistic knowledge such as Augmented Transition Networks (Woods, 1970). Simmons and Slocum (1972) adapted an ATN grammar to produce linguistic output from a semantic network, using a variety of semantic tests and "paraphrase rules" at each arc in the ATN to guide the production of output. Shapiro (1975) explicitly modeled generation as the inverse of the parsing process using an ATN grammar. Generation using an ATN, like parsing, consists of tracing a path through a network of transitions. The main problem is to determine which path to follow from each state in the network. This determination has generally been made by applying tests, or queries, to select a path based on what is being expressed.

The BABEL generator used in the MARGIE system (Goldman 1974, 1975) also used an ATN grammar to apply syntactic constraints in the generation of language. Like the Simmons and Slocum system, Goldman's algorithm produced sentences by first selecting a verb, using queries as to the nature of the structure of the semantic input. These queries were organized into a discrimination net which distinguished verbs by their conceptual content. Starting from a Conceptual Dependency representation (Schank, 1975), the system selected a discrimination net based on what "primitive act" was to be expressed and traversed the net according to particular aspects of the input concept. For the primitive act "INGEST", for example, BABEL would test whether the object being ingested was air, in which case it would select the verb "breathe". If the object was smoke, it would choose the verb "smoke", if the object was fluid, "drink", and solid, "eat". The bases for lexical choices were thus explicit within the system.

While the ATN-based generation systems had the capacity to produce a wide range of outputs, their ability to use their linguistic capabilities was hampered by the difficulty of encoding the appropriate rules and heuristics to guide the choice process in many circumstances. The addition of a new lexical item to such systems required the encoding of new queries to distinguish the conceptual basis for the use of the lexical item from concepts which would be expressed differently. Since the generators dealt with fairly general concepts, equipping them with sufficient knowledge to differentiate one complex concept from another was a difficult task, posing a practical limitation on their linguistic capabilities. Other programs achieved greater fluency by operating in domains in which the conceptual knowledge was highly constrained. The tic-tac-toe program of Davey (1979) and the psychoanalytic patient of Clippinger (1974) were able to imitate the use of complex linguistic phenomena effectively within a limited context.

While appearing to demonstrate a substantial language capacity, most language generators operating within severely restricted domains failed to set the tone for further research. Their main handicap was the same as that which made other early AI programs quickly lose their appeal: The model they provided was not sufficiently general to be extended or adapted to new applications. The work of Simmons and Slocum and of Goldman, however, has proved to be of more than historical interest when taken independently of the question-answering tasks for which their generators were used. Many of the problems identified by these systems as well as approaches to their solutions have continued to have an influence on the field. The principal interrelationships among linguistic representation, knowledge representation, and mechanisms to produce appropriate output are all highlighted by this research.

Analysis

A major result of the early research on language generation was the recognition of important problems which had generally not been explored within the generative linguistic paradigm. The focus of computational linguistics has been on the description of linguistic competence, rather than on the forces which control the use of language. Models of linguistic competence encode the knowledge required to guide the syntactic correctness of a generated output, but fail to encode the knowledge necessary to guide the *appropriateness* of the output. Early language generation systems reacted to this deficiency either by adding large amounts of *ad hoc* rules and heuristics to a syntactic model, or by attempting to describe an independent model of language production within a limited domain. The first approach, evidenced in the adaptation of ATNs for generation, has the awkward consequence of treating knowledge about language *use* as fundamentally different from knowledge about language. The second approach, evidenced in the more restrictive systems, often ignores the role of a competence altogether, and treats the production of language as a phenomenon distinct from language analysis or competence. Both approaches achieved substantial successes, and together pointed the way towards the refinement of linguistic models of knowledge *about* language, models which could be used to describe the knowledge required for competence, analysis, and generation, rather than treating each individually.

A second essential factor in the generation task made apparent by the earlier research is the sensitivity of the language generator to the representation scheme used for encoding its semantic input. An important part of Goldman's generation mechanism was the discrimination net used to select verbs. The need for this mechanism was due primarily to the use of a small set of primitives in the Conceptual Dependency representation. Each primitive act, such as "INGEST" or "ATRANS", could potentially lead to the generation of a wide range of verbs, depending on the results of the tests in the discrimination nets. Other systems which employed a richer set of predicates were able to avoid the need for these tests by having a closer correspondence between lexical items in the language and predicates in the representation. But the proliferation of predicates made it difficult to explain how surface utterances similar in meaning could be produced from different representations. In Goldman's generator, the similarity in meaning was represented at the level of the underlying knowledge representation. In Simmons' and Slocum's generator, it was represented by the use of paraphrase rules at the lexical and syntactic levels.

Observation: The nature of the knowledge representation used is critical to the task of choosing linguistic output.

Related to the underlying representation and the grammatical representation used for generation is the question of how the knowledge representation influences the choice process. In many of the early generation systems, as well as in programs designed using the early systems as a model, the knowledge used to guide the choice process was specifically encoded as choice knowledge. In effect this knowledge was specified as a procedure which the generator would execute to determine the appropriate choice of output. Unfortunately, the encoding of knowledge specifically as knowledge about the choice process makes it difficult to take advantage of generalizations and to use the same knowledge for understanding. Intuitively, it seems that much of the knowledge encoded as choice knowledge is in fact knowledge about the relationship between language and meaning, which ideally could be used for both analysis and generation. Confining this information to the choice points makes it near impossible to use it in analysis; furthermore, it makes the consideration of many unlikely choices inevitable, since only after examining a particular choice can its applicability be determined.

Observation: The choice of language seems to depend on the interaction of linguistic and conceptual knowledge.

A distinction which has prevailed throughout the short history of language generation is the division between deciding *what* to say and *how* to say it. This distinction was enforced in most of the early systems, which required that decisions about the conceptual content of the output be made prior to decisions about its linguistic structure. As the field has developed, the independent consideration of the conceptual decisions and linguistic decisions has remained convenient. However, the strict division of language generation systems into conceptual choice and linguistic choice mechanisms poses certain difficulties: It assumes that the conceptual decisions provide enough information to specify completely the knowledge necessary to guide the linguistic decisions. Thus, if there are linguistic constraints which have associated with them conceptual constraints, it may often be necessary to reconsider the conceptual decisions. Thus a system which maintains the separation between the "what to say" and "how to say it" components demands considerable interaction between these components.

The successes and limitations of the early work on generation shaped the task of automatically producing language into the interaction of several goals, among them the following:

-- the refinement of declarative representations of linguistic knowledge suitable for use in both analysis and generation
-- the development of models of the relationship between linguistic knowledge and underlying conceptual structures
-- the construction of generation mechanisms operative within a variety of domains
-- the interaction of conceptual and linguistic knowledge of varying levels of specificity within the generation task

2.2. Generation as Language Planning

Much work on language generation in particular, and language processing in general, has either implicitly or explicitly assumed language processing to be a mapping between language and meaning. In other words, the role of the language generator would be to produce a linguistic structure as a direct translation of a representation of its meaning. This assumption proves adequate within the confines of very simple question-answering and description tasks, where the function of the language being produced is quite specific; i. e. to inform a user of the answer to a question or to provide a description of an object or event. Language in general, however, is not restricted to this model.

The notion that the meaning, or propositional content, of an utterance does not always adequately describe its communicative function dates back to early work in the philosophy of language. The modern study of this idea, generally described as the theory of speech acts, was inspired by Austin (1962), who pointed out that statements such as "I promise to do my homework" do not really convey information about a fact about the world but rather serve to commit the speaker to carrying out a particular task. The examples given by Austin primarily fall into the class of *performatives*, statements which themselves affect the world rather than describe it. Austin argued that performatives such as "I promise to take out the garbage," "I hereby pronounce you husband and wife," and "I order you to leave" illustrate a class of utterances which could not be described by classical truth-valued logic, since these examples serve to communicate more than a proposition about the world.

Austin distinguished a number of classes of *illocutionary acts*, or actions involving the production of language. An illocutionary act is the use of language to convey a *perlocutionary effect*. The realization of an illocutionary act embodies an *illocutionary force* and a *propositional content*. The illocutionary force of an utterance is the effect which it conveys on the world, not necessarily directly related to its propositional content. These distinctions were clarified in the work of Searle (1969, 1979a, 1979b), who formalized and popularized the speech act theory. Searle presents a taxonomy of illocutionary acts encompassing five categories:

(1) *directives* e. g. "Take out the garbage!"

(2) *commissives*, e. g. "I promise to be home by six,"

(3) *representatives*, e. g. "I took out the garbage,"

(4) *declarations* e. g. "I pronounce you husband and wife,"

(5) *expressives*, e. g. "I'm sorry I insulted you."

The illocutionary force of the above examples may be evident from the nature of the utterance. For example, "I'm sorry" explicitly expresses an apology. However, the illocutionary force of an utterance is not necessarily conveyed by its surface form. For example, "Can you pass the salt?" is on the surface a question, but its illocutionary force may be that of a directive. "Play ball" may be considered to be both a directive and a declaration, as its illocutionary force is both to order the players to begin the game and to mark the official start of play. Utterances whose illocutionary force is different from that which would ordinarily be conveyed by their surface form are known as *indirect speech acts*.

Searle thus establishes several levels of description of an utterance: The illocutionary level identifies the intention conveyed by the utterance. The propositional level describes its meaning content, and the utterance level its linguistic realization. Language as communication in the speech act theory may thereby be viewed at several levels of abstraction; thus the problem of relating linguistic representation to meaning representation does not itself fully describe the knowledge required to construct an utterance when its illocutionary force also is to be considered.

The introduction of the speech act theory into the linguistic components of artificial intelligence systems is due primarily to the work of Bruce (1975) and of Perrault, Allen, and Cohen (*cf.* Cohen and Perrault, 1979). In the OSCAR system at Toronto, language analysis was modeled as the identification of the user's goals based on the linguistic input, and language production the realization of the system's goals (generally matching those of the user) in the output. In addition to recognizing basic surface speech acts, OSCAR handled interactions between the system and user such as the following:

User: What time does the train for Montreal leave?
System: 6:00 at track 9.

In this type of response the system identifies the user's goal of wanting to know when a train leaves and infers that the user will also need to know the track number. The goals of informing the user of the time and track number are subsumed in the response.

The consideration of speech acts and intentionality in language processing led to new metaphors for language generation. Cohen (1978) models language generation as a *planning* process, in which the choice process is guided by the goals that the speaker wishes to satisfy in producing the utterance. Treating language production as planning avoids the oversimplistic view of generation as direct translation, but gives birth to new challenges, such as describing the role of language in satisfying the communicative goals of the speaker.

Appelt (1982) designed and implemented a complete language generator driven by a general purpose multiple-agent planning system. The system used a hierarchical planner, KAMP, to work from the input goals and by successive refinement specify a fully-realized plan for the output of an utterance. The system was able to perform such complex language planning tasks as the production of references sensitive to knowledge about the hearer, and the combination of utterance acts with other actions, such as pointing. The planner produces utterances such as *"Remove the pump with the wrench in the tool box"* given a formal description of the systems goals and user's knowledge in an assembly task.

Appelt describes four levels of abstraction in the production of an utterance. The highest, as in Searle's description, is the illocutionary level, but Appelt defines a *surface speech act* level between the illocutionary level and the *concept activation level*, which corresponds to Searle's propositional acts. The lowest level, like Searle's, is the utterance level. By axiomatizing the knowledge at each of the four levels, Appelt enables the planner, with interactions among the various levels, to take advantage of knowledge of different degrees of abstraction to produce a successful utterance.

Appelt's system currently provides the most complete model of language generation. The use of a complex planning mechanism and a model of belief based on the possible world semantics (Moore, 1980), allows the program to take into account the goals and beliefs of speaker and hearer as well as of knowledge about the task domain and knowledge about language at a range of levels of abstraction. The language planner can theoretically operate within a variety of domains; equipped with sufficient knowledge, the same program can be used to perform a range of communicative functions.

Analysis

While the implementation of the KAMP language planner demonstrates the possibility of combining language generation with other planning activities, the system has a number of interesting problems. An important practical difficulty with the program is its inability to operate in real time: Many phases of the generation task require a surprising number of cycles, adding up to a program which cannot reasonably be used within a natural language interface. A number of causes of inefficiency prove to be technical problems that could have easily been overcome with additional tuning; others are inefficiencies in the planning model. The operation of the planner requires the consideration of a great number of options at each level and the evaluation of each, rather than selecting a possible plan and attempting to carry it through to linguistic realization. The consideration of many options at each stage in the generation process, as well as the ability of the system to back up at any given stage, are time-consuming characteristics of the system which also seem at odds with the way human beings produce language.

This problem is similar to a drawback of many generation systems: The decision-making model adopted fails to provide a practical way of limiting alternatives, and forces the equal consideration of unlikely possibilities. Without a mechanism for quickly determining *which* choices are worthy of consideration, the production of appropriate utterances presents a major search problem. In BABEL, the difficulty was evidenced mainly at the level of lexical choice, where "smoke", "breathe", and "eat" could all be evaluated because of a common conceptual primitive. In KAMP, the proliferation of alternatives may appear also at higher levels. In actual cooperative tasks such as those modeled by KAMP, "Remove the pump with the wrench in the tool box" may be used regularly in contexts where many other possibilities, such as "The pump may be removed with the wrench" or "The wrench will remove the pump" are equally appropriate. In the KAMP model, giving the generator the ability to produce all these forms forces the generator to consider them as alternatives.

Human beings are equipped with the capacity to express themselves in a broad range of forms, but do not seem to consider many alternatives, particularly in spoken language. For example, there is a tendency to be consistent with previous sentence structures or descriptive terms. If a person is asked, "What will remove the pump?", the response "The wrench will remove the pump" is favored. Evidence suggests (*cf.* Cohen, 1985) that such responses are not evaluated with respect to other possible utterances, as there is increased fluency in producing such responses.

Observation: The manner in which knowledge is retrieved biases the generation process.

The consideration of the way in which language is often used indicates that certain factors serve as aids to the generation process by *biasing* the retrieval of structures, rather than by influencing the evaluation of candidate structures. This bias may even cause the speaker to ignore the surface speech act implications of an utterance; for example, in cooperative dialogues speakers use imperatives without really intending to command. It is difficult to have this flexibility in a hierarchical planning or decision-making model, where decisions at the surface speech act level are evaluated and made before lower levels are considered.

In addition to a variety of problems with efficiency, certain representational problems are apparent in Appelt's system. The possible worlds representation, which allows for the encoding of belief knowledge into potentially infinite sets, poses some awkward stumbling blocks for the planner. This seems not to be indicative of any major difficulty with the planning model, but rather of some of the issues now being worked out in the developing field of the semantics of belief. Problems with linguistic representation present themselves as well: The linguistic representation in KAMP is limited, handling no relative clauses or complex modifiers and only the more basic surface structures. This de-emphasis was due to the focus of the system is on the integration of language into a general multiple-agent planner. Appelt (1983) has since employed a unification grammar for the linguistic knowledge of the generator, as it facilitates the encoding of discourse information. Unification grammar will be discussed in the next section and in Chapter 5.

Considering language production as a planning process adds a new dimension to the generation task by requiring the integration of knowledge of goals, intentions, and communicative acts with that of language and meaning. While the "planning" metaphor seems best applicable to the strata of deciding "what to say" rather than "how to say it", it provides a model powerful enough to describe the multiple levels of the task as well as the interaction among these levels. There are some practical problems with the treatment of language strictly as a planning process. The most formidable of these are the solidification of the representation schemes used and the discovery of methods of controlling the search through the large space of potential utterances.

2.3. Linguistic Representation

Quite a number of linguistic formalisms have been used in natural language-based Artificial Intelligence systems, but the field of language generation has consistently played favorite to a handful of linguistic representation schemes. The generators of Goldman and of Simmons and Slocum have led to considerable work using Transition Networks similar to those described by Woods (1970). Most of the other recent generation systems have employed some notation within the systemic/functional tradition, particularly systemic grammar (Halliday, 1967) and functional grammar (Kay, 1979), which has come to be known by its current variant, called unification grammar or functional unification grammar (Kay, 1984). It is often difficult to identify strict distinctions among these linguistic formalisms, as their adaptation to the task of language production serves rather to highlight their many similarities.

The Augmented Transition Network formalism contributed substantially to the development of natural language interfaces by establishing a means of organizing grammars which lends itself well to computer implementation. Each state in an ATN represents some point in the analysis or generation of an utterance, and each arc leading from a state represents a possible linguistic structure which may occur at that point. One problem with ATNs, as discussed earlier, is encoding within the transition network the information required to select, at any given state, the most appropriate arc during generation. Related to this is the difficulty of associating semantic properties with the various arcs in the network. The ATN formalism proves more easily adapted to parsing than to generation, because the traversal of the ATN is necessarily driven by linguistic structure. Conceptual and structural choices in generation must thus be made only when required at a particular stage in the traversal of the network; it thus becomes awkward to implement an ATN generator where such choices are made in an order different from that in which their corresponding linguistic structures appear.

Unlike ATNs, systemic grammar presents a linguistic formalism in which conceptual and structural choices may clearly be separated from surface order constraints. In systemic grammar, these choices are made by *choice systems*, which operate on sets of *features*. Features may be used to convey specific conceptual, grammatical, or lexical information, ranging from high-level semantic qualities such as *intensional* or *collective* to low-level lexicographic information. A system takes as input an expression of initial features to be realized, and produces a *selection expression,* the set of grammatical features chosen based on these constraints. As in a discrimination net, each step in the traversal of a choice system may involve querying the membership of a particular feature in the input expression and selecting a grammatical feature which depends on the result.

The production of an utterance using systemic grammar (Mann, 1982) proceeds from the application of a choice system to the invocation of a set of *realization operators,* which produce grammatical structures, lexical features, and ordering constraints from the chosen set of grammatical features. Each realization operator carries out a particular task within the general domain of furthering the specification of the final output. The joint application of these operators results in the complete specification of the surface structure and lexical choices in the output.

The systemic grammar formalism is attractive for the purposes of language generators because the choices necessary in the formation of an utterance are explicit within the grammar, as well as the knowledge required to realize a set of choices. Much of the information contained in the systemic grammar is semantic or functional knowledge associated with the choices, making this knowledge part of the model rather than an awkward addition to the model. Systemic grammar is unusual in that it has developed since the earlier days of the generative paradigm, yet is specifically designed to describe language *use.*

Ironically, two of the attributes of systemic grammar which make it effective for language generation also provide evidence of potential difficulties with the formalism. The first of these is the diversity of realization operators, and the complexity of the relationships among these operators. These operators are convenient for the production of language because they represent direct correspondences between constraints and surface realizations. Unfortunately, as systemic grammars grow the intricacies of the relationships among realization operators become more and more significant and it becomes extremely difficult to formulate realization rules. The construction of large systemic grammars is thus a monumental task; however, it is not clear whether the problem is in fact more difficult than it would be using another formalism or whether such complications are intrinsic to the task of developing large grammars.

The second feature of systemic grammar which can prove problematic is the tendency to underspecify the language being generated. This tendency is in fact indicative of a major practical difference between grammars primarily used for generation and those generally used for language analysis. In analysis, having a grammar which underspecifies a language leads to the negative result of increasing the range of inputs which the system will fail to recognize. Overspecifying a language, on the other hand, simply increases the likelihood of treating an ungrammatical input as being acceptable, which is hardly much of a concern in most natural language interfaces. Overspecification has disastrous effects for generation, however, as a system which actually produces ungrammatical output will be deemed inferior to one which produces a restricted set of grammatical outputs. In the case of systemic grammar, underspecification often results from a failure to identify the relationships among features produced by the choice networks, thereby limiting the ability to realize these features.

Case grammar (Fillmore, 1968) has had an influence on the representation of both linguistic and conceptual knowledge used for generation. Case grammar is not really a "grammar" at all, but a set of basic relationships which may be used to associate linguistic structures that are similar to one another in meaning. While these relationships are in principle compatible with most linguistic formalisms, some of the basic principles of case grammar have led to modifications of existing representation schemes or to changes in the manner in which they are used.

The most important impact of case grammar is to suggest the categorization of knowledge about the meaning of language which is reflected in the structure of language. These categories are exemplified by the "cases" in the grammar. The *agent* case*, for example, represents a particular relationship between the motivator of an action and the action, and corresponds closely to the subject of active "action" verbs. The *recipient* case suggests one that receives a transferred object, and is often realized linguistically by an indirect object, the object of the preposition "to", or the subject of a passive "transfer" verb.

Within grammar formalisms, case categories are generally described as being properties of particular linguistic structures, or features. Cases represent a particular class of features which may be related directly either to surface constraints or to the conceptual content of an utterance. Because cases may thus be implemented as a type of feature, case grammar affects more the nature of the knowledge encoded in linguistic formalisms than the notational constraints on the formalisms themselves.

Functional unification grammar (Kay, 1984) is one of the more recent products of the systemic/functional tradition and lends itself well to the design of language production mechanisms. The formalism, like systemic grammar, belongs to the class of feature and function notations, where constraints on linguistic structures are determined by sets of features. Features are attribute-value pairs which may be used to represent a wide range of information as in systemic grammars--morphological, lexical, syntactic, and functional knowledge. The value of a feature may be a literal, special symbol, or a composite set of features.

An important feature within a linguistic structure in unification grammar is the "pattern" feature, which specifies the surface location of each constituent specified by the feature set. The term "pattern" is often used to specify the pattern along with the associated features in the set. During generation, patterns are selected based on an input "functional description", or "FD", which is matched against the linguistic patterns of the grammar to determine the linguistic output.

The following is an example of a simple unification grammar pattern†, representing the characteristics of verbs used with helping verbs, such as "has gone":

* The particular set of cases, as well as the names for the cases, varies drastically from one system to another. The terms used above are among those which are most consistently used.

† More detailed examples of unification grammar will be presented in Chapter 5.

```
[
CAT = verb
PATTERN = ( HELPER VB )
FORM = perfective
HELPER = [
        CAT = verb
        ROOT = have
        TENSE = ^ TENSE
        PERSON = ^ PERSON
        NUMBER = ^ NUMBER
        ]
VB =    [
        CAT = verb
        FORM = participle
        TENSE = past
        ROOT =^ ROOT
       ]
]
```

In the above, the "^" is used to mark features whose value corresponds to the given feature of the matched FD. This pattern could be matched against the following input FD:

```
FD =
 [
  CAT = verb
  ROOT = go
  TENSE = present
  FORM = perfective
  PERSON = third
  NUMBER = singular
 ]
```

The matching of this FD with the given pattern would result in the eventual production of the verb "has gone", based on the joining of features from the FD with the features of the matched pattern.

What would ordinarily be represented by a range of nonterminal productions in traditional grammars is represented implicitly by the alternation of patterns in unification grammar. Instead of rules of the form (S -> NP VP) a unification grammar knowledge base has information of the form (S = [CAT = sentence ... { t1, t2, t3, ... }]), where { t1, t2, ... } represents the alternative sets of features which the sentence may have. Thus there are no grammar rules *per se*. Choice information is explicitly represented by alternation within feature values. A feature of the form F = { t1, t2, ... } thus represents a set of alternative choices which may be evidenced in the linguistic structure produced. The "grammar" is entirely contained within the set of alternative sentence patterns. The "unification" in unification grammar refers to the process by which these patterns are matched against the input, which in generation consists of a complete functional description of the utterance to be produced. This functional description is then "unified" with the grammar, by a process which is quite similar to logical unification. The unification process is applied recursively through the patterns contained in the grammar until the output surface structure and lexical choices are completely specified.

Unification grammar has the advantages of systemic grammar for generation: the ability to express features corresponding to various levels of utterance realization, the encoding of functional information, and the explicit representation of alternatives within the grammar. Furthermore, unification grammar has the attractive advantages of a uniform representation scheme for various types of linguistic knowledge and thus the ability to apply the unification process recursively until an utterance is successfully completed. This simplicity has made the use of unification grammars within actual AI systems possible. Unfortunately, along with the simplicity of the unification algorithm comes its computational complexity, which causes inefficiency unless the consideration of alternatives is somehow constrained.

Unification grammar bears a strong similarity to the *pattern-concept pair* representation conceived by Wilensky (1981). Pattern-concept (PC) pairs, like unification grammar, were intended to provide a declarative representation to be used for both analysis and generation. The unification grammar notation and PC pairs are compatible, but the PC pairs stress particular aspects of language use: They facilitate the encoding of specialized linguistic knowledge and enable the interaction of conceptual and linguistic knowledge in language processing. For example, the following pattern-concept pair corresponds to the use of the verb "remove":

```
[
PATTERN = ( <agent> <p-o-s = verb  root = remove> <object>
                  <<word = from> <container>> )
CONCEPT = (state-change
            (object ?OBJECT)
            (state-name location)
            (from (inside-of (object ?CONT)))
            (to (not (inside-of (object ?CONT)))))
TENSE = (value 2 tense)
OBJECT = (value 3)
CONT = (value 5)
FORMS = (active-s passive-s)
]
```

Specifications of constituents in angle brackets (<>) includes linguistic information (ROOT = remove) or conceptual categories (agent, container) or a combination of linguistic and conceptual specification. Additional information associated with each PC pair determines the correspondences between elements of the conceptual structure and constituents of the linguistic structure: The special "value" indicator designates the association of a property of the PC pair with a property of one of its constituents, specified by number. Thus "TENSE = (value 2 tense)" implies that the tense of the pattern is the tense of the second constituent, the verb. "CONT = (value 5)" indicates that the token unified with the variable "?cont" in the conceptual template corresponds to the fifth constituent, the object of "from". This PC pair could be used, depending on the concept being expressed, to produce the sentence "You should remove the files from your directory" or the infinitive phrase "to remove a file from the top level directory". The use of a PC pair in building a linguistic structure requires the *unification* of the concept part of the PC pair with the concept to be expressed, *elaboration* of the constituents of the pattern part to include information obtained from this unification, and *combination* of the PC pair with another pattern which determines surface order.

Analysis

Among the similarities of the grammar formalisms discussed above is the potential for the encoding of functional and semantic information into the features which constrain linguistic output. This result can be achieved in ATN grammars adapted for generation by using register sets to represent this information. For guiding the generation process, the use of semantic and functional features is important because this information influences the choice process. A second similarity is the explicit representation of the choices available at each stage in the generation process. This distinguishes ATNs, systemic grammar, and unification grammar from traditional grammars and makes them attractive for language generation systems.

Two primary considerations raised here are (1) how to represent the information required to guide the choice process, and (2) how to describe the relationship, if any, between the linguistic features explicitly represented in the grammar and conceptual knowledge in the system. The correspondence of syntactic features to semantic and pragmatic knowledge is often nebulously defined: for example, information about cases, presumably used to describe the underlying conceptual structure of language, is often encoded strictly as syntactic knowledge in grammatical representations. "Choice knowledge" in generation systems is often represented procedurally at choice points in the linguistic representation. Isolating knowledge about syntax, syntactic choice and lexical choice from conceptual knowledge may limit the ability of a system to take advantage of generalizations. These representational concerns, as well as solutions to other problems in the formalisms discussed here, will be considered in detail in Chapter 5.

Each of the representation schemes discussed here has found its way into implemented natural language generation systems. The ATN formalism was used in the Simmons and Slocum and Goldman generators discussed earlier, and is still popular, particularly in some of the European systems. Systemic grammar and unification grammar are the foundations for linguistic representation in recent programs. The next section covers the design of these programs and their representational and processing strategies.

2.4. Applications of Generation Systems

Much of the recent work on language generation has resulted in computer implementations suitable for use within large-scale AI systems. These successful realizations of language production models highlight both practical concerns and theoretical problems. This section presents an overview of these systems; many of their details will be further developed in future chapters.

One of the most extensive projects within the language generation domain is the MUMBLE generator designed and developed by David

McDonald (1980). MUMBLE is a portable mechanism which can serve as the natural language back end to an expert system or other AI program. Its linguistic knowledge base features extensive grammatical coverage of the English language, carefully separated from any conceptual knowledge to allow for domain-independence. The generator has been used to produce multi-sentential texts from several different internal representations. One example of the output produced by MUMBLE is the following text, produced from a representation of a proof in a logical form used by Chester (1976):

> Assume that there is some barber who shaves everyone who doesn't shave himself (and no one else). Call him Giuseppe. Now, anyone who doesn't shave himself would be shaved by Giuseppe. This would include Giuseppe himself. That is, he would shave himself, if and only if he did not shave himself, which is a contradiction. This means that the assumption leads to a contradiction. Therefore, it is false, there is no such barber.

McDonald divides the generation process into three components: the *expert program,* which handles problem solving or reasoning within the task domain; the *speaker component,* which makes the decision of "what to say" and constructs a "message" describing the content of the utterance to be produced, and the *linguistic component,* which, using its syntactic knowledge as well as a "dictionary" to link domain-dependent components of the message to linguistic structures, produces the final utterance. Identifying himself with the systemic tradition, McDonald presents the function of the linguistic component of his program as a "decision making" process. The production of an utterance involves a sequence of decisions made during various stages in its realization. Much of the knowledge required to control these decisions is embedded in "interface functions" which essentially determine from a message element what features should be attached to its corresponding linguistic structure. These functions, like the dictionary, are reimplemented for each domain in which the generator is used.

The process of constructing an utterance in MUMBLE is analogous to the application of a systemic grammar as described in the previous section. The operation of the speaker component, which determines what message elements are to be passed along to the linguistic component, corresponds to the "choosers" operating in systemic choice systems to select features. The interface functions of the linguistic component, which process each message element and guide syntactic structure and lexical choice, correspond to the realization operators described earlier. The form of message elements in MUMBLE, however, is much more flexible than in most systemic implementations. In fact, most message elements produced by the speaker component of MUMBLE are entries in the knowledge base of the expert program; thus the expert program determines the nature of the message elements.

The interface functions of MUMBLE may be divided into two classes: those which apply the grammar of the system to produce syntactic structure from message elements, and those which apply the dictionary of the system to produce lexical elements from the message. McDonald thus divides the linguistic component into two stages. The first process is the construction of the surface structure of the output, and the second the traversal of this structure while making lexical choices subject to constraints. Lexical and structural decisions are thus separated in MUMBLE.

McDonald attributes a number of the aspects of the design of MUMBLE to a combination of psycholinguistic and practical concerns. His decision-making model, like other systemic approaches, is subject to inefficiency in the search of possible choices. To allay this problem, McDonald makes several assumptions. Notable among these is the "indelibility" stipulation: that once the generator has produced a linguistic element and attached it to the current surface structure tree, the decision to attach it is final. The process is therefore not subject to time-consuming backtracking. The ability to meet this stipulation is provided in part by the division of the linguistic component into two phases, the first stage producing the surface tree and the second stage traversing the tree and applying surface constraints. While structures attached in the first phase are indelible, they may be augmented during the second phase. The operation of the generator thus models the psychologically plausible hypothesis that dynamically augmenting output as it is produced is preferred to rebuilding structure. It is unclear whether the no-backtracking strategy, however, can lead to the failure to produce an utterance that would otherwise be within the capabilities of the system.

The TEXT generation system (McKeown, 1982), like McDonald's program, tackles the language production problem by dividing the task into "what to say" and "how to say it" components, but concentrates on the particular domain of producing explanations in response to questions about databases. Unlike many question-answering problems, the type of question that McKeown concentrates on is the simple question requiring a substantial response:

Q: What kind of data do you have?

A:

> 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.

McKeown's system is driven at a high level by "rhetorical techniques", schemata which represent textual organizations essentially corresponding to specialized plans for producing particular explanations. Attached to each schema are rules for passing information to the next level in the system, which incorporates "relevant" information into the explanation, filling out the schema with details of the explanation. A discourse component adds focusing constraints to the completed schema, before it is passed to the *tactical component* to be realized in English. The schema at this point corresponds to a completed functional description of the output, and a unification grammar is used to produce the final response.

The "what to say" elements of TEXT, which constitute what McKeown refers to as the *strategic component* of generation, divide the role of MUMBLE's speaker component into several levels. The generation process thus consists of a "pipeline" of processes, starting with the application of rhetorical techniques, followed by the "filling", and focusing mechanisms. The "how to say it" process, like McDonald's speaker component, is divided into lexical and grammatical elements, although in TEXT the lexical decisions are made first.

In TEXT, as in MUMBLE, special care is taken to separate the knowledge influencing the generation task at various levels. This has the great advantage of modularity. In MUMBLE and in other generators, modularity is exploited for the purpose of developing a portable generation mechanism. In TEXT, it is exploited in part to allow for concepts such as focus in discourse and rhetorical schemata to be treated in detail by specialized mechanisms. Modularity was also used for division of labor in TEXT, as portions of the tactical component as well as the database representation scheme (McCoy, 1982) were developed semi-independently of the higher level strategic components. This separation of knowledge has its adverse effects, however, as it requires that the tactical component, searching through a unification grammar to satisfy its input constraints, not access any conceptual knowledge to guide its selection. It thus is relegated to a straightforward and potentially time-consuming unification algorithm.

There have been a number of other noteworthy efforts devoted to the development of explanation capacities for expert systems. Swartout (1981) produces coherent medical explanations for DIG, a digitalis therapy advisor. Kukich (1983) implemented a generator called Ana which automatically produces stock reports from daily statistics of the behavior of Dow Jones averages. While these systems are interesting in their own right, it is difficult to isolate their underlying theory from the strict constraints of the knowledge representation from which they generate.

A number of research projects are geared to the task of producing connected texts in general. Meehan (1977) developed a program, TALE-SPIN, using knowledge about the goals of story characters to control the production of fairly lengthy texts. The idea of using "story grammars" (Klein, 1975), extending traditional grammars for use in text analysis and generation, sparked a great deal of debate, but gradually lost popularity. Yazdani (1982) has continued to attempt to automate story writing using a problem-solving approach. Mann and Moore (1980, 1981) have exploited the principles of systemic grammar and made use of the idea of rhetorical points in the undertaking of a major text generation project. This work, while ultimately geared towards the construction of powerful computer programs, has concentrated to date on the development of a large systemic grammar.

Much of the research presented in this thesis was inspired by the development of PHRED (PHRasal English Diction) (Jacobs, 1983, 1985), a natural language generator designed for use in a variety of domains. PHRED was constructed to share a knowledge base of pattern-concept pairs with PHRAN (PHRasal ANalyzer) (Wilensky and Arens, 1980) as part of a real-time user-friendly interface. The knowledge base of pattern-concept pairs is similar to a unification grammar, but the manner in which it is used differs from unification grammars in at least two major ways: First, the interactions among syntactic and conceptual features are facilitated while distinguishing the conceptual nature of conceptual features. Conceptual features are elements of the conceptual structure from which PHRED generates, while linguistic features are either input constraints or are selected during the generation process. Second, the generator accesses its knowledge base using a specialized "predisposition" mechanism to retrieve candidate patterns, applying the expensive unification algorithms only after the predisposition phase has produced an ordering of potential patterns.

The process by which PHRED produces an utterance from its conceptual representation involves three stages. *Fetching* is the process of retrieving candidate patterns from the knowledge base of pattern-concept pairs. *Restriction* is the matching of a candidate pattern to the specifications derived from the concept and the addition of constraints to a selected pattern. *Interpretation* is the production of grammatical output from the chosen pattern. These phases are applied recursively and repeatedly until the process results in the complete specification of an utterance. At each application of the three-stage process, the fetching mechanism performs a hash based on the combination of features which specify the conceptual content and linguistic constraints which are to be satisfied by a constituent. This results in a stream of candidate patterns, which are further specified by the restriction mechanism. The restriction process uses a unification algorithm to assign feature values, but also adds ordering and agreement constraints based on the combination of pattern-specific information with general linguistic knowledge. The interpretation mechanism then applies these constraints to complete the constituent or recursively invoke the fetching mechanism.

Linguistic knowledge in PHRED, consisting of general syntactic constructs, specific idiomatic knowledge, and lexical/morphological information, is all handled by the same mechanisms. Thus, while the overall scheme of the generator is the successive refinement of the output specification, similar to the strategies used by Appelt, McDonald, and McKeown, the process is simplified by the uniform treatment of knowledge. This knowledge is, however, restricted to the tactical, or "how to say it", level of generation.

PHRED and PHRAN were used as central components of the user interface to the original version of the UNIX Consultant System (UC) (Wilensky, Arens, and Chin, 1984). This system answers questions and solves problems related to the UNIX operating system. UC passes the conceptual form of its responses, usually either questions or answers to questions, to the PHRED generator, which expresses them in the user's language. The consultant can answer questions and produce its responses in either English or Spanish, as evidenced by the examples below:

User: How do I delete a file?

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

User: Cómo se borra un archivo?

UC: **Para borrar el archivo de nombre filename del espacio de trabajo, se escribe 'rm filename'.**

User: How do I delete all the files from a directory?

UC: **Typing 'rm *' will remove all files within the current directory from the current directory.**

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

There are a number of practical advantages to PHRED as the generation component of a natural language system. Having a knowledge base shared between analyzer and generator eliminates the redundancy of having separate grammars and lexicons for input and output. It avoids possibly awkward inconsistencies caused by such a separation, and allows for interchangeable interfaces, such as the English and Spanish versions of the UC interface. It operates in real time: The Unix Consultant requires no more than nine seconds of CPU time to answer the questions above, of which only three seconds or less is generally used by the generator.

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

While PHRED affords extensibility, simplicity, and processing speed, its design incorporates a cognitive motivation as well. It diverges from the traditional computational approach to language as a symbol manipulation process, and treats it more as an associative process among knowledge structures. The phrasal approach minimizes the autonomy of the individual word, the bane of some Artificial Intelligence approaches to language. The treatment of most linguistic knowledge as declarative bears cognitive as well as practical significance. PHRED has proven to be a simple and effective program, but the difficulty in extending the program to incorporate higher-level language production components has suggested an approach in which abstract knowledge may be more easily exploited.

2.5. Dialogue, Context and Memory

At least two of the programs discussed in the previous sections, those of McKeown and of Appelt, have made extensive use of focusing constraints on the linguistic structures generated. Others, such as McDonald's (1980), have considered discourse structure and constraints. Most of the systems apply at least a simple set of heuristics to guide pronoun reference and other context-dependent phenomena.

Much of the current treatment of focus and dialogue stems from the work of Grosz (1977), who distinguished two types of focusing information: *global* and *immediate* focus. Grosz concentrated on the problem of representing information about global focus, items which remain in a speaker's center of attention throughout a discourse, as opposed to those which are transiently drawn into focus within a particular utterance. She handled issues of global focus by inventing *focus spaces*, sets of items within the speaker's knowledge base which are "opened" or brought into focus when one of the items is referred to.

Sidner (1979) added the consideration of immediate focus to the literature, concentrating primarily on anaphora and their relation to the shift of focus. Sidner specified four rules for maintaining and shifting focus, which influence the constraints on anaphora within the discourse.

Grosz's and Sidner's work was aimed at the comprehension of dialogues, but McKeown was able to extend many of their basic ideas to aid language generation within a dialogue. The pool of "relevant knowledge" alluded to earlier, used to fill out TEXT's rhetorical schemata, corresponds to Grosz's focus space. In the next stage, as the schema is filled with propositional information to be realized in the output text, the generator may have to select from a set of potential propositions. In this stage Sidner's immediate focus rules are used to filter out propositions which violate focus constraints.

Among the propositions remaining, TEXT applies a simple set of heuristics to make the choice of immediate focus. The ordering of these processes may seem awkward, but is effective in producing natural anaphoric references, essential for the multisentential explanations generated in McKeown's system.

Hobbs (1978) raises the issue of how to describe *coherence* in discourse, a question related to focus but more devoted to structural and lexical decisions than to anaphora. Hobbs' work details *local* and *global* coherence, in keeping with the divisions of Grosz. The research is geared primarily to the analysis of dialogues; however, it is directly applicable to language generation. Many of the problems covered by Hobbs' work are not addressed in any current language generation system.

Focus, coherence, anaphora, ellipsis, and other dialogue-related phenomena are all aspects of *context* in language processing. Understanding anaphora, for example, is accomplished by the interaction of knowledge about the syntactic constraints on anaphoric references with knowledge about referents which are in the global focus. Generating anaphora can likewise be viewed as the product of the interaction between syntactic knowledge and knowledge about the conceptual objects which are available for reference. As Hobbs makes clear, this interaction between conceptual knowledge about a discourse and knowledge of its syntactic structure is a widespread phenomenon. It is thus an attractive idea to consider the modeling of context in general in such a way that local and global foci, and syntactic and conceptual influences, can interact in both the understanding and generation of language in discourse.

Arens (1982) developed a "context model" for use in language understanding, primarily in resolving references. The main idea behind the model is that such understanding seems to be the result of the resolution of a variety of effects on knowledge structures "active" in memory. These effects may include biases due to syntactic constraints, preconceptions or presuppositions, or the semantic and pragmatic interpretation of the ongoing dialogue. The model proposes a "context buffer" consistent with Grosz's "focus space" idea, but meant to be more generally applicable. The context buffer is a set of concepts which are active in the current context. Activation may come from direct reference in the discourse, or may spread through "clusters" of associated concepts in long-term memory. This model is related to work in cognitive psychology (cf. Quillian, 1966; Anderson and Bower, 1973). When a new reference requires resolution, it activates possible referents in the context buffer and is matched against the most active. The Unix Consultant uses this method uniformly to resolve references dependent on contextual biases.

There are numerous difficulties with using a spreading activation model in computer systems. The obvious technical problems are: (1) simulating an inherently parallel process, (2) determining the strength of association necessary to control the spread of activation, and (3) determining *which* concepts are associated at all. The first problem detracts mostly from the efficiency of the model, but the second and third problems are fundamental. They will likely be solved only by a simulation of the knowledge acquisition process. Introspection and experimentation are both extremely limited in their ability to provide functional networks for spreading activation models.

Analysis

The practical task of incorporating knowledge about context into a language generation system is one of modeling the implicit effect of a large body of knowledge. The research discussed in this section suggests that formulating the rules required to apply contextual constraints to language involves the consideration of a tremendous quantity of knowledge, and doing so explicitly is a computationally unattractive prospect. The TEXT system demonstrates that applying these rules at several levels in the generation process improves the quality of output, but increases the complexity of the task. The key to facilitating contextual influences in generation seems to be in the representation of knowledge which can be used to simulate the effect of relevant context on the selection of linguistic structures.

Summary: A Schematic View of Some Generation Research

The following diagram presents a general view of some of the aspects of language generation discussed here, with respect in particular to some of the more recent generation systems and others presented in this chapter. The rectangles indicate rough groupings of work according to linguistic representation, emphasis, "strategic" approach, and primary test domain:

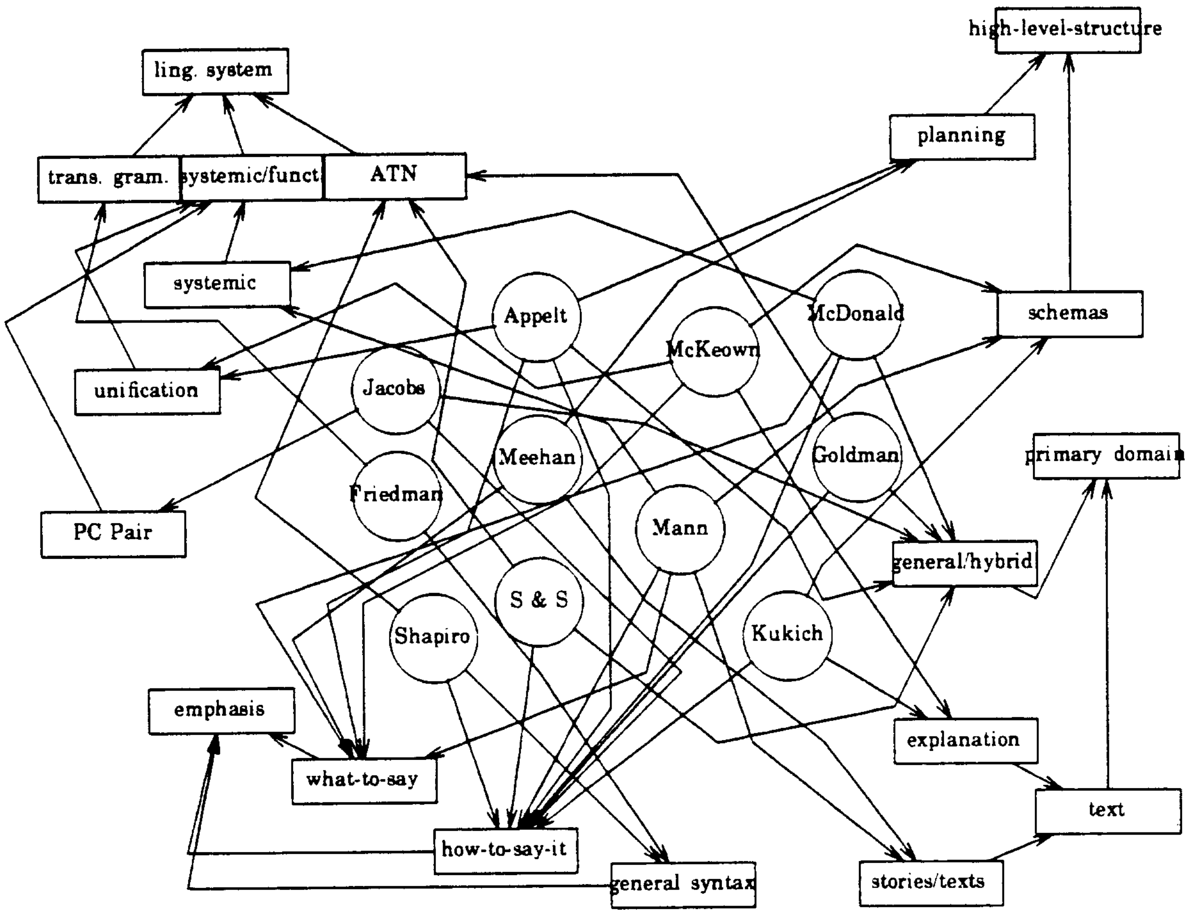

Figure 2-1.

3. A Framework for Knowledge about Language

One of the main practical problems with the language generation systems discussed in Chapter 2 is the difficulty in extending and adapting these systems. I have proposed that this difficulty is due to the narrowness in scope of the design of such programs, and that the key to extensibility and adaptability is to focus on the representation and interaction of knowledge in a system.

The research presented here has been largely driven by the successes and limitations of PHRED. The problems with knowledge representation in PHRED highlight some theoretical aspects of language processing which seem particularly relevant to the development of more powerful generation systems. Foremost among these is the need to take advantage of abstractions or generalizations in the representation of specialized knowledge. The consideration of this requirement is part of the theoretical foundation of the Ace representation and the KING generator.

This chapter presents examples of how the problem of exploiting generalizations appears in language generation, and gives an overview of the aspects of the Ace framework which address this problem.

3.1. Specialization and Generalization

Part of the justification for the development of the PHRAN and PHRED systems came from work in cognitive linguistics (cf. Chafe, 1968, 1984), founded in the consideration of actual spoken and written texts. Such research often highlights the specialized constructs prevalent in ordinary language, and tends to downplay the importance of a "core" of grammatical knowledge. Linguistic knowledge is viewed as a wide range of specialized linguistic patterns with associated knowledge about the meaning of these patterns. Related research on "sublanguages" (Harris, 1968; Kittredge and Lehrberger, 1983) formed part of the theoretical basis of Ana (Kukich, 1983). Ana was able to generate fairly sophisticated stock market reports by utilizing knowledge about the special phrases and constructs used in that domain. PHRED, similarly, produced output for the UNIX Consultant primarily by using specialized linguistic knowledge.

The use of specialized phrasal knowledge can have a substantial influence on the robustness and efficiency of a natural language system. If specialized linguistic knowledge is indeed as pervasive as Chafe would argue, a system which deals only with "core" grammatical and productive constructs will handle but a small portion of a language. A generator working within such a system would be severely limited in the range of utterances which it could produce and in its ability to produce an output appropriate to a given context. On the other hand, failing to take advantage of linguistic generalizations can introduce redundancy and possibly inefficiency into the knowledge base. Robust and efficient language processing therefore demands a balance between specialized and generalized knowledge. This balance was one of the theoretical goals for PHRAN (Wilensky and Arens, 1980), but the actual implementation of PHRAN and PHRED did not exploit abstract knowledge as effectively as possible.

The need for command of specialized constructs in natural language generation systems is not obvious, but surfaces frequently as the capacity of the systems is increased. For example, the UNIX Consultant is called upon to give responses such as the following:

> You *need* write permission on the parent directory.
> Typing 'vi <filename>' *gets* you into the 'vi' editor.
> The 'ps' command *gives* you the status of your processes.

Terms such as "write permission" and "parent directory", discussed earlier, are specialized in that they are specifically used conventionally to refer to concepts in the UNIX world. The use of expressions for *needing* write permission, *getting into* an editor, and *giving* the status of processes also were handled in PHRED using specialized patterns. But these expressions seem closely related to the constructs used in the following examples:

> You don't *have* write permission on the parent directory.
> Type ':q' to *get* out of the editor.
> 'hostname' *gives* you the name of the machine.

Treating knowledge about the constructs above entirely as specialized knowledge makes it difficult to take advantage of the similarities among them. This difficulty aggravates the problem of adding new, related knowledge to the system. For example, one can "need" write permission, "have" write permission, "get" write permission, and "give" write permission. These verbs are often used in the UNIX domain in relation to concepts having to do with access, control, and permission. The specialized meaning of "needing" write permission seems closely related to the specialized meaning of "having" or "getting" write permission. Similarly, "getting into" an editor is related to "being in" an editor and "getting out of" an editor. "Giving" names and status is related to "getting" names and status. Where such relationships exist, it should be possible to take advantage of them by encoding certain knowledge as knowledge about a class of related expressions. This eliminates the redundancy of treating all the knowledge as specialized and limits the amount of information which must be encoded for a new related expression. Therefore, it seems that there are generalizations about the use of these expressions that should be exploited by a generation system which produces them.

Generalizing about Give and Take

The ability of a generator to take advantage of generalizations in producing constructs such as the above depends on its command of what I call *high frequency, low content* verbs. These include "need", "have", "give" and "get" above. Other high frequency, low content verbs include "take", "make", "be", "do", and "put". All of these verbs may be used in a variety of contexts and take on specialized meanings according to how they are used. Their consideration here is important for two reasons: (1) A generator with a realistic command of the English language must necessarily use these verbs frequently, and (2) the interpretation of the verbs is heavily dependent on context, thus making an excellent testing ground for the representation used by the generator.

In order to determine the types of generalizations that can be exploited in representing knowledge about high frequency, low content verbs, it is helpful to consider examples from domains other than the UNIX Consultant. The following represents a small, representative set of examples of the use verbs "give" and "take" in a variety of contexts*:

(1a) Frazier gave Ali a punch/hook/jab.
(1b) Ali took a punch/hook/jab from Frazier.
(2a) John took a vacation from work.
(2b) John's boss gave him a vacation.
(3a) The doctor took (over) his colleague's patients.
(3b) The doctor's colleague gave him all his patients.
(4a) Could you give me some time?
(4b) Take a few minutes.
(5a) The operation took Frank's voice.
(5b) The operation gave him his voice back.
(6a) Humpty took a great fall/spill
(6b) ? What gave Humpty his fall/spill?
(7a) Tip gave the floor to Ted.
(7b) Ted took the floor.
(8a) Give Jesse a chance.
(8b) Jesse doesn't take chances.
(9a) Al was given command/charge of the troops.
(9b) He took charge/command.

The examples above illustrate a range of uses of the verbs "give" and "take" which would seem to suggest that many of the specialized or non-prototypical uses of the verbs are based in some common metaphorical relationship, possibly in a metaphorical concept of possession transfer. This does not, however, serve as a general explanation for the use of the verbs. There are, to give a few, problematic constructs such as the following:

(1c) Ali took a clout/sock/? from Frazier.
(2c) ? John took a vacation from his boss.
(4c) I'll take my time.
(5c) ? The operation gave him his speech back.
(5d) The operation gave him his hearing/sight back.
(5e) ? John took his hearing back from the operation.
(6c) ? Humpty took a slip.
(8c) ? Jesse took a chance from the New York primary.
(10a) We gave Reagan the benefit of the doubt.

* The indication "?" here is used to mark sentences which are awkward or incomprehensible. It is not a syntactic judgement.

(10b) ? Reagan took the benefit of the doubt.

Some of the examples above illustrate uses of "give" and "take" which are consistent with uses in the first set of examples, but are simply not generated, often because of some other way of expressing a given meaning. This may be the case with (1c), (5c), and (6c). Examples (2c), (4c), and (8c) represent constructs involving "give" and "take" which sound awkward or take on meanings which do not seem consistent with similar constructs in (2b), (4a), and (8a). Examples (5e) and (10b) seem to fail because of the implausibility of the idea that the subject could have played an agent-like role. (4c) and (8c) appear to demonstrate that a specialized construct commonly used to mean one thing is sometimes blocked from taking on another meaning. (2c) sounds awkward due to a combination of these two factors.

These examples indicate that the metaphorical view of certain concepts as possession concepts and certain events as possession transfers might provide a good general rule, but does not entirely explain the use of constructs containing "give" and "take". Often the specialized uses of "give", "take", and "have" show a consistency in the types of objects with which they are used, but retain subtle variations in their meaning when used with these objects. For example, "giving time", "taking time", and "having time" along with "giving a minute" "taking an hour", and the like, illustrate the convenient generalization that verbs describing the creation, allocation, and consumption of a resource may be used to refer to the allocation and use of time. "Take my time" in (4c), however, invokes a specialized meaning of "take" different from "take the time [to do something]". Sometimes such constructs exhibit inconsistencies as well. For example, "John has hearing" sounds awkward, while "John has good hearing" does not, presumably because the content of the former is ordinarily expressed by "John can hear" or "John is able to hear". The generalization that "have" may be used with the same class of objects as "give" would fail in this case. Applying generalizations about these verbs too broadly thus could cause a generator to produce awkward or inappropriate sentences. The problem for generation systems is to exploit the general relationships which may apply to constructs such as those involving "give" and "take" while also facilitating the representation of specialized knowledge.

The following two sections discuss two features of a knowledge representation framework designed to facilitate the use of generalizations in language production.

3.2. Views and Indirect Reference

One aspect of a knowledge representation which makes the encoding of linguistic generalizations easier is the explicit representation of referential relationships. These include relationships between linguistic structures and conceptual structures, and relationships between conceptual structures which may be used in indirectly referring to concepts. The most common such relationship is called a *view* (*cf.* Wilensky, 1984; Jacobs and Rau, 1984). A view is a relationship between two concepts which represents the fact that an instance of one concept may be expressed, or viewed, in terms of the other, without requiring that the instance be a instance also of the second concept. This notion is useful in generation, where one may indirectly or metaphorically describe a concept by describing a related concept.

The first set of examples involving "give" and "take" illustrated how the two verbs make be used in expressions similar in meaning. "Give" and "take" may often be used to describe indirectly the same event, as may the related verbs "get" and "receive". In the case of "John gave Mary a book" and "Mary took a book from John" one might argue that the event being described *is* a *giving* or a *taking* event. With "Ali took a punch from Frazier" and "Frazier gave Ali a punch", which might express the same concept, it is difficult to say whether this concept can be classified as either a *giving* or a *taking*. But there are consistencies between the two sentences; that is, the "punch" is always the object, Ali is in a metaphorical sense the recipient of the punch, and Frazier is the source. One way for a system to take advantage of this consistency is to represent as an abstract concept the class of events to which these verbs often refer, and to represent explicitly the relationship between these verbs and the abstract concept.

The abstract concept which many uses of "give" and "take" express is the *transfer-event* concept. We can hypothesize that the verb "give" describes a concept *giving*, and "take" describes a concept *taking*, and that these concepts are in turn related to the *transfer-event* concept. If we call the agent of a *taking* action the *taker*, and say that the *transfer-event* takes place among *source*, *recipient*, and *object*, we can make the following generalization:

> *Transfer-events may be viewed as taking, with the recipient playing the role of taker.*

The relationship between the *transfer-event* concept and the *taking* concept is called a *view* because it is a relationship whose application depends on context. In other words, a given event may be classified as a *transfer-event* and not as a *taking* event, but in a particular context the same event may be *viewed* as *taking*. This view is often applied in the context of producing language, as language provides a common means of expressing *taking* and not of expressing *transfer-event*. A similar view relates *transfer-event* to *giving*. By describing *giving* or *taking*, one is *indirectly* describing the *transfer-event*.

Views are a type of *structured association*. A structured association (Wilensky, in preparation) is a link between two concepts which in turn joins other concepts whose relationship depends on that link. These secondary relationships are called *role-plays*, such as that between *taker* and *recipient* above. Role-plays are *structural relationships* associated with the view or other structured association. When *transfer-event* is viewed as *taking*, the *recipient* of the *transfer-event* is viewed as *taker*. There are several types of structured associations implemented in Ace; these will be discussed in Chapters 4-6.

One application of views is in the representation of metaphorical relationships such as those described by Lakoff and Johnson (1980). These relationships often serve to enable the use of a linguistic construct to refer indirectly, or metaphorically, to a concept, much in the manner described above. In the "give" and "take" examples, we can postulate a metaphorical relationship between *actions* and *transfer-events* such as the following:

Actions may be viewed as transfer-events, with the actor playing the role of source, the object playing the role of recipient, and the action itself playing the role of object.

This metaphorical relationship may be used in the following manner: A given *action* may be viewable as a *transfer-event*, which in turn may be viewable as a *giving* or *taking* action. This enables the use of "give" and "take" to refer indirectly to a range of actions, as well as providing the *role-play* relationships which guide the linguistic structure produced. By allowing the use of linguistic structures describing *giving* or *taking* indirectly to describe other actions, a generator can then make use of these views to produce a range of constructs. The following examples conform to this general description:

(1a) Frazier gave Ali a punch.
(1b) Ali took a punch from Frazier.
(11a) John was given a kiss on the cheek by Mary.
(11b) John gave Mary a hug.
(11c) John gave Mary a massage.

The "give" and "take" constructs above may all be explained using the views presented here. For example, the concept described indirectly in (1a) and (1b) may be represented as a *punching* action, where Frazier is the *actor* and Ali is the conceptual *object* or patient. The view of *actions* as *transfer-events* relates this action to an abstract metaphorical *transfer-event* in which Frazier is the *source*, Ali the *recipient*, and the *punching* is the object. The view of *transfer-events* as *giving* relates this event in turn to a *giving* action in which Frazier is the *actor*. This enables the indirect description of the event using "give" as in (1a). A similar view between *transfer-events* and *taking* actions represents part the information necessary to produce (1b). The grammatical roles of Frazier, Ali, and "punch" in both sentences may be predicted by the *action as transfer-event* view. The use of these views in the generation process is covered in detail in Chapters 7 and 8.

Examples such as these illustrate how views may be exploited as generalizations in the production of language, particularly with respect to uses of the verbs "give" and "take". The next section discusses a second method of representing generalizations. The way in which these generalizations are used in the representation of specialized knowledge is discussed toward the end of this chapter.

3.3. Hierarchies

Hierarchies can allow for parsimony in conceptual representations by permitting knowledge common to multiple categories to be encoded at an abstract level and *inherited*. This also makes the addition of specialized knowledge in a system easier by allowing certain properties of a new knowledge structure to be implicitly represented by virtue of membership in an existing class of structures.

Inheritance has an important place in this representation. An example of a hierarchical piece of knowledge is that *giving* is a subcategory of *action* in which the *giver* plays the role of *actor**. This allows the concept of *giver*,

* Those familiar with other knowledge representation schemes should beware that the use of terms such as *actor*, *action*, and *event* here differs from their use in most other systems. KL-ONE applications

for example, to inherit knowledge about *actor*s, such as the knowledge that the *actor* of an *action* is animate.

Hierarchical associations also permit the application of abstract views to more specific knowledge structures, and thus allow for these abstract views to be used in a variety of constructs. For example, the *communication-transfer* event is a subcategory of the *transfer-event* in which the *message* plays the role of the *object* transferred. The action *telling* is a view of *communication-transfer* category. The knowledge that a *communication-transfer* has a *source* and a *recipient* is inherited from *transfer-event*. This knowledge may be used in constructing dative phrases, just as it is used with the verb "give". Thus the relationship between Mary in "John told Mary a story" and the role which Mary plays in the underlying concept utilizes the same generalized knowledge as the relationship between Mary and *recipient* in "John gave Mary a book."

Relationships among knowledge structures in this framework are themselves treated as knowledge which may be hierarchically organized. Thus a view may be a subcategory of another view. For example, the relationship between *communication-transfer-event* and *telling* may be represented as a subcategory of the relationship between *transfer-event* and *giving*. An important feature of this is that the structural relationship between *teller* and *source* derives from the relationship between *giver* and *source*. The knowledge about the correspondences between conceptual and linguistic structures is thus often represented at an abstract level in the system and inherited for particular relations. In this way, *giving*, *selling*, and *telling* share common knowledge, as do *buying* and *taking*. Knowledge about the similarities in linguistic structure among descriptions of these concepts is thus encoded parsimoniously. Related concepts can be easily added by creating each new concept as a subcategory of an existing concept.

Hierarchies can be used to organize both conceptual and linguistic knowledge. I have suggested that linguistic constructs may themselves be subcategories of other linguistic constructs, and in these cases parsimony is served by the representation of the category structure. For example, there is a general verb phrase category in Ace under which a number of types of verb phrase constructs, such as finite verb phrases, infinitive phrases, and gerund phrases, are organized. Certain knowledge which is associated with this verb phrase category may be inherited by all subcategories. For example, an infinitive phrase exhibits the same grammatical structure as a finite verb phrase in terms of the ordering of verb, indirect object, etc., though the constraints on how the infinitive phrase is used differ from those on finite verb phrases. A more complex hierarchy is evidenced in the organization of knowledge about helping verbs: Modal auxiliaries are a subcategory of auxiliaries, which are a subcategory of "helping verbs". Helping verbs include verbs such as "keep" and "get", as in "keep going" and "get stranded"; auxiliaries include "be" and "have", and modal auxiliaries include "do", "would", and "should". The hierarchical organization of these verbs allows aspects of their behavior to be inherited from their category groupings. Auxiliaries, for example, may all appear before the subject in a question form, as in "Has John gone?" Modal auxiliaries, furthermore, are always followed by a tenseless form of a verb.

---

especially tend to distinguish *actions* as being abstract concepts and *events* as instances of *actions* which have actually occurred. Here *actions* are views of *events*, and both are abstract concepts.

Hierarchies here, as in other knowledge representation frameworks, may be used to take advantage of generalizations by allowing knowledge structures to derive certain properties from more abstract concepts. This allows the use of generalizations for certain linguistic knowledge, such as the use of the dative form in "John told Mary a story" and "John gave Mary a book." This may be accomplished by representing relationships such as that between the *recipient* of the abstract *transfer-event* and the linguistic representation of the indirect object, so that this relationship may be used in many or all such dative forms. This type of representation is covered in Chapter 6.

Hierarchies and views are often alternative ways of making use of a generalization. In the case of "tell" above, the generalized knowledge which can be used to constrain the dative form derives from the representation of the *communication-transfer* as a subcategory of *transfer-event* in the conceptual hierarchy. In the case of the metaphorical uses of "give" and "take" in the previous section, the common knowledge derives from the representation of a variety of actions as views of *transfer-event*s. The same effect might be accomplished by grouping such actions under the category of *transfer-event* in the hierarchy. The main argument against doing so is that it seems counterintuitive to classify such events under a common conceptual category merely because they may be described metaphorically using the same set of verbs. Thus we maintain the distinction between a concept which may be viewed as another and a concept which is a subcategory of another.

The previous section, in describing how views are used in this representation scheme, discussed the use of metaphorical views to enable the production of constructs such as "John gave Mary a hug". This section outlined the use of hierarchies to represent general knowledge about actions and events. The generalizations described here must be selectively employed; in other words, there must be a facility for handling exceptions to generalizations. The next section deals with how specialized knowledge is used to restrict these generalizations.

3.4. Restricting the Use of Views

One problem with the use of views to expand the range of concepts to which a linguistic structure can refer is the overgeneralization problem. Most of the relationships which are captured by the notion of a view are nothing more than generalizations, and are not applicable under many circumstances. For example, the indiscriminant use of the view of *action*s as *transfer-event*s could lead to awkward constructs such as the following:

(11d) ? A kiss on the cheek was taken by John from Mary.
(11e) ? Mary took a hug from John.
(11f) ? Mary took a massage from John.
(11g) ? John gave Mary an embrace.

The *application* of a structured association is the use of that association to instantiate a new knowledge structure from an existing structure. In the case of generation, the application of views leads to the instantiation of new concepts which might or might not be useful in constructing an utterance. There are three basic means of restricting the application of general views, thus preventing the production of inappropriate constructs such as (11d-g). The first means is that there may be constraints on certain roles, and that views are blocked which would place a concept in a role where it violates a constraint. Thus, if there is a constraint on the *taker* role which dictates that the *taker* must play an agent-like role in the *taking* action or meet some other stipulation, this constraint could be used to prevent sentences such as (11d-f), where Mary does not seem to play an agent-like role.

The second means of restricting the range of constructs which may be produced by applying a view is by using *macro-associations*. A macro-association is a sequence of structured associations which joins a specific concept to other knowledge structures. For example, in the case of "giving a kiss", we can hypothesize that *kissing* is related to *giving* by a sequence of two views, i. e., *action as transfer-event* and *transfer-event as giving*, which are not applied independently. This would ordinarily prevent the use of "take" to refer to *kissing*. Unlike the constraints described above, such associations can be easily used to explain arbitrary preferences among views. This aspect of Ace will be further discussed in Chapter 6.

The third means of restricting the use of views is the fact that views are seldom "triggered" by abstract knowledge. The application of a view is generally performed only when some special knowledge is available. For example, the view of *action*s as *transfer-event*s is not used except when specific knowledge, such as the macro-association described above, links it to a particular action. Such a view is thus represented as a general structured association between *action*s and *transfer-event*s, but also represented is knowledge about the concepts to which the view is particularly appropriate. The manner in which views are selected by the generator will be detailed in Chapters 7 and 8.

3.5. Uniformity

While sections 3.2 and 3.3. focussed on representational issues which can be considered mostly in isolation from processing concerns, uniformity in the Ace framework is primarily a means of making processing easier. Using common representational tools throughout a knowledge network allows a few basic mechanisms to be used for a variety of knowledge manipulation functions.

The main goal of uniformity here is to allow for the linguistic and conceptual hierarchies to make use of the same inheritance mechanism. Conceptual attributes are inherited by conceptual structures, and linguistic attributes are inherited by linguistic structures. This type of uniformity was proposed during much of the work on KL-ONE (Brachman et. al., 1979), but few systems have actually taken advantage of inheritance for linguistic features. In the work on PHRED, it was apparent that inheritance could lead to a more parsimonious representation, but redundancy in the representation was not considered a real problem. The inability to exploit general knowledge, however, became more of an imposition as the difficulties in extending the system appeared.

A second important reason for uniformity is the objective of utilizing common mechanisms for language analysis and generation. This objective was realized in an early Ace prototype system, but proved somewhat awkward due to a variety of practical limitations. Nevertheless, this goal had a significant positive effect: The process of *mapping*, or utilizing a structured association to build one knowledge structure from another, is accomplished by a uniform mechanism which may be used in different aspects of language processing. This mechanism will be discussed in Chapters 7 and 8.

The idea behind having a uniform mapping mechanism is that the process of applying relationships such as views to produce meaning from language could exploit the same mapping apparatus as the process of applying the same relationships to produce language from meaning. Perhaps more important, however, is that the same mapping apparatus can be used repeatedly in the application of sequences of structured associations. In the current version of KING, this means primarily that the same mechanism which applies views is also used to apply associations between meaning and language. This uniformity has a substantial effect on the task of applying these associations in KING, as the process which applies them takes advantage of a common set of rules for selecting which associations to apply, regardless of whether it is producing new concepts or new linguistic structures.

Summary

This chapter has posed a problem of particular relevance to the practical issues of extensibility and adaptability in language processing. This problem, that of exploiting generalizations in the encoding of specialized knowledge, suggests a framework for knowledge representation which has been implemented in the Ace system. This chapter has laid the groundwork for the details of the system by focusing on theoretical aspects of language use, as evidenced in particular examples. Chapters 4, 5, and 6 will present the Ace representation in some detail, giving the particulars of the representation as applied to these and other examples. The design and implementation of KING will be covered in Chapters 7 and 8.

4. Knowledge Representation Fundamentals

I have suggested that the development of extensible and adaptable natural language systems depends on a knowledge representation framework within which generalizations are effectively exploited. The Ace representation (Jacobs and Rau, 1984) embodies such a framework. The theoretical goal of Ace is to apply a uniform, hierarchical knowledge representation scheme to the task of representing knowledge about language. This realization of this goal was accomplished by implementing and extending a knowledge representation called KODIAK (Wilensky, 1984) and applying this representation to the problem of linguistic representation.

This chapter presents the basic knowledge representation principles behind Ace, and presents an example of how conceptual knowledge is represented.

4.1. Basic Principles

Many knowledge representation systems, however different they appear superficially, may be shown to have the same formal expressive or inferential power. This discussion will avoid the question of formal power and center instead on the nature of the knowledge which must be expressed. The knowledge representation framework presented here is not intended to suggest a particular formal representation, but to provide a means for expressing the essential knowledge in a form suitable for encoding within a representational formalism.

The following Ace principles guide the encoding of knowledge important in the generation task:

Representation Principle 1. The Inheritance of Conceptual Relations
Concepts in memory are organized into a hierarchy of categories, in which more specific concepts inherit "features" from more general concepts. This inheritance is a representational tool which has been employed throughout the history of Artificial Intelligence (*cf.* Quillian, 1966; Roberts and Goldstein, 1977; Bobrow and Winograd, 1977; Brachman, et. al., 1979) The question of what exactly is inherited, however, can be answered in a variety of ways. Ace takes advantage of *structured inheritance,* (*cf.* Brachman, et. al., 1979) in which concepts linked to a particular structure may inherit from supercategories of that structure. For example, knowledge about the *seller* of a *selling* action may be inherited from knowledge about the *giver* of a *giving* action..

Representation Principle 2. The Proliferation of Conceptual Categories.
Individual concepts are themselves categories, and any concept about which there is particular knowledge is considered to form a category. Thus categories proliferate: Probably, there are far more conceptual categories than there are lexical items in the system. For example, it will be shown later in this chapter that it is reasonable to postulate a concept specifically for the action of paying money in exchange for merchandise, although there is no lexical item corresponding to this concept. The lexical term "pay" is associated with a more general concept, that of providing money in exchange for virtually anything. The lexical term "give" may be associated with a general *giving* concept, but giving

to charity, giving an idea, and giving a chance are distinct concepts with distinct linguistic manifestations. For example, the use of the verb "give" without object or indirect object as in "Bill gave" and "I gave at the office" is a linguistic phenomenon which appears almost exclusively when referring to charitable giving.

Representation Principle 3. Referential Relationships among Categories. There are a range of conceptual relationships important in language use which are not easily described as factual or ontological relationships. The one which is considered here is the *view* relationship, which helps to determine how concepts may be used in expressing other concepts. The concepts of *giving* and *taking* discussed in Chapter 3 are all views of a common transfer event, but the instantiation of the abstract *giving* and *taking* concepts cannot be factually inferred from the instantiation of a transfer event. For example, "John gave five dollars to charity" does not imply that a charitable organization *took* the five dollars from John. "Mary took the money from John" does not imply that John *gave* Mary the money. In many circumstances, however, the same event may be described using "give" or "take". For example, (1) "John gave Mary five dollars for the book" may imply (2) "Mary took five dollars from John for the book". Representing *giving* and *taking* as views of *transfer-event* permits the encoding of knowledge about describing *transfer-events* without requiring a given event to be classified as *giving* or *taking*. These views may thus represent the knowledge that "John took <x> from Mary" and "Mary gave John <x>" *might* be used to describe the same event. Such views will be shown to be useful in determining how linguistic structures are used to refer to events.

The next section describes the basic elements of the hierarchical framework in Ace.

4.2. Structured Associations in Ace

Ace makes use of a notation in which there are two types of entities: *objects* and *structured associations**. A structured association is a relation among two or more objects which also relates corresponding objects associated with the related objects.

Three types of structured associations, taken from the KODIAK representation, are of special importance in Ace†. The first of these, *DOMINATE*, associates a subcategory with its parent category. One can assert

(DOMINATE *chair recliner*)

to indicate that the concept of a *recliner* is a subcategory of the concept of a *chair*, or

* This term, and the idea of using general structured associations as a language processing tool, are due to Wilensky.

† These three associations, as well as many of the ideas here, have evolved during a series of seminars among the Berkeley Artificial Intelligence Research group, led by Robert Wilensky. Other participants in these discussions were: Richard Alterman, Margaret Butler, David Chin, Charley Cox, Marc Luria, Anthony Maida, James Martin, James Mayfield, Peter Norvig, Lisa Rau, and Nigel Ward.

(DOMINATE *color red*)

to specify that the concept *red* is a *color* concept. DOMINATE is thus akin to the "a-k-o" link and other similar relationships in some systems.

The second association, *INSTANTIATE*, links an individual to its parent category, for example,

(INSTANTIATE *person John1*)

means that the token *John1* designates an individual which is a member of the *person* category. Both INSTANTIATE and DOMINATE permit multiple inheritance: An individual in Ace can INSTANTIATE more than one category, and a subcategory may be DOMINATEd by more than one parent category.

The third type of structured association, *MANIFEST*, connects an object to a related category which is considered *aspectual* with respect to it; in other words, the conceptualization of the MANIFESTed object depends on its relationship with the MANIFESTing object. An Ace assertion of the form

(MANIFEST *action actor*)

indicates that the concept *actor* is aspectual to the concept of an *action*, so that an instantiation of the *actor* concept is necessarily an instance of that concept with respect to an instantiation of the *action* concept.

These three structured associations are analogous to links and slots in other similar representation systems; the motivation behind KODIAK was to preserve the ideas behind frame-based representations while clarifying the semantics of a "slot". For a comparison of KODIAK with other research, see Wilensky (forthcoming).

The term ROLE-PLAY, also taken from KODIAK, is used to indicate corresponding concepts across structured associations. For example, the assertion,

(INSTANTIATE *action action1* with (ROLE-PLAY *actor John1*))

indicates that the token *action1* represents an instance of the *action* category, and that the token *John1* "plays the role of", or corresponds to, the *actor* of the *action*.

MANIFEST, INSTANTIATE and ROLE-PLAY are thus combined to indicate a relationship among concepts that is underexploited in frame-based representations. The assertion above represents the knowledge that *John1* fills the *actor* "slot" of the action concept; however, this slot is itself a concept which may be related to others in the conceptual hierarchy. For example, *actor*s of *action*s are *participant*s of *event*s. Thus knowledge about a "slot", or aspectual, is not confined to its relationship to a particular concept.

Conceptual relationships are often best presented in diagram form, and throughout this thesis diagrams will be used to illustrate the Ace hierarchy. The following illustration will be used to represent DOMINATE relations:

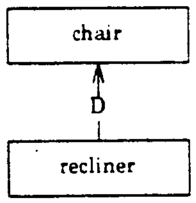

Figure 4-1.

In the above diagram, the "D" label is used to indicate the DOMINATE relation between *chair* and *recliner*.

The INSTANTIATE relation will be shown in a similar fashion to the DOMINATE relation, as in the following diagram:

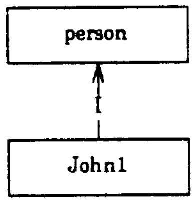

Figure 4-2.

The "I" label is used to indicate the INSTANTIATE relation between *John1* and *person*.

The MANIFEST relation is illustrated as in the following diagram:

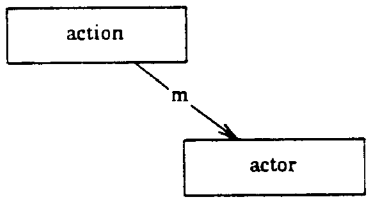

Figure 4-3.

The label "m" above is used to indicate the MANIFEST relation between *action* and *actor*. In general, DOMINATE and INSTANTIATE links are drawn vertically, and MANIFEST links are drawn diagonally.

The ROLE-PLAY relation, which may be used in combination with all structured associations in Ace, is shown as follows:

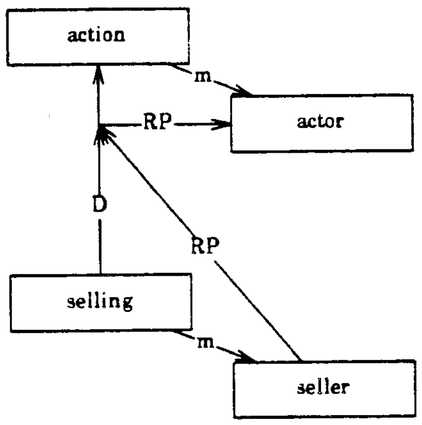

Figure 4-4.

The ROLE-PLAY link, labeled "RP", originates at the *seller* concept and joins this concept with the *actor* concept. This ROLE-PLAY is associated with the DOMINATE relation between *action* and *selling*, indicating that any concept playing the role of *seller* in a *selling* action is also playing the role of *actor*. A shorthand way of drawing the same concept, in which the ROLE-PLAY relation is implicit, is the following:

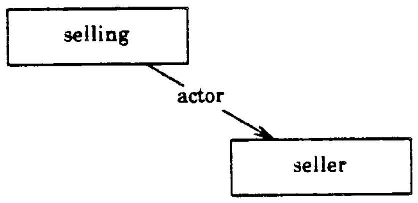

Figure 4-5.

The notation above will be used in most of the diagrams which follow. While it is more concise that the explicit ROLE-PLAY in figure 4-4, the reader must infer which structured associations have ROLE-PLAYs associated with them.

This section has presented the basic structured associations used in the Ace representation. The next section describes how these associations are used to represent knowledge about a particular complex event.

4.3. The Commercial Transaction Example

This section will consider in some detail an example of conceptual representation within the Ace framework which relates to the problem of exploiting generalizations in language processing. Specifically, we address the question of representing knowledge about *buying* and *selling*. I have suggested that it is useful to generalize about linguistic constructs such as the dative form in "John gave Mary a dollar", "John told Mary a story", and "John sold Mary a book", and that these generalizations often depend on the representation of the concepts being expressed. Consider the concept of the *commercial transaction* (Fillmore, 1977). The *commercial-transaction* represents an event in which a merchandise object is exchanged for legal tender. There is a great deal of knowledge which may be related to this event, such as the relationship between buyer and seller, the occupation of buyer and seller, the location of the transaction, and the like. Most essential, however, is the knowledge about what is being exchanged in the transaction. The representation of this knowledge and its relation to the *commercial-transaction* concept can be especially useful in the encoding of knowledge about expressing the *commercial-transaction* concept.

The *commercial-transaction* concept is not easily classified. Certainly it is an event, but the knowledge about events does not help to represent the important concepts associated with the *commercial-transaction*. Events sometimes have participants, and often seem to have objects which undergo changes of state as a result of the event. But the particular roles of participants and objects in the *commercial-transaction* are not easily specified without further decomposing the event. The *commercial-transaction* is a complex event, composed of at least two simpler events. We can assert the following:

```
(DOMINATE event complex-event)
(DOMINATE event simple-event)
(MANIFEST event participant)
(MANIFEST simple-event object)
```

These assertions capture the knowledge that *simple-events* and *complex-events* are subcategories of the general concept of an *event*, that *events* have *participants*, and that *simple-events* have *objects*. *Events* may have many *participants* and *objects*; the *simple-event* category represents the class of *events* which affect a single *object*. This is similar to the *state-change* concept in Conceptual Dependency (Schank, 1975).

The *commercial-transaction*, then, is a type of *complex-event*, which is composed of two *simple-events*, as follows:

```
(MANIFEST complex-event sub-event)

(DOMINATE complex-event commercial-transaction
    with (ROLE-PLAY sub-event ct-merchandise-transfer)
    with (ROLE-PLAY sub-event ct-tender-transfer)
    with (ROLE-PLAY participant merchant)
    with (ROLE-PLAY participant customer))
```

The above assertions capture the information that the *commercial-transaction* is made up of a *ct-merchandise-transfer* and a *ct-tender-transfer*. These two *simple-events* are both subcategories of the class of *transfer-events*:

```
(DOMINATE simple-event transfer-event)
    with (ROLE-PLAY participant source)
    with (ROLE-PLAY participant recipient))
```

```
(DOMINATE transfer-event ct-merchandise-transfer
     with (ROLE-PLAY object merchandise)
     with (ROLE-PLAY source merchant)
     with (ROLE-PLAY recipient customer))

(DOMINATE transfer-event ct-tender-transfer
     with (ROLE-PLAY object tender)
     with (ROLE-PLAY source customer)
     with (ROLE-PLAY recipient merchant))
```

These assertions indicate that the *ct-merchandise-transfer* component of the *commercial-transaction* consists of a transfer of *merchandise* from *merchant* to *customer*, and that the *ct-tender-transfer* component consists of a transfer of *tender* from *customer* to *merchant*. The events *ct-merchandise-transfer* and *ct-tender-transfer* are aspectuals of the *commercial-transaction*; that is, they are concepts which cannot be instantiated without instantiating a *commercial-transaction* concept. There may be other concepts intimately related to the *commercial-transaction* concept, some of which will be considered here; the knowledge about *ct-merchandise-transfer* and *ct-tender-transfer* is most critical in describing the *commercial-transaction* event.

The relation of the *commercial-transaction* event to its components is illustrated by the following diagram:

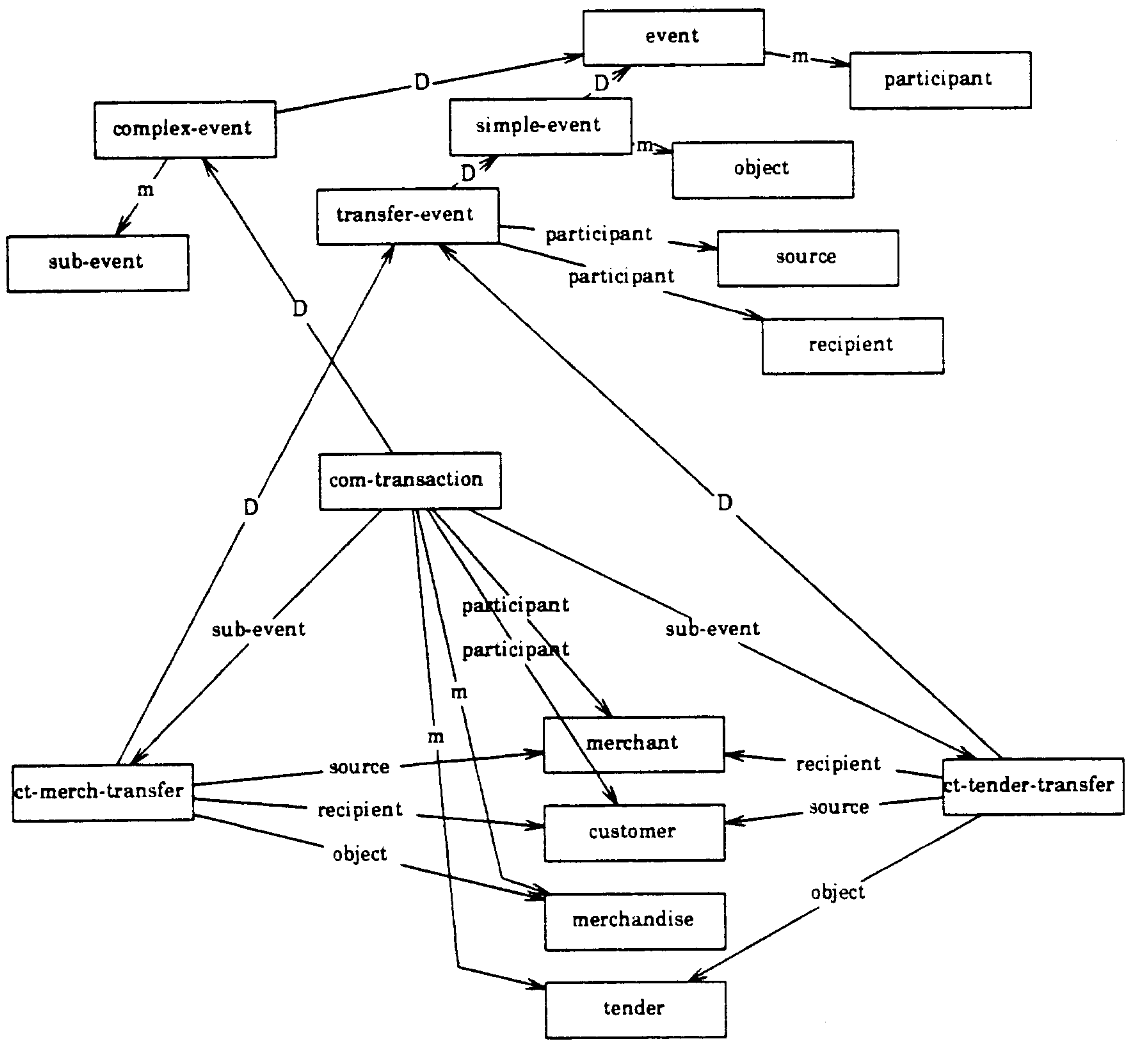

Figure 4-6.

The above diagram illustrates the knowledge that the *commercial-transaction* is a *complex-event* which consists of a transfer of merchandise and a transfer of tender. The *merchant* receives the *tender* from the *customer*, and the *customer* receives the *merchandise* from the merchant. Concepts such as *merchant*, *customer*, *merchandise*, and *tender* are aspectuals of the *commercial-transaction*; that is, they are specific concepts whose meaning is undetachable from the *commercial-transaction* event. However, much of the knowledge about these concepts, such as the *recipient* and *source* roles, is inherited from other concepts. As in other frame-like systems (Roberts and Goldstein, 1977; Bobrow and Winograd, 1977), this organization allows roles of a concept to be inherited in this manner. The ROLE-PLAY relationship in Ace, however, permits more than this simple form of inheritance: It allows for the semantics of aspectuals to be defined in terms of other aspectuals. For example, the meaning of the *merchant* aspectual of the *commercial-transaction* here is represented in part by the ROLE-PLAY relation which links this aspectual to the *source* of the *ct-merchandise-transfer* and that which links it to the *recipient* of the *ct-tender-transfer*.

The assertions above form an important core of knowledge about *commercial-transactions*. We might also want to add quite a bit of other knowledge which links together the elements of a *commercial-transaction* in other ways; for example, that the transfer of merchandise is part of a social contract for the transfer of tender. We can represent the gist of the *obligation* relationships of the *commercial-transaction* via the following assertions:

(DOMINATE *state obligation*)
(MANIFEST *obligation obligator*)
(MANIFEST *obligation obliged-event*)

(DOMINATE *obligation mt-obligation*
with (ROLE-PLAY *obligator merchant*)
with (ROLE-PLAY *obliged-event ct-merchandise-transfer*))

(DOMINATE *obligation tt-obligation*
with (ROLE-PLAY *obligator customer*)
with (ROLE-PLAY *obliged-event ct-tender-transfer*))

These assertions represent knowledge about two *obligation* concepts. *mt-obligation* is the obligation of the *merchant* to transfer the *merchandise*, and *tt-obligation* is the obligation of the *customer* to transfer the *tender*. These obligations are elements of the *commercial-transaction*, as in the following diagram:

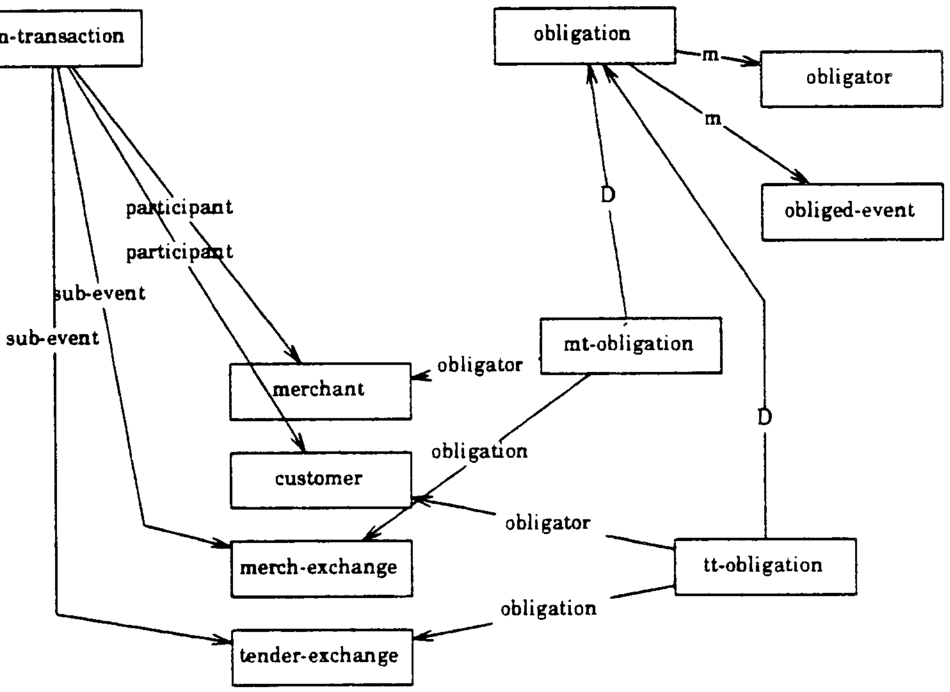

Figure 4-7.

With the *obligation* concepts represented above involving the transfer of merchandise and transfer of tender components of the *commercial-transaction*, we have encoded much of the inviolable knowledge about such events. This knowledge is important in the way language is used to describe such events. For example, it will be shown in Chapter 6 that the knowledge that *merchandise* and *tender* play *object* roles is linked to knowledge about transitive verb forms, so that phrases such as "bought a book" and "paid five dollars" conform to a general rule.

The next section discusses how concepts such as *buying* and *selling*, used to refer to the *commercial-transaction* concept, are represented in Ace.

4.4. Actions as VIEWs of Events

I have suggested that verbs such as "give" and "take" refer to the concepts *giving* and *taking*, and thus refer indirectly to the general *transfer-event* concept. The motivation for this analysis is to facilitate the representation of knowledge about the roles which *giver* and *taker* play, thereby enabling "John gave Mary a book" and "Mary took a book from John" to describe indirectly the same event, as "Ali gave Frazier a punch", and "Frazier took a punch from Ali" may indirectly describe the same event.

The *commercial-transaction* event is generally described using the verbs "buy", "sell", and "pay". "Sell" and "pay" behave similarly to the verb "give"; "buy", behaves more like "take". For example, "John sold Mary a book", and "Mary paid five dollars for the book" both use the dative form, and "John bought the book from Mary" exhibits a structure identical to "John took the book from Mary". The representation of the concepts *buying* and *selling* in Ace relates these concepts to *giving* and *taking* so that knowledge about expressing *giving* and *taking* may be used also for *buying* and *selling*.

As the discussion in Chapter 3 proposed, the concepts *giving* and *taking* in Ace are related to the *transfer-event* concept by a structured association called a *VIEW*. VIEWs are used to represent knowledge about concepts which may be used in expressing other concepts. The following Ace assertions represent the basic knowledge about *giving* and *taking*:

```
(DOMINATE action giving
      with (ROLE-PLAY actor giver))

(DOMINATE action taking
      with (ROLE-PLAY actor taker))

view1: (VIEW transfer-event giving
             with (ROLE-PLAY source giver))

view2: (VIEW transfer-event taking
             with (ROLE-PLAY recipient taker))
```

This encodes the knowledge that *giving* and *taking* are both *actions*, and that both are VIEWs of the *transfer-event* concept. When the *transfer-event* is VIEWed as *giving*, the *source* of the *transfer-event* plays the role of *giver*, which plays the role of *actor*. The labels *view1* and *view2* are used to refer to the VIEW associations, which are treated as concepts in the system. These associations are presented in graphic form below:

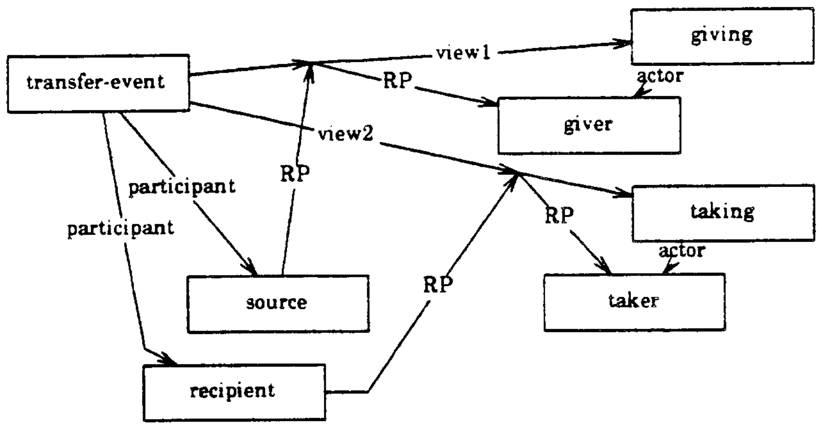

Figure 4-8.

This diagram represents the knowledge used to relate the concepts *giving* and *taking* to the concept of the *transfer-event.* The relationships between this knowledge and linguistic structures are described in Chapter 6, and the manner in which this information is used in generation is discussed in Chapters 7 and 8. The VIEWs represented here may be used in describing *transfer-events*, to other concepts which may be VIEWed as *transfer-events*, and to more specific concepts which are DOMINATED by *transfer-event.* For example, the analysis which follows will show how these VIEWs help to relate *buying* and *selling* to the *commercial-transaction* event.

The *ct-merchandise-transfer* component of the *commercial-transaction* is DOMINATEd by *transfer-event.* The *ct-merchandise-transfer* may be VIEWed as a *buying* or *selling* action, as is represented in the following assertions:

```
(DOMINATE giving selling
     with (ROLE-PLAY actor seller))
(DOMINATE taking buying
     with (ROLE-PLAY actor buyer))


view3: (VIEW ct-merchandise-transfer selling
            with (ROLE-PLAY merchant seller))

view4: (VIEW ct-merchandise-transfer buying
            with (ROLE-PLAY customer buyer))
```

These assertions indicate that *buying* and *selling* are ways of VIEWing the *ct-merchandise-transfer* event. This VIEW is used to indicate the correspondences between the concepts playing the *actor* role in *buying* and *selling*, and the *participants* of the *commercial-transaction*. These may be illustrated by the following diagram:

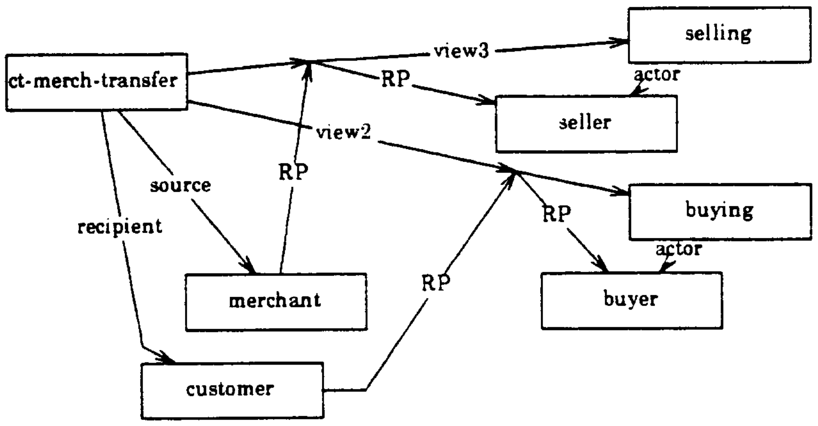

Figure 4-9.

VIEWs such as *view3* and *view4* above are concepts in Ace, and thus structured associations such as that between the *selling* action and the *ct-merchandise-transfer* are elements in the conceptual hierarchy. Treating VIEWs as concepts allows this association, *view3*, to be related to the association *view1* between the *giving* action and the concept of a *transfer-event*. Specifically, we can assert the following:

(DOMINATE *view1 view3*)
(DOMINATE *view2 view4*)

The structured association *view3* is thus treated as a subcategory of the association *view1*, and *view4* as a subcategory of *view2*, as shown in the following diagram:

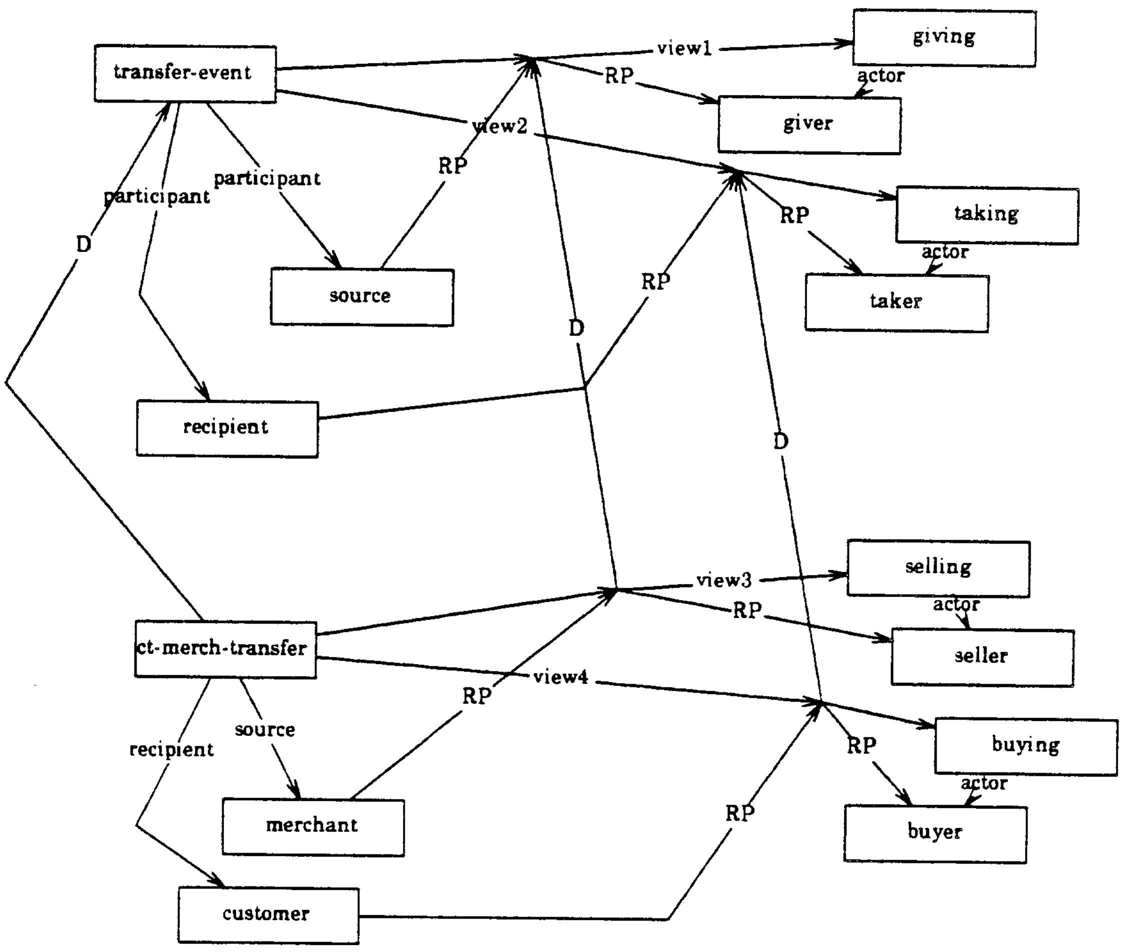

Figure 4-10.

This example demonstrates on a small scale how the hierarchical arrangement of VIEWs is used in the encoding of structured associations. Structured associations such as *view1* between *transfer-event* and *giving* DOMINATE other more specific relations, such as *view3*. Note that this makes the explicit representation of ROLE-PLAY relations for *view3* unnecessary, as the relationship between *merchant* and *seller* in *view3* is specified by the relationship between *source* and *giver* in *view1*.

The representation of the *selling* concept is a simple example of how Ace encodes abstractions which may be used in language processing. The abstraction here is the relationship between a general category *giving* and a general category *transfer-event*. There are two ways in which this abstraction may be used, both to be covered in future chapters: (1) A more specific association may be represented as a subcategory of the abstract association. This is the case in the *selling* example presented here. In this case, knowledge about the abstract association may be used in applying the specific association, thus knowledge about expressing an abstract concept may be used in expressing a more specific concept. This allows much of the same knowledge to be used for phrases involving "giving" and "selling". (2) A concept which is associated by another VIEW with the abstract concept may then also be expressed using the abstract VIEW. This is the case with expressions such as "give a punch", which takes advantage of the abstract *action as transfer-event* view in combination with the *transfer-event as giving* VIEW. These examples will be further considered in Chapter 6 and in the processing chapters.

Summary

This chapter has illustrated the use of the Ace representation in the representation of conceptual knowledge. The *commercial-transaction* example demonstrates how the Ace tools may be used to exploit abstract knowledge about *transfer-events* in encoding knowledge about *buying* and *selling*. The most important advantages of Ace, still to be considered, are those which allow for the association of knowledge about language with conceptual knowledge, and thus allow for the selection of linguistic structures from their associated concepts. The next two chapters discuss the application of the Ace framework to the representation of linguistic knowledge and to the representation of the knowledge which associates linguistic and conceptual structures.

5. Linguistic Representation

The previous chapter presented the fundamentals of the Ace representation, and showed how this representation is used in the hierarchical encoding of conceptual knowledge. The same representational framework is used to encode linguistic knowledge in Ace. In linguistic representation, as in conceptual representation, the utility of the Ace framework depends on the hierarchical representation of abstract and specialized knowledge. This chapter describes how linguistic knowledge is encoded in the Ace hierarchy.

5.1. Principles of Linguistic Representation

One of the problems with many of the linguistic formalisms described in Chapter 2 is the lack of an efficient organization of linguistic knowledge. The organization of linguistic information in a system should allow for the parsimonious distribution of knowledge. Redundancy in representation is generally a sign that adapting and extending a knowledge base will be difficult. This section clarifies some of the problems with redundancy in linguistic representation and describes how the hierarchical organization of linguistic knowledge in Ace addresses this issue.

For its basic linguistic structures, Ace adopts the fundamental linguistic notation common to *feature systems* such as unification grammar (Kay, 1984) lexical functional grammar (Kaplan and Bresnan, 1983), and pattern-concept pairs (Wilensky and Arens, 1980) The advantages of this notation for generation systems have been touched on in Chapter 2. Most importantly, the notation allows for the specification of grammatical knowledge in a form which may be used for either language analysis or language production. It assumes the ability to produce grammatical constructions directly from their conceptual form, without independent syntactic derivation.

The term *feature* is used here to designate a linguistic property consisting of an *attribute* and a *value*. Attribute names are written in capital letters in this notation. Sets of features which represent linguistic structures are referred to as *templates*, and are presented within square brackets. This corresponds to what has often been called a "pattern" in spite of the fact that it contains linguistic information which may or may not include a pattern. *Patterns* here designate only those structures which specify an ordering of the *constituents*, or structural elements, contained in a template. A constituent of a template is a component of the template which represents part of the surface form of an utterance. A set of constituents which together form a linguistic structure but whose surface ordering is not specified will be called a *relation*. The term *structural description* will be used to refer to the patterns and relations specified within a template.

Templates in this section are presented using the *feature notation*, which denotes templates by sets of features within square brackets. This notation is used to illustrate some templates which are encoded in Ace. The internal representation of these templates is achieved using the same framework as the conceptual representation discussed in Chapter 4. The following is an example of a simple template:

```
[
ROOT = have
TENSE = present
PERSON = third
NUMBER = singular
VOICE = active
LEX = "has"
]
```

The above template contains a set of features, such as "TENSE = present". In each feature, the attribute is given first, followed by an "=", followed by the value of the feature. This template can be used by the generator to produce the appropriate form of the verb "to have" to meet a set of constraints which matches the feature set.

A more complex template may contain more information, including features which themselves contain templates and surface ordering information. For example, the following template specifies the general form of prepositional phrases:

```
[
PATTERN = ( PREP PREP-OBJ )
P-O-S = prep-phrase
PREP-OBJ = [
                  P-O-S = NP
                  CASE = objective
           ]
PREP = [
                P-O-S = prep
       ]
]
```

In this template, the PATTERN feature indicates that the linguistic realization of the template requires a PREP constituent followed by a PREP-OBJ constituent. The PREP-OBJ constituent must be a noun-phrase in the objective case, and the PREP constituent must be a preposition. The P-O-S attribute, which stands for "pattern-of-speech", is used here to designate a class of templates to which the given template belongs. This corresponds roughly to the CAT attribute, or "category", generally used in unification grammar; the term "pattern-of-speech" used in the pattern-concept pair representation is employed here to emphasize that the P-O-S attribute may designate classes not usually recognized as syntactic categories, and that it is not necessary in all templates.

Some Problems with Unification Grammar

As an example of the difficulty in achieving parsimony in a linguistic representation of this type, consider the task of representing knowledge about passive sentences. In some implementations of unification grammar a sentential template such as a passive sentence is represented using the PATTERN feature to designate the structure of the passive, for example*:

```
[
PATTERN = ( ... VERB-PART ... BY-ADJUNCT )
P-O-S = sentence
VOICE = passive
VERB-PART = [
                   P-O-S = verb
                   VOICE = passive
            ]
BY-ADJUNCT = [
                   P-O-S = prep-phrase
                   PATTERN = ( PREP-BY PREP-OBJ )
                   PREP-OBJ = <agent>
                   PREP-BY = [
                                  LEX = "by"
                                  P-O-S = preposition
                             ]
                   PREP-OBJ = [
                                  P-O-S = NP
                                  CASE = objective
                              ]
             ]
]
```

The "..." in the pattern above indicates that other constituents may appear between the VERB-PART constituent and the BY-ADJUNCT constituent without affecting the use of this template. The feature "PREP-OBJ = <agent>" indicates the correspondence of the object of the preposition "by" to the semantic "agent" role. This template adequately specifies the relationship between the passive verb and the prepositional phrase which indicates the agent of the corresponding action.

There are, however, several technical problems with this template. For example:

(1) The relationship between this template and that of the passive sentence which does not have a BY-ADJUNCT is only implicit. This seems to promote redundancy: For example, the voice of a sentence is always the voice of the verb part, or complete verb, of the sentence; it seems redundant for this information to be repeated in all sentence patterns. Also, the semantic knowledge associated with the passive structure, such as semantic role of the underlying subject, must be duplicated for each passive pattern.

* This example is a slightly modified version of one presented on p. 332 of Winograd (1983). Other grammars include a different treatment of sentence level templates with the same information, such as that on p. 216 of McKeown (1982), and present slightly different problems.

(2) It is unclear that the specification of the structure of the adjunct prepositional phrase here is necessary, as it is a grammatical phrase in any case if it has the "P-O-S = prep-phrase" feature. Ideally, the implementation of the grammar should allow for this information to be omitted. It is sufficient to know that the adjunct is a prepositional phrase with preposition "by".

(3) If there is special knowledge about the relationship between a passive verb and the BY-ADJUNCT, it does not appear to be fundamentally different from the relationship between other verb structures and BY-ADJUNCTs. For example, the phrase "the invasion of Grenada by the US" seems to embody the same passive-adjunct relationship as the sentence "Grenada was invaded by the US." In a system where it is important to associate linguistic structure with the appropriate meaning, the similarity of the BY-ADJUNCTs in these examples can be of substantial aid.

(4) In unification grammar, this representation of the passive sentence template requires that the selection of the passive without agent be made only when an agent is not present. This precludes the option of simply not expressing the agent during this selection. McKeown's grammar eliminates this potential difficulty by assuming that the agent (the "protagonist" in her system) is always present. In PHRED, the grammar included a separate "ordering pattern" which included the *by-adjunct*, but it was designated as being optional. This was awkward because an optional adjunct had to be specified as a constituent of every pattern in which it could appear.

(5) The PATTERN feature used in this template must necessarily interact and overlap with other patterns in order for the generator to form a complete sentence: In this example, the PATTERN of the passive template must be combined with other sentence and verb phrase patterns to form the structure of a complete sentence. This results from the use of the "..." in the pattern to allow for intervening phrases. The PATTERN feature thus specifies only partially the structure of the sentence. It seems also that this structure must be specified for other prepositional phrases used in verb phrases in the grammar. Thus the reason for the existence of this template seems to be not to specify a surface structure but to provide *some* template to which to attach the special meaning of the BY-ADJUNCT. It would seem that this could be accomplished without using a special pattern.

These are not inherent problems with unification grammar or other feature systems, but are difficulties in using such systems to encode basic linguistic knowledge without certain awkward and redundant templates. Such awkwardness and redundancy may be avoided by applying some simple knowledge representation techniques to the encoding of linguistic templates.

The Ace Linguistic Principles

The Ace linguistic hierarchy takes advantage of the following principles to alleviate the difficulties described above:

Representation Principle 4. The Inheritance of Linguistic Features. Sets of features which are common to a certain class of templates need not be specified independently for each template in the class. Thus, if there is a set of features shared among passive sentences, or among prepositional phrases, these features belong by default to any template in the class. This would eliminate the need for fully specifying the structure of the adjunct phrase in the passive template above, as most of its features are common to all prepositional phrases.

Representation Principle 5. The Proliferation of Linguistic Categories. In order to take advantage of the inheritance of features, there must be a wide range of classes of templates which share sets of features. Often these categories depart from the traditional syntactic classifications. Requiring that a template be a member of a unique category in the case of a gerund or nominalization can prove difficult, as these may inherit certain attributes from verbs and certain attributes from nouns. Thus any template may inherit features, including structural descriptions, from multiple categories. Categories are arranged hierarchically, so each category inherits from all its ancestors in the hierarchy.

Representation Principle 6. Distinguishing Grammatical Relations from Grammatical Patterns. A great deal of linguistic information seems associated with structural relationships between linguistic constituents which are dependent neither on their order in a surface structure nor on the precise nature of the structure in which they appear. For example, the relation between subject and verb retains its linguistic features regardless of how the subject and verb appear in any surface structure: The agreement between subject and verb in "John was given the book by Mary" is the same as in "Was John given the book by Mary?" as is the conceptual *recipient* role which John plays. Such information does not pertain to a particular surface structure, but to any surface structure in which a noun phrase and verb are in the *subject-verb* relation. In "John kissed Mary on the cheek" and "The kiss on the cheek pleased Mary", the role of "on the cheek" as it relates to "kiss" is independent of whether the prepositional phrase is part of a verb phrase or noun phrase. In general, structural linguistic relationships are not limited to those which are directly linked to constituent order, and thus a more general facility than the pattern feature is required to represent these relationships.

Principle 4 above, the Inheritance of Linguistic Features, suggests a means for the more perspicuous encoding of many of the features in the passive sentence template. The knowledge that the voice of a sentence is the voice of its verb part is knowledge about all sentences which need not be explicitly represented in each template. The structure of the *by-adjunct* also need not be explicitly encoded, as it inherits its structure from the prepositional phrase.

Principle 5, the Proliferation of Linguistic Categories, suggests that groups of templates and other linguistic structures with common features be organized into categories wherever necessary. Principle 4 proposes a hierarchical organization, and Principle 5 suggests that this organization can best be exploited by allowing the organizational nodes in the hierarchy to proliferate. Thus the various templates in the system can be hierarchically organized into classes of templates. The proliferation of these classes reduces the amount of information that has to be encoded for a given template by allowing the template to inherit features from a general category. For example, the traditional verb phrase category inherits its structure from a more general category which includes gerund phrases and infinitives as well. In fact, it is even feasible to have a "by-adjunct" category, whose meaning is generally determined by its relation to the verb part of a sentence or clause. The role of such categories will be pursued further in subsequent examples.

Principle 6, Distinguishing Grammatical Relations from Grammatical Patterns, is employed in passive sentences to eliminate the need for a special pattern to handle the function of the *by-adjunct* in these sentences. In the Ace linguistic hierarchy, adjuncts within verb phrases are joined to the verb part of the sentence by a *verb-adjunct* relation, which includes the relation between the passive verb and its *by-adjunct* as well as many relations between verbs and adverbial prepositional phrases. The *verb-adjunct* relation furthermore shares a common supercategory with the relation between the nominal and postnominal modifier constituents of complex noun phrases. Thus "the kiss on the cheek" and "the game yesterday" exploit a common relation with "John kissed Mary on the cheek" and "The game was played yesterday". For this linguistic class, the choice of what adjuncts will appear in a sentence is not a structural choice but a decision of whether to express a particular relation.

The idea of the grammatical relation is one which has been part of a number of linguistic theories (*cf.* Kaplan and Bresnan, 1983). However, the treatment of relations as categories in the Ace hierarchy facilitates certain uses of relations which would be difficult to implement otherwise. For example, the relation between a modal auxiliary and a main verb constrains the verb to be in non-finite form. The way in which this relation may be realized in a verb phrase, however, is knowledge about a more general relation. This example will be considered further in section 5.3.

These three principles suggest modifications to the way basic linguistic representation has been handled in language generation systems. These modifications are not fundamental changes to the common representational framework of the unification-based systems, but are extensions to the framework which facilitate a more parsimonious and effective encoding of linguistic knowledge. The first two principles involve minor changes in the way a grammar is implemented, in order to minimize redundancy. The third requires the addition of a new type of linguistic structure, the grammatical relation, to avoid one of the awkward uses of linguistic patterns.

The next section describes how these principles are applied to linguistic representation in Ace and gives some examples of the encoding of simple grammatical information in the Ace hierarchy.

5.2. Basic Grammatical Knowledge in Ace

The principles outlined above motivate a linguistic representation in which knowledge is dispersed throughout the hierarchy, with a greater number of structures, each containing more limited information. Linguistic knowledge in Ace is organized into a hierarchy which incorporates this type of organization. This modularity leads to a simple set of basic linguistic templates, with little redundancy and relative ease of extension. Active and passive constructs are handled using the same set of patterns. To see how this is done, we will consider a basic sentence template in feature notation, and then show how it may be represented parsimoniously in Ace:

```
[
PATTERN = ( NP VP )
P-O-S = Basic-S
PARENT-P-O-S = sentence
NP = [ P-O-S = noun-phrase ]
VP = [ P-O-S = finite-verb-phrase ]
RELATION = [
                PARENT-RELATION = subject-predicate
                SUBJ = NP
                PRED = VP
             ]
]
```

This template covers NP-VP constructs regardless of voice, and thus can be used for passive with agent, passive without agent, or active sentences. The RELATION feature above is used to indicate that the NP of the basic sentence PATTERN plays the role of subject in relation to the VP of the PATTERN. The RELATION feature, like the PATTERN feature, is somewhat specialized, used to indicate structural relationships among pattern constituents. The RELATION feature occurs in most templates which have a PATTERN feature, and more complex patterns may be associated with more than one RELATION.

The features PARENT-P-O-S and PARENT-RELATION are used to show the membership of linguistic structures in categories. Since each template is itself a category, its name as well as the name of the group to which it belongs must be specified. Its name is specified by the P-O-S feature. No name is given for the RELATION feature, which is asserted to be a subclass of the *subject-predicate* relation.

The *subject-predicate* relation in the template above represents a linguistic category which incorporates much of the information about basic sentences. For example, the knowledge that the verb phrase agrees with the noun phrase in the sentence is knowledge about the *subject-predicate* relation, and knowledge about the roles of the concepts referred to by the NP and VP is associated with this relation. Thus this knowledge may be applied to the other forms in which the *subject-predicate* relation may appear, such as the WH- and inverted forms. The use of this knowledge in generation is detailed in Chapter 8.

As the Basic-S template belongs to the sentence category, it may inherit features of that category and may be used where linguistic constraints require a sentence. The knowledge that the tense and voice of the Basic-S correspond to the tense and voice of the VP constituent is inherited from the sentence category.

The reason for allowing levels of sentence categorization is demonstrated by the ways in which embedded sentences appear. In some cases, any sentence pattern may appear as part of another sentence pattern. For example, the following sentences are acceptable: "To delete foo, type 'rm foo'" and "To delete foo, can I type 'rm foo'?" In other cases, such as complements, a Basic-S is required: "I thought that can I type 'rm foo' ?" is not acceptable. The knowledge about preposed constituent patterns thus specifies that the preposed elements may be followed by any sentence pattern, while the pattern for "that" complements specifies the more specific Basic-S pattern. Having a hierarchical representation of these sentence categories seems natural, but is precluded by most implementations of feature systems.

I have proposed that uniformity of representation, while not motivated by any particular representational problem, is a useful goal to be applied to linguistic representation. This goal may be stated by the following principle:

> *Representation Principle 7. Uniformity of representation.*
> Linguistic knowledge *is* knowledge, and thus can be encoded using the same representational framework as conceptual knowledge. The same structured associations used in the conceptual hierarchy can be used in the linguistic hierarchy*. Having such uniformity of representation has the practical value of facilitating the interaction of conceptual and linguistic structures.

The sentence template above can be easily encoded using the structured associations of Ace described in Chapter 4. Features of linguistic structures are generally MANIFESTed by those structures, although certain features related to inheritance, such as the "p-o-s" attribute, have a different status: They serve to specify which linguistic categories DOMINATE a given template. Within the Ace hierarchy, categories such as *sentence* and *noun-phrase* are types of *linguistic-templates*. These templates have numerous aspectuals, among them *l-pattern* and *l-relation*, corresponding to the PATTERN and RELATION attributes in the above template. These aspectuals are of type *pattern* and *relation*, respectively. This may be stated in the following Ace assertions:

(DOMINATE *linguistic-template sentence*)

(MANIFEST *linguistic-template l-pattern*)
(MANIFEST *linguistic-template l-relation*)
(DOMINATE *relation l-relation*)
(DOMINATE *pattern l-pattern*)

Relations are made up of aspectuals *rel-part*, for "relation part", and patterns are made up of aspectuals *const*, for "constituent". Constituents must be ordered, but the order of relation parts is not important. The particular relation *subject-predicate* has a *subj* aspectual and a *pred* aspectual, both playing the role of *rel-part*:

* The use of a knowledge representation language to encode linguistic knowledge has been practiced with KL-ONE (Brachman et. al, 1979; Sondheimer, Weischedel and Bobrow, 1984) and its successors (cf. Brachman, Fikes, and Levesque, 1983), also favoring uniformity of representation. Such systems have not been used, to my knowledge, to encode associations between conceptual and linguistic knowledge.

```
(MANIFEST relation rel-part)
(MANIFEST pattern const)

(DOMINATE relation subject-predicate
     with (ROLE-PLAY rel-part subj)
     with (ROLE-PLAY rel-part pred))
```

The *Basic-S* template can be encoded using the following assertions:

```
(DOMINATE sentence Basic-S
     with (ROLE-PLAY l-relation basic-s-relation)
     with (ROLE-PLAY l-pattern basic-s-pattern))

(DOMINATE pattern basic-s-pattern
     with (ROLE-PLAY const bs-np)
     with (ROLE-PLAY const bs-vp))

(DOMINATE subject-predicate basic-s-relation
     with (ROLE-PLAY subj bs-np)
     with (ROLE-PLAY pred bs-vp))

(DOMINATE noun-phrase bs-np)
(DOMINATE finite-verb-phrase bs-vp)
```

Adhering to the convention that each aspectual be given a unique name leads to the use of names such as *bs-np* and *basic-s-pattern* to designate the NP and PATTERN features of the template.

These assertions, relating the basic sentence template to linguistic knowledge in the Ace hierarchy, may be illustrated by the following diagram:

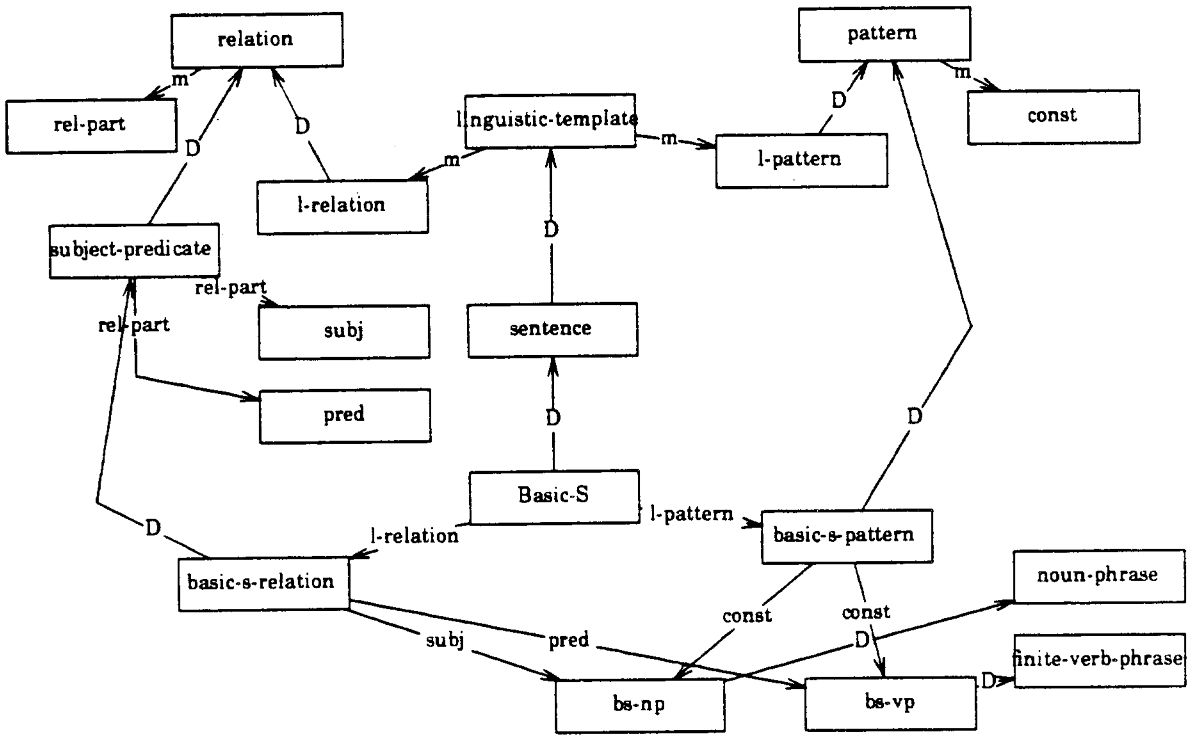

Figure 5-1.

There is a direct correspondence between the diagram above and the Ace assertions given earlier. In future discussions, many of the examples will be presented only in diagram form.

The basic sentence template handles active and passive sentences using the same NP-VP pattern, as is common in most grammars. Unlike the unification grammar example shown earlier, Ace also handles passive verb phrases with or without *by-adjuncts* using the same templates as it does for active verb phrases. The verb phrase template used for the passive with agent, as well as for many other constructs, is the following:

```
[
P-O-S = adjunct-verb-phrase
PARENT-P-O-S = verb-phrase
PATTERN = ( VP ADJUNCT )
VP = [ P-O-S = verb-phrase ]
ADJUNCT = [ P-O-S = adjunct ]
RELATION = [
                PARENT-RELATION = verb-adjunct
                VERB = VERB-PART of VP
                ADJUNCT = ADJUNCT
           ]
]
```

In Ace, this piece of knowledge is represented as follows:

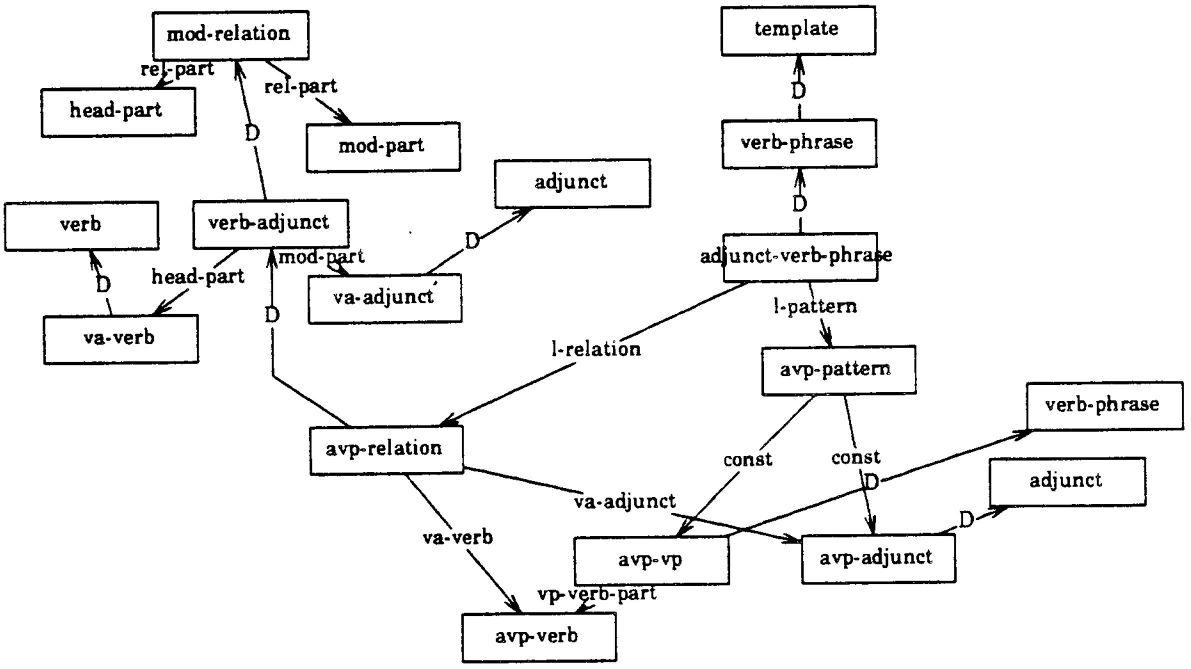

Figure 5-2.

The *verb-adjunct* relation in the figure above is used in the generation process to express conceptual elements of events which do not appear elsewhere in the verb phrase. Its parent category, *mod-relation*, represents the class of linguistic relations which includes adjectival and adverbial modifiers. The *verb-adjunct* relation is a linguistic category which in conjunction with a pattern such as that of the *adjunct-verb-phrase* determines the position of the adjunct in the verb phrase. This use of relations and patterns in the generation process is part of the *restriction* phase of generation, which will be described in Chapter 7.

This *adjunct-verb-phrase* template allows the appendage of adjuncts to verb phrases and specifies that the verb part of the verb phrase is related to the appended adjunct via the *verb-adjunct* relation. No additional grammatical information is needed to represent the role of the *by-adjunct* in the passive verb phrase; its special semantic role is represented by knowledge associated with the *passive-by-adjunct* subcategory of the *verb-adjunct* relation. The term *adjunct* is used loosely here to refer to prepositional phrases and adverbials which are part of the specification of an event, including constituents which are often designated as *arguments*. Since the linguistic behavior of passive *by-adjuncts* is the same as that of adjunct prepositional phrases in general, the pattern above may be used to produce any such adjunct constructs. The following examples from KING illustrate the common use of the *adjunct-verb-phrase* pattern for two different phrases:

```
Input concept:      (kissing (kissee mary1) (surface cheek1))
Type desired:       gerund-phrase
------------------------------------------------------------
Relations used:     (verb-adjunct (va-verb (kissing)) (va-adjunct cheek1)))
                    (verb-obj-relation (vo-verb (kissing)) (obj mary1)))
------------------------------------------------------------
Patterns used:      adjunct-verb-phrase
                    basic-verb-phrase
------------------------------------------------------------
Phrase produced:    kissing Mary on the cheek
==============================

Input concept:      (kiss-giving (recipient mary1) (given-kiss (kissing)))
Type desired:       finite-verb-phrase
------------------------------------------------------------
Relations used:     (verb-adjunct (va-verb (kiss-giving)) (va-adjunct mary1)))
                    (verb-obj-relation (vo-verb (kiss-giving)) (obj (kissing)))
------------------------------------------------------------
Patterns used:      adjunct-verb-phrase
                    basic-verb-phrase
------------------------------------------------------------
Phrase produced:    gave a kiss to Mary
==============================
```

Relations associated with a template specify the correspondence between constituents in the grammatical relation and constituents in the pattern of the template. Thus a relation can be instantiated by the generator before it chooses the template structure which will realize it. This is similar to the way in which PHRED handled flexible word order, by requiring that "patterns" of flexible order be combined with "ordering" patterns to produce output. The PHRED representation, however, made it difficult to use the flexible patterns to represent grammatical fragments which only partially specified the constituents of the final structure. Thus PHRED, like most unification grammar implementations, had separate patterns for dative verbs and verbs which took complements, and had distinct patterns for active and passive sentences and verb phrases. This forced the redundant specification of the aspects of verb phrase structure which are common to active and passive phrases, as well as those which are common to dative and basic verb phrase constructs. For example, information about the case of subject and object, as well as their roles in a sentence, is not influenced by the presence of an indirect object. Thus to represent all this information specifically for the dative verb phrase template would be wasteful. Such redundancy is avoided in Ace by allowing for more flexibility in the patterns and by the use of relations, which permit the encoding of fragments of syntactic information.

The above sentence and verb phrase templates are sufficient for the basic structure of both active and passive sentences. The voice of the verb is not considered at the level of the sentence template, and the *by-adjunct* is likewise treated grammatically as any other adjunct. There must, of course, be some structure which includes the knowledge that the *by-adjunct* attached to a passive verb is special semantically, if not syntactically. This knowledge is associated with the *passive-by-adjunct* relation. This relation is a subcategory of the *verb-adjunct* relation, as illustrated below:

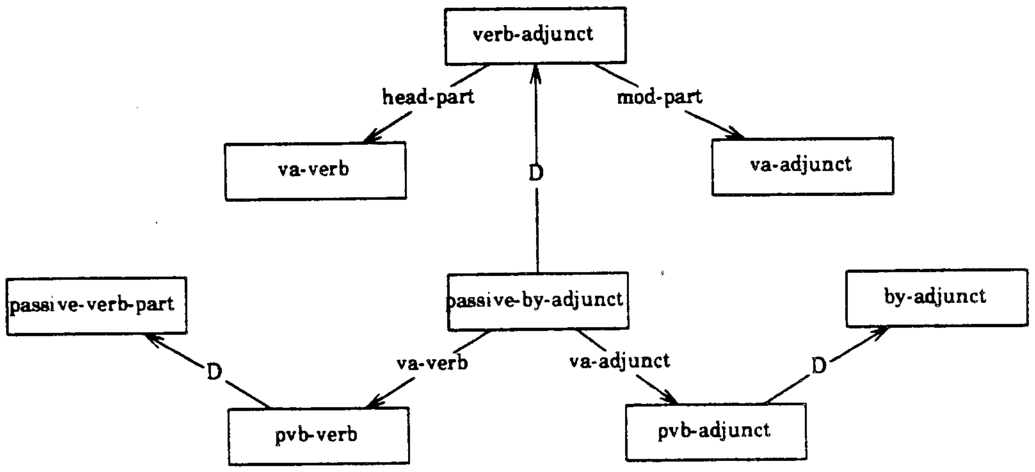

Figure 5-3.

While the *verb-adjunct* relation associates a verb part with the general *adjunct* category which restricts how these constituents appear in a surface structure, the *passive-by-adjunct* relation above constrains the make-up of the verb part and the adjunct. The specification that the VERB component of the relation , i. e. *pvb-verb*, must be of the category *passive-verb-part* constrains the verb part to be of passive voice. The *by-adjunct* category which governs the ADJUNCT part of the relation means that the adjunct must be a prepositional phrase with preposition "by". This is ensured by the following *by-adjunct* template, first in feature notation:

```
[
 P-O-S = by-adjunct
 PARENT-P-O-S = prep-phrase
 PREP = [ LEX = "by" ]
]
```

The organization of this template in the Ace framework is demonstrated in the following figure, where *lex_by* designates the lexical category of the preposition "by":

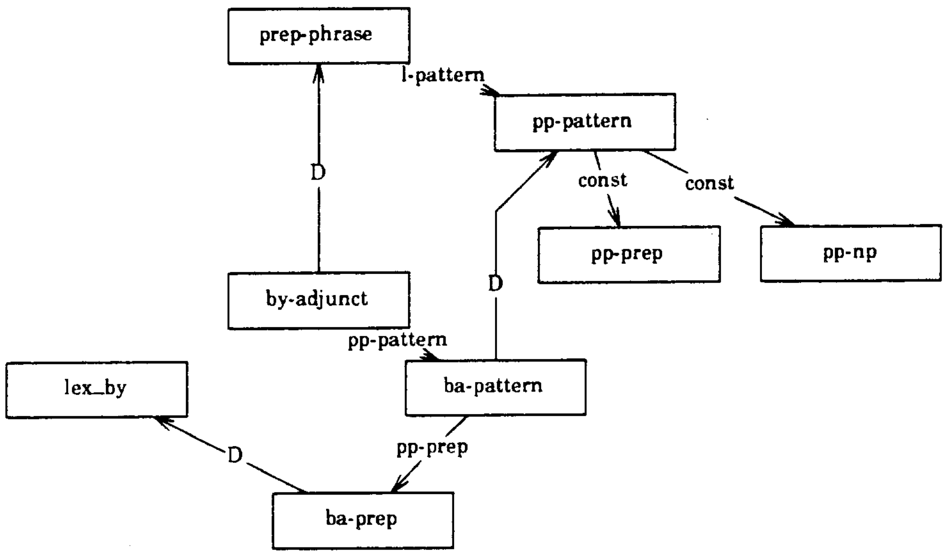

Figure 5-4.

The above knowledge is used to distinguish the *by-adjunct* from other types of adjuncts in Ace, and is used by KING in the construction of the passive-with-agent form. Unlike the passive-with-agent patterns discussed earlier, here information is broken up into small chunks and is attached to linguistic groupings at various levels in the Ace hierarchy. The perspicuity of these groupings becomes fully apparent only when one considers a range of syntactic phenomena which are to be expressed. "John was given a kiss by Mary", "the house built by the French", and "the file to be deleted by the system from your current directory" all make use of the linguistic relationship between passive and *by-adjunct*. The special semantic content of this relation is handled using the REF structured association to be introduced in section 6.2. In terms of parsimony of representation for analysis and generation, it is especially useful to be able to take advantage of the same *verb-adjunct* relationship to explain a variety of surface manifestations, as it allows the association of a relation such as *by-adjunct* with the appropriate semantic information without encoding redundantly the structural information for the various linguistic forms in which it can appear.

This section has presented the basic linguistic templates of the Ace hierarchy using the feature notation. The discussion which follows demonstrates how this linguistic knowledge may be encoded using the framework described in Chapter 4, and discusses the effect of property inheritance within this framework on linguistic knowledge.

5.3. Multiple Inheritance in the Ace Linguistic Hierarchy

This section shows how the structured associations of Ace, and particularly the capacity for multiple inheritance, are used to encode some of the linguistic knowledge used in the construction of simple sentences.

Verb phrases in Ace

Verb phrases provide a good example of the use of a linguistic hierarchy because they exhibit a variety of surface forms while obeying certain regularities. One type of verb phrase is the *dative-vp*, a *verb-phrase* made up of a constituent *dvp-indir*, which includes the verb and noun phrase corresponding to the indirect object, followed by the noun phrase corresponding to the direct object. The treatment of the *dvp-indir* as a separate constituent is done to facilitate the handling of other dative forms. This knowledge about *dative-vp* is represented by the following assertions:

```
(DOMINATE linguistic-template verb-phrase)
(DOMINATE verb-phrase dative-vp)
      with (ROLE-PLAY l-pattern dvp-pattern)
      with (ROLE-PLAY l-relation dvp-relation))


(DOMINATE pattern dvp-pattern
      with (ROLE-PLAY const dvp-indir)
      with (ROLE-PLAY const dvp-np))

(DOMINATE indir-vp dvp-indir
      with (ROLE-PLAY ivp-verb dvp-verb))
(DOMINATE noun-phrase dvp-np)
```

These assertions represent the knowledge that the pattern of a dative verb phrase consists of an *indir-vp* followed by a *noun-phrase*. The *indir-vp* template is represented as follows:

```
(DOMINATE verb-phrase indir-vp)
      with (ROLE-PLAY l-pattern ivp-pattern)
      with (ROLE-PLAY l-relation ivp-relation))

(DOMINATE pattern ivp-pattern
      with (ROLE-PLAY const ivp-verb)
      with (ROLE-PLAY const ivp-np))

(DOMINATE verb-part ivp-verb)
(DOMINATE noun-phrase ivp-np)
```

This knowledge represents the make-up of *dvp-indir*, *dvp-np*, *dvp-verb*, and *dvp-pattern* of the *dative-vp*. These aspectuals may then be used to assign knowledge to all or part of the dative verb part pattern. For example, the following assertions indicate that *dvp-verb* and *dvp-np* are in the *verb-obj-relation*, the relation between verbs and direct objects:

(DOMINATE *relation verb-obj-relation*)
    with (ROLE-PLAY *rel-part vo-verb*)
    with (ROLE-PLAY *rel-part obj*))

(DOMINATE *verb-obj-relation dvp-relation*
    with (ROLE-PLAY *vo-verb dvp-verb*)
    with (ROLE-PLAY *obj dvp-np*))

The knowledge about the relationship between the verb and indirect object is knowledge about the *indir-vp*, encoded as follows:

(DOMINATE *relation verb-indir-relation*
    with (ROLE-PLAY *rel-part vi-verb*)
    with (ROLE-PLAY *rel-part iobj*))

(DOMINATE *verb-indir-relation ivp-relation*
    with (ROLE-PLAY *vi-verb ivp-verb*)
    with (ROLE-PLAY *iobj ivp-np*))

The representational effect of the hierarchy of linguistic structures and features is illustrated in the assertions above. That the *dative-vp* category is DOMINATEd by *verb-phrase* determines the syntactic role of the *dative-vp* in a sentence or clause, as well as providing for the *l-pattern* aspectual. The relationship the verb and indirect object of the *dvp-pattern* is represented as knowledge about the *indir-vp* template, and the semantic knowledge about the relationship between the *dvp-verb* and *dvp-np* can be associated with the *verb-obj-relation*. The knowledge about the dative verb phrase is shown in the following diagram:

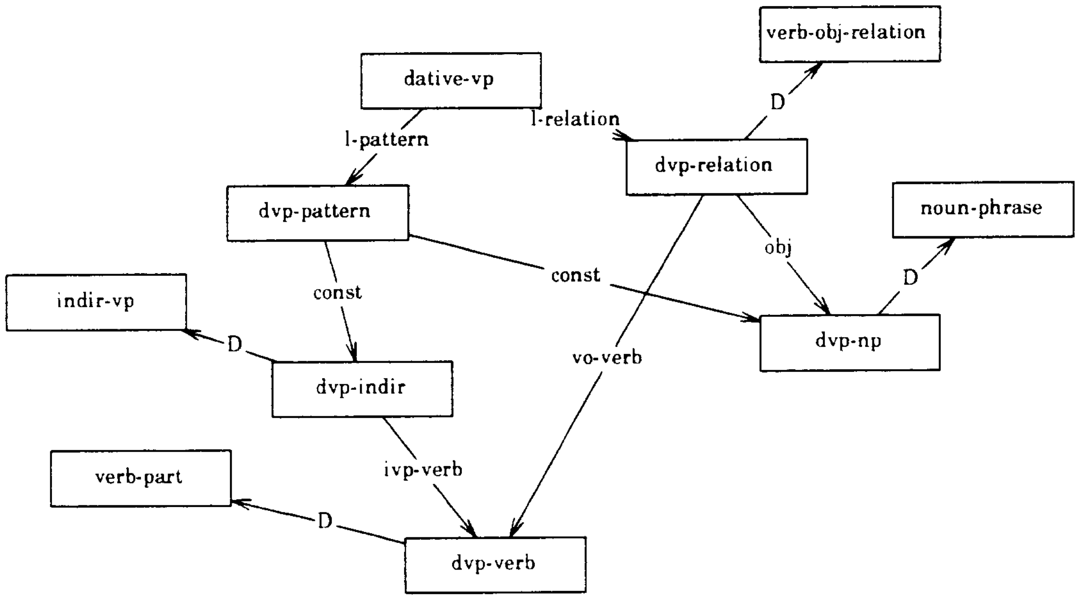

Figure 5-5.

By allowing aspectuals which represent patterns and pattern constituents to play multiple roles, the representation of linguistic knowledge in this manner emphasizes the role of inheritance in the linguistic hierarchy. This topic is considered below.

Multiple Inheritance

One important aspect of the Ace notation is the ability of a linguistic structure to inherit features from more than one parent category. This ability has been alluded to, particularly in the discussion of the similarities between ordinary verb phrases, gerund phrases, and infinitive phrases. Much of the knowledge about these structures is common to all three: All consist of a verb or verb derivative possibly followed by a noun phrase or noun phrases (in the dative structure) and possibly modified by adverbials and adjuncts. But infinitive phrases behave in most cases as noun phrases, exhibiting certain special syntactic properties, and gerund phrases are nouns. These phrases can be handled by allowing the inheritance of features from the a verb phrase category while each template may itself belong to a different p-o-s category. The following templates represent three such categories using feature notation:

```
[
  P-O-S = inf-phrase
  PARENT-P-O-S = verb-phrase
  FORM = infinitive
]

[
  P-O-S = gerund-phrase
  PARENT-P-O-S = verb-phrase, noun
  FORM = progressive
]

[
  P-O-S = finite-verb-phrase
  PARENT-P-O-S = verb-phrase
  FORM = finite
]
```

The hierarchy of verb phrase structures is represented in Ace as illustrated by the following graph:

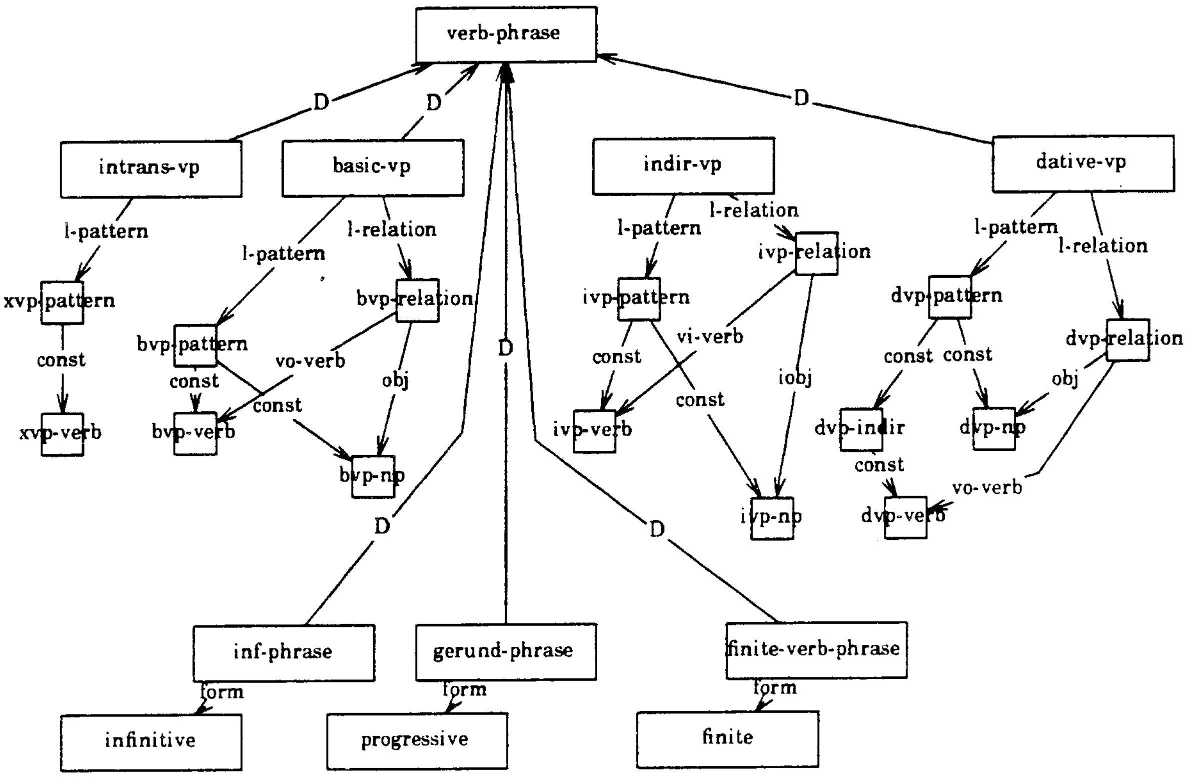

Figure 5-6.

The various verb phrase patterns fall beneath the *verb-phrase* template in the linguistic hierarchy, as do the gerund phrase, infinitive phrase, and finite verb phrase nodes. In order to produce a finite verb phrase, a node lower in the hierarchy must be instantiated. This hierarchical organization permits the gerund phrase and infinitive phrase to have the same linguistic structure as the verb phrase, modulo the form of the verb part. An instantiated verb-phrase in Ace thus inherits most of its internal structure from one category, for example *dative-vp*, and its external behavior from another, for example, *finite-verb-phrase*. The *verb-phrase* category itself plays no external syntactic role--there is no pattern in which a constituent belongs to the *verb-phrase* category and to no lower category, although theoretically there could be. The effective organization of information about verb phrases stems directly from the application of the basic knowledge representation principles of Ace to linguistic knowledge.

Another example of the hierarchical encoding of knowledge in Ace is the representation of "helping verbs". For the purpose of this discussion, the term *compound verb** refers to verbs which are composed of other verbs and in which the first verb is a "helping verb" that carries the tense, person and number of the compound. Verbs such as "jump start" and "hog tie" do not belong to the class of compound verbs considered here. Verbs such as "get paid" and "keep going" do, as do "have gone" and "was given". In some

* This term is used elsewhere in a variety of ways, primarily for verbs which include other parts of speech. The meaning here is somewhat more restricted.

cases, the helping verb can be classified as an auxiliary; that is, it may appear in a variety of constructs such as in subject-aux inversion and tag questions. When the helping verb is an auxiliary, it can be used in a number of constructs, such as in "Has John gone?" and "The book was given to John, wasn't it?" Other helping verbs may not belong to this class: "Kept John going?" and "John got paid, gotn't he?" are unacceptable.

The main organizational category for knowledge about verbs and auxiliaries in Ace is the *helper-verb* relation, which links a helping verb with a main verb. Associated with this category is the knowledge that the person and number of the complete verb correspond to the person and number of the helping verb. The *aux-verb* relation is a subcategory of *helper-verb*, and can be realized in a broader range of linguistic constructs. The *modal-verb* relation is a type of *aux-verb* and carries with it the constraint that the verb must be in non-finite form. This knowledge may be represented by the following hierarchy:

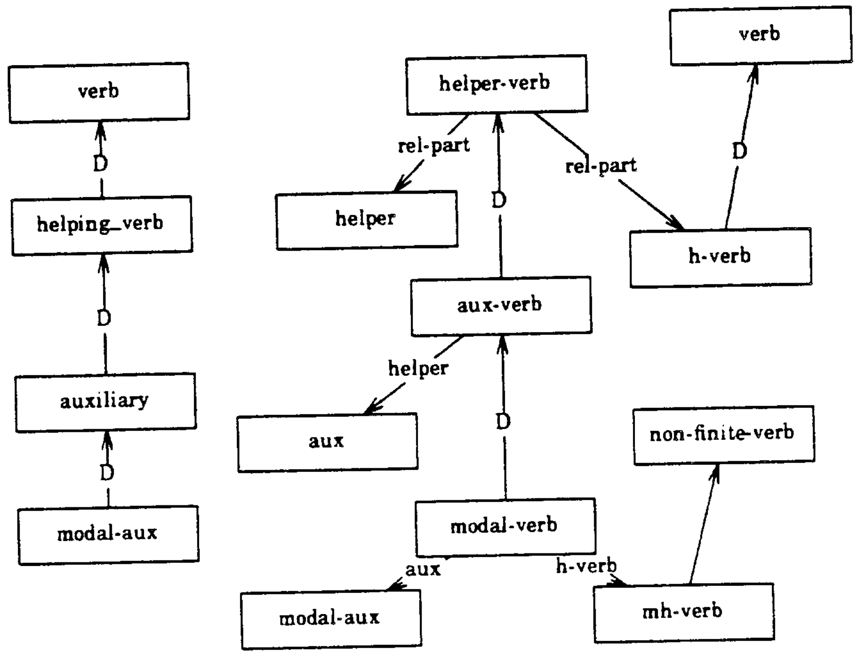

Figure 5-7.

The role of the helpers in determining tense and aspect, as well as in determining the form of the main verb, is thus represented in the specific *helper-verb* relations. The way in which the *helper-verb* relations are realized in surface structure is determined by various pattern templates. The compound verb template, which corresponds to "got beaten" as well as "has left" as these forms might appear in a basic sentence, is as follows:

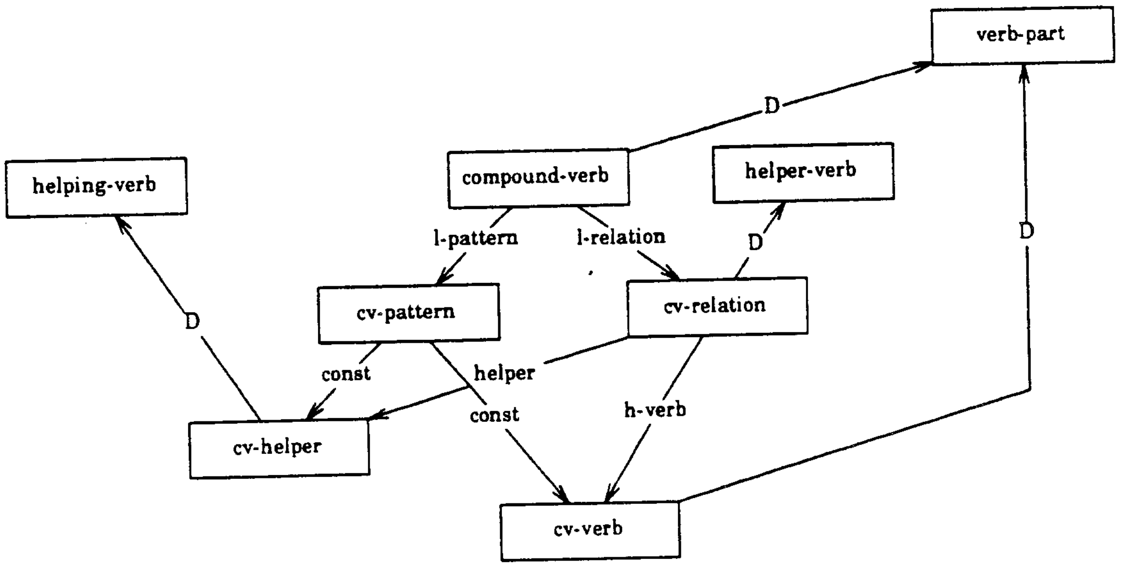

Figure 5-8.

The knowledge above represents one way in which verbs appear with helping verbs. The form of the main part of the verb is not constrained by the template, but is determined by the specific subcategory of the *helper-verb* relation to which the relation being realized belongs. For example, in "must go", the pattern determines the order of helper and main verb, while membership in the *modal-verb* relation determines the finite form of the verb. This is similar to the way in which the *passive-by-adjunct* relation interacts with the *adjunct-verb-phrase* pattern. The restriction of patterns using constraints on relations is discussed in Chapters 7 and 8.

Another form in which the *helper-verb* relation may appear is in a question where the helper appears before the subject, as shown in the following diagram:

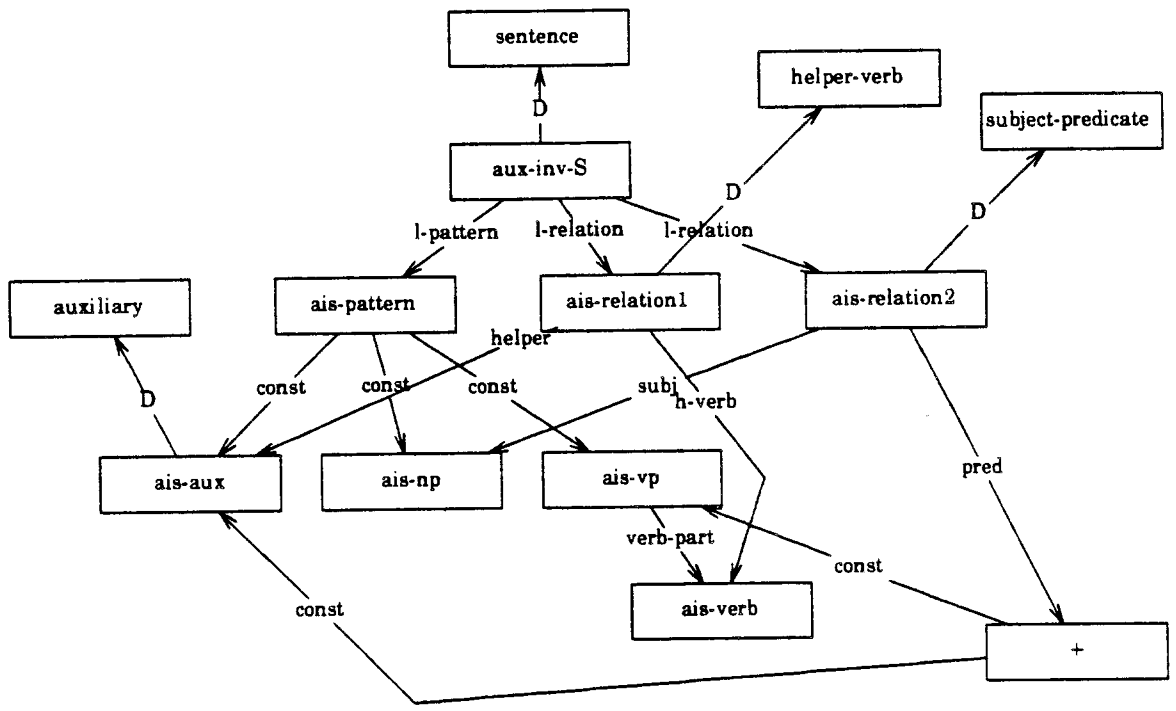

Figure 5-9.

The *aux-inv-S* template above specifies that the constituent in the AUX position must be an element of the *auxiliary* category. Other information about the form of the main verb is determined on the basis of what type of auxiliary is used. This type of sentence has the special property that its predicate includes the auxiliary, as if the auxiliary appeared after the subject. This property is specified by the combination of *ais-aux* and *ais-vp* using the "+" structure. The "+" is used as a special symbol to represent the knowledge that non-adjoining constituents combine to play a single role. The *subject-predicate* relation, which DOMINATEs *ais-relation2*, mandates the agreement of the subject with the complete verb. The *helper-verb* relation, DOMINATing *ais-relation1*, specifies that the complete verb consists of the AUX part in relation to the verb part of the verb phrase. The knowledge that the person, number, and tense of the complete verb is determined by the auxiliary is obtained from the following knowledge about the *helper-verb* relation:

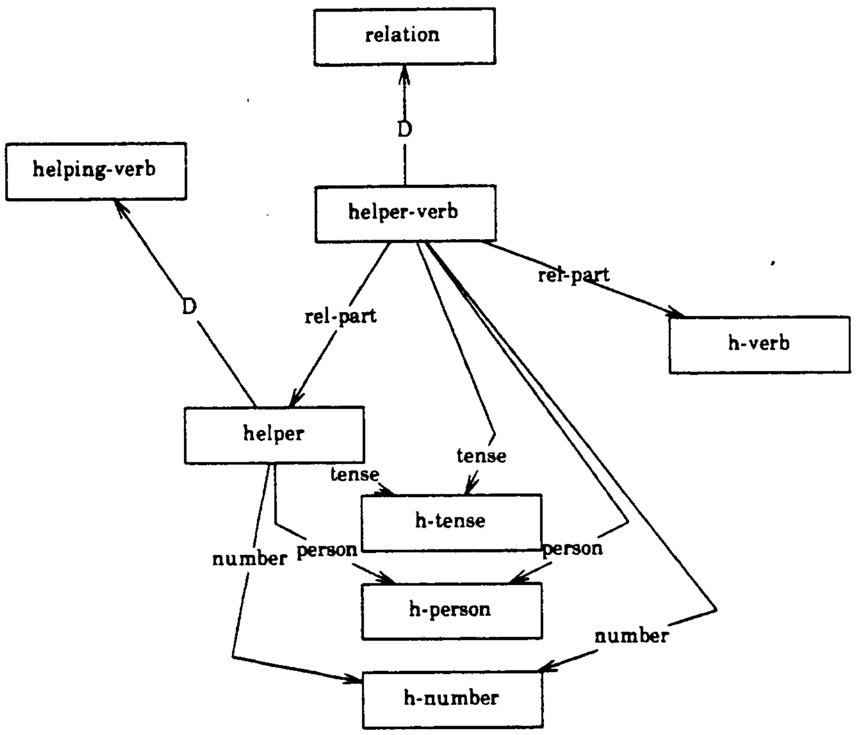

Figure 5-10.

The above representation indicates the correspondence of attributes of the helping verb to attributes of the complete verb. These features correspond to attributes marked by "value" in pattern-concept pairs, and are a substitute for the use of explicit variables in unification grammar. In the production of a sentence of the *aux-inv-S* form, a generator will apply its knowledge of the *subject-predicate* relation to determine that the verb phrase must agree with the noun phrase in person and number. It must then determine the person and number of the noun phrase, and apply its knowledge of verb phrases to constrain production so that this is the person and number of the verb part of the verb phrase. The knowledge above is then applied to specify the person and number of the auxiliary.

The hierarchy of helping verb relations described here highlights the role of inheritance in Ace. Different aspects of linguistic structure and behavior are inherited from different structures in the linguistic hierarchy. For example, knowledge about agreement is represented at the level of the *helper-verb* relation, while knowledge about the form of the verb part may be represented at the level of *modal-verb*. The role of multiple inheritance in the case of helping verbs is more subtle. One example of such multiple inheritance is in the case of passive verb parts. The helping verbs "get" and "be" are both treated as passivizers in Ace. Thus "John got told the story by Mary" and "John was told the story by Mary" may be handled using the same linguistic basic linguistic knowledge. "Be", however, is an auxiliary, while "get" is not. These verbs thus inherit certain aspects of their linguistic behavior from different categories. The form of the main verb is determined by membership in the *passive-verb* category, while auxiliary behavior is determined by membership in the *modal-verb* category. The relation *vpas*, which represents the relation between passive "be" and a main verb, belongs to both categories, and has the further constraint that the helping verb is "be".

The examples given here present the general handling of helping verbs in the Ace framework, particularly the interaction between patterns and relations. The two main differences between this formulation and that which exists in most implemented generation systems are the following:

(1) The hierarchical encoding of linguistic knowledge avoids redundancy in the encoding of knowledge about verbs. In most feature systems, for example, knowledge about a particular auxiliary is represented by a set of features which may duplicate sets of features of other auxiliaries. In Ace, these sets of features are replaced by more general linguistic categories.

(2) The use of the relation to associate helper and verb allows linking of their lexical and semantic properties to relations rather than to the several patterns in which they may appear. For example, the knowledge about passive verbs in Ace is encoded as knowledge about the *passive-verb* relation, independent of the structures in which these relations are realized. Other systems have either encoded this knowledge redundantly or applied various rewriting rules to represent similarities in linguistic structure.

Summary

This chapter has presented the basic facilities of Ace for linguistic representation and has given examples of the use of the Ace feature system for the encoding of grammatical knowledge. The discussion has focussed on the application of four principles to distributing linguistic knowledge in the Ace hierarchy. This distribution leads to a more parsimonious encoding of linguistic knowledge which facilitates the addition of new knowledge to the hierarchy. The next chapter discusses how this linguistic knowledge is related to the conceptual hierarchy introduced in Chapter 4.

6. Associating Language and Meaning

The most essential representational tools for language generation are those which facilitate the mapping from conceptual structures to linguistic structures, and those which help to guide the choice of language based on conceptual or intentional information. This chapter will focus on the aspects of the Ace representation which satisfy this need.

Chapter 4 presented the foundations of the Ace framework and its application to meaning representation, proposing that concepts be hierarchically organized, linked together via structured associations. Chapter 5 described how linguistic knowledge can also be hierarchically organized within this framework, and how features can thus be inherited through linguistic categories. The idea of a linguistic relation was proposed, to distinguish structural relationships from ordering relationships in grammatical constructs. Naturally, the knowledge required to produce correct and appropriate utterances includes both of these classes in addition to the links which bind the classes together.

6.1. Basic Principles

The goal of taking advantage of explicit referential knowledge, discussed in Chapters 1 and 3, along with the framework presented in Chapters 4 and 5, suggests two more principles directed towards the association of linguistic and conceptual knowledge:

Representation Principle 8. Correspondence of Linguistic and Conceptual Structures.
Linguistic structures, such as lexical terms, linguistic categories, and grammatical structures, are directly associated with conceptual categories. General linguistic categories, such as *verb-indir-relation*, correspond to general conceptual categories, such as *transfer-event*. Specific lexical terms, such as "buy" and "sell", are linked to more specific concepts, such as *buying* and *selling*. Grammatical structures, such as the *modifier-noun* relation, are associated with conceptual relations, such as MANIFEST.

Representation Principle 9. Association of Linguistic Features with Conceptual Attributes.
The structured association permits the relation of linguistic aspectuals with conceptual aspectuals via ROLE-PLAY. Linguistic features, such as *dvp-pattern* and *ivp-verb* as described in Chapter 5, are represented as aspectuals in Ace, as their status depends on the template of which they are a part. The association of these features with conceptual attributes goes along with the association of the template with a concept. For example, the indirect object feature *iobj* is an aspectual of *verb-indir-relation*. The linguistic feature *iobj* is linked to the conceptual attribute *recipient*, an aspectual of the *transfer-event* concept. The direct-object feature *obj*, an aspectual of *verb-obj-relation*, is linked to the conceptual *object* concept, an aspectual of the *simple-event* concept. Thus linguistic features have conceptual correlates.

The next section describes how these principles are realized in the association of language and meaning in Ace.

6.2. Linking Linguistic and Conceptual Structures Using REF

The main tool for representing relationships between linguistic structures and conceptual structures in Ace is a structured association called REF. The REF association, designating "referential" relationships, is similar to VIEW, except that it joins language to concepts instead of concepts to concepts.

The linguistic knowledge presented earlier has included information about how verbs and their objects or indirect objects may appear in surface structure. The knowledge essential to build these structures, however, is contained in the correspondence between linguistic relations and conceptual entities. This information is presented below.

```
(REF verb-indir-relation transfer-event
     with (ROLE-PLAY vi-verb transfer-event)
     with (ROLE-PLAY iobj recipient))
```

The above assertion indicates that the linguistic relation *verb-indir-relation* is associated with the concept of a *transfer-event*, and that the indirect object of the relation corresponds to the *recipient* of the *transfer-event*. Furthermore, the *vi-verb* of *verb-indir-relation* relation refers to the *transfer-event* itself.

A similar piece of knowledge is used to relate the *verb-obj-relation* to *simple-event*:

```
(REF verb-obj-relation simple-event
      with (ROLE-PLAY vo-verb simple-event)
      with (ROLE-PLAY obj object))
```

As these relations are realized in the dative verb phrase pattern considered in the previous chapter, the dative verb phrase is indirectly linked to *transfer-event*s and *simple-event*s. *Transfer-event*s, as introduced in Chapter 3, provide an abstract category with respect to which the meaning of aspectuals such as *recipient* can be defined. *Simple-event*s are events which designate a single *object*. The assertions above define relationships in Ace between the linguistic aspectual *iobj* and the *recipient* of a transfer, and between the linguistic aspectual *obj* and the conceptual *object*.

The knowledge which links the dative pattern to its related conceptual structures is diagrammed below:

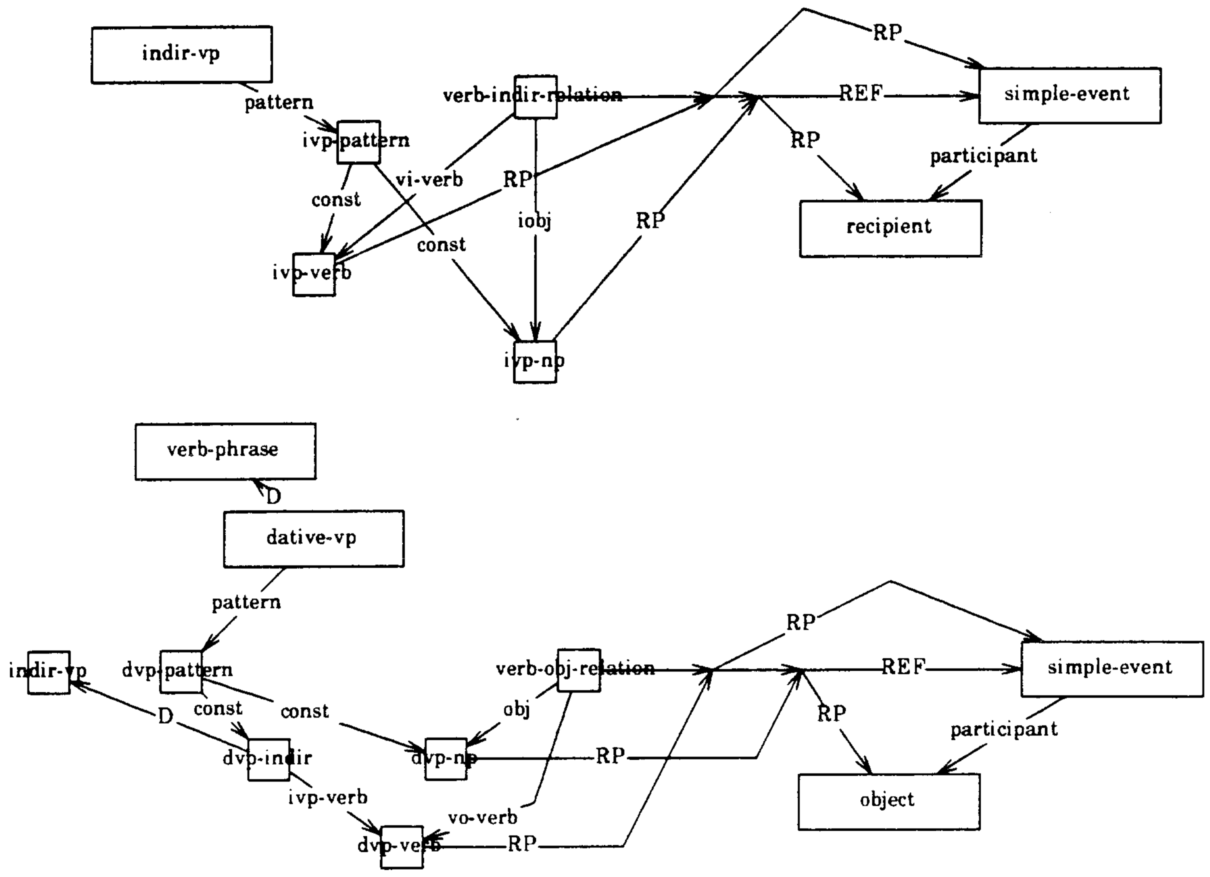

Figure 6-1.

The above diagrams illustrate how the syntactic structure of the dative verb phrase is associated with its conceptual structure. The dative verb phrase is composed of a verb part *dvp-verb* followed by two noun phrases. It is, of course, possible to have adverbial modifiers in various places, which is facilitated by treating the verb part and its indirect object constituent as a distinct pattern. This pattern, *ivp-pattern*, is shown in the top diagram above. The verb part and noun phrase in the *ivp-pattern* pattern belong to the *verb-indir-relation*, which associates with the indirect object the concept of *recipient*. This diagram presents a slightly abbreviated version of the Ace representation given in Chapter 5, as *ivp-verb* and *ivp-np* are associated only indirectly with the *verb-indir-relation* via the *ivp-relation* aspectual.

The structured association between the *verb-obj-relation* and the *simple-event* concept in the lower diagram of Figure 6-1 links the object of the verb in the dative verb phrase to the object of the simple event. In this association, as in the association of the *verb-indir-relation* with the *transfer-event*, the verb part *dvp-verb* is associated via ROLE-PLAY with the event itself, rather than with any aspectual of the event.

Like the pattern-concept pair or the unification grammar template, REF is a means of associating linguistic structure with conceptual structure. The template, however, is replaced with an explicit structural link in the knowledge network. This makes it easier to perform the knowledge-driven aspects of generation because no querying or complex matching is necessary. The use of the REF association also facilitates incremental generation by encoding knowledge about referential relationships as structured associations at various levels in the Ace hierarchy*.

In many ways the REF structured association is similar to DOMINATE and VIEW. For example, in the assertion

(DOMINATE *action selling*
with (ROLE-PLAY *actor seller*))

the association between *actor* and *seller* is connected to the association between *action* and *selling*. In the assertion

(VIEW *ct-merchandise-transfer selling*
with (ROLE-PLAY *merchant seller*))

the association between *merchant* and *seller* is linked to the VIEW association between *ct-merchandise-transfer* and *selling*. In the above example, we have the association

(REF *verb-obj-relation simple-event*
with (ROLE-PLAY *obj object*))

which binds the relation between *simple-event* and *verb-obj-relation* to the REF association between the aspectuals *object* and *obj*. The path between a linguistic structure and its meaning, therefore, is represented as a sequence of structured associations. While these associations are distinguished in the notation according to the types of entities which they relate, they share sufficient structure that a relatively simple mechanism can be used to guide their application and interaction in the generation process. This mechanism is described in Chapters 7 and 8.

The graphical form of the representations described here corresponds directly to the assertions in the list notation. The Ace representations presented henceforth are provided only in the diagram form.

In general, lexical categories in Ace are related to one or more conceptual categories by a REF association. We thus have:

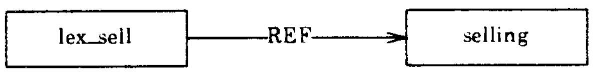

Figure 6-2.

The diagram above shows the REF association between the lexical category "sell" and the *selling* action. The form of a verb is determined by

* Those familiar with KL-ONE and other similar representation languages will observe that the Ace representation tends to avoid matching by assuming that the concept being generated from is assigned to a specific category. Thus some of the work done by template matching in systems such as PHRED is done during the classification of a concept in Ace. This is consistent with the Proliferation of Conceptual Categories principle: the generator need not obtain conceptual information that is more specific than that contained in the category in which a concept has been classified.

the lexical category in conjunction with the other linguistic categories to which the verb belongs. The following examples from KING illustrate the use of lexical categories constrained by linguistic properties:

```
Input concept:      (given-kiss (kissing))
Type desired:       noun
-----------------------------------------------------------------
Constraints used:   lex_kiss
                    number singular
-----------------------------------------------------------------
Word chosen:        kiss
=================================
Input concept:      (selling)
Type desired:       gerund
-----------------------------------------------------------------
Constraints used:   form progressive
-----------------------------------------------------------------
Word chosen:        selling
=================================
Input concept:      (selling)
Type desired:       finite-verb
Input constraints:  tense past
                    number singular
                    person third
-----------------------------------------------------------------
Word chosen:        sold
=================================
```

The means by which constraints are propagated and lexical choices are made will be considered in Chapter 7.

6.3. Linking Linguistic Structures to Aspectuals Using REF

Chapter 4 discussed the idea of *aspectual* concepts, or concepts which have meaning only in relation to other concepts. Aspectuals, however, are still well-defined concepts in Ace, and may be joined to linguistic structures using REF associations. Consider, for example, the *actor* aspectual. There seems to be a close relationship between this aspectual and the active voice feature. I hypothesize that in the sentence "John gave Mary a book", the verb phrase "gave Mary a book" is linked by a structured association to a conceptual structure. This conceptual structure does not seem to be an absolute concept, but corresponds to the meaning of "played the role of *giver* in *giving* Mary a book." The "activeness" of this phrase is the linguistic realization of the meaning "play the role of *actor*". In "John was given a book by Mary", the verb phrase corresponds to the meaning of "played the *role* of *recipient* ...." It is not practical to treat this correspondence as dependent on the correspondence between sentence and concept, because phrases such as "giving Mary a kiss" may refer to a role without explicitly saying anything about the player of the role. In Ace, we may represent the relationships between linguistic features such as *voice_active* and aspectuals such as *actor* using REF links, as illustrated by the following diagram:

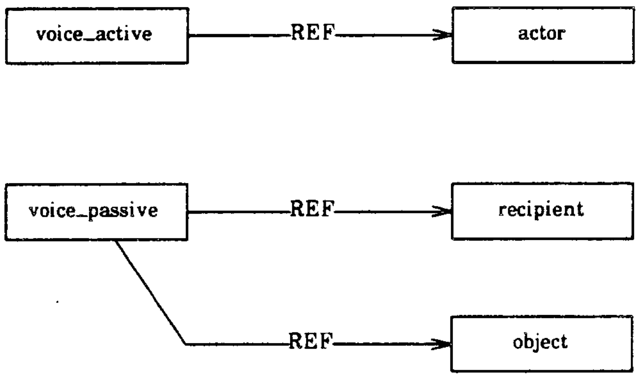

Figure 6-3.

In the above diagram, the linguistic features *voice_active* and *voice_passive* represent constraints on linguistic structure. These are joined via REF links to conceptual structures. This represents the knowledge that the active voice feature relates to the *actor* aspectual, and the passive voice feature to the *recipient* or *object* aspectual. Thus a linguistic structure which is constrained by the active voice feature corresponds to the meaning, "played the role of *actor* in ..." In the case of "gave Mary a book", the verb phrase means "played the role of *giver* in *giving* Mary a book", because the *actor* role in *giving* is played by the *giver*. This knowledge about the active and passive voices is thus represented independently of the linguistic structures which the voices constrain.

The *subject-predicate* linguistic relation described in Chapter 5 is associated by a REF link with a conceptual structure called a *predication*. This structure relates a conceptual entity to a role which it plays, called the *c-predicate* (for "conceptual predicate") of the *predication*. *Predications* may be referred to in a variety of ways other than by the *subject-predicate* relation, as in "John's sale of the book", and "knowing the ropes". As discussed above, the predicate part of a sentence seems to correspond to the meaning "played the role of ...." The *c-predicate* role thus seems to be played by another aspectual. For "John sold Mary a book", the *predication* may be illustrated by the following figure:

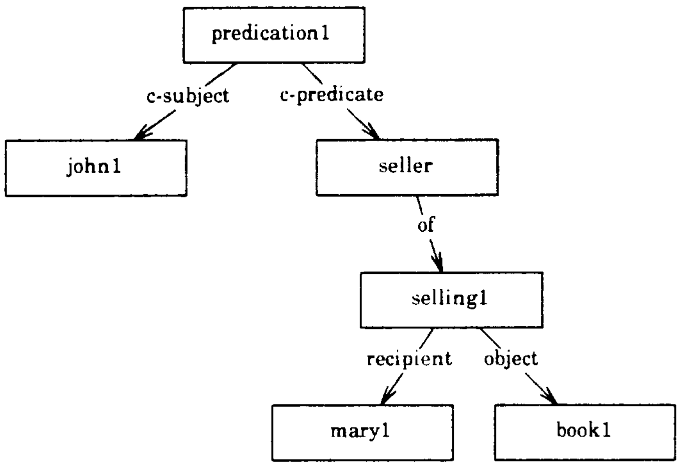

Figure 6-4.

The above *predication* represents a concept which might be expressed in "John's selling Mary the book" as well as in "John sold Mary the book"; The role of *c-predicate* in the above structure is played by the role of *seller* in the concept *selling1*. The *c-predicate* in the above examples is expressed by "...selling Mary the book" and "...sold Mary the book". Because the *seller* aspectual is in a ROLE-PLAY relation with the *actor* aspectual, *seller* is implicitly associated with the *voice_active* structure. Thus the *c-predicate* of the above knowledge structure is expressed in the active voice. This representation allows certain knowledge about expressing such predicates to be independent of the corresponding subjects. This permits a generator to make choices such as selecting the voice of a verb phrase based on the meaning of the verb phrase, rather than of the sentence in which the verb phrase appears.

*Predication*s are an abstract entity in the Ace hierarchy. Any conceptual relation may be viewed as a *predication*, although only certain such predications may be easily described in English. The relationship between concepts and *predication*s is special because the *c-predicate* of the *predication* is linked to the role which the *c-subject* plays. When a concept is viewed as a predication, some aspectual of that concept becomes the *c-predicate*, and the concept playing the role of that aspectual becomes the *c-subject*. This may be diagrammed as follows:

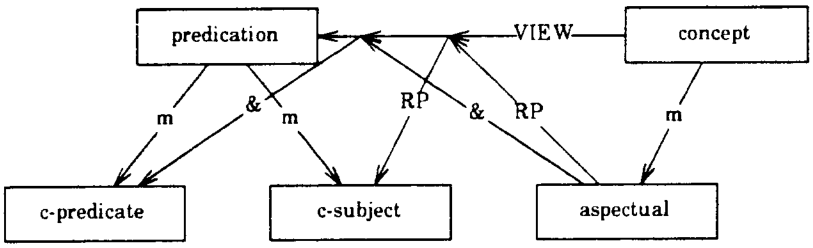

Figure 6-5.

The "&" in the above diagram indicates a special type of ROLE-PLAY relation, in which the role played by the *c-subject* is associated with the *c-predicate*. The normal ROLE-PLAY relation "RP" indicates that whatever plays the role of *aspectual* plays the role also of *c-subject*. The "&" relation, on the other hand, indicates that the *aspectual* itself plays the role of *c-subject*. For example, the *predication* shown in figure 6-4 may be derived by applying this VIEW to *selling1*: The concept *john1*, which plays the role of the aspectual *seller*, is joined by ROLE-PLAY to play the role of *c-subject*. The aspectual *seller* is joined by "&" to play the role of *c-predicate*. This VIEW is thus used to select a particular aspectual of a concept for the *c-predicate* role. Since most concepts have more than one aspectual, a generator using this VIEW must choose among the aspectuals of a concept to determine how to form a *predication*. It is this choice which determines, for example, whether a sentence will be in active or passive form. Chapter 8 will provide more detail on how these VIEWs are used in the production of an utterance.

6.4. Knowledge About Specialized Constructs

The examples given in section 6.2 are simple illustrations of the use of associations between language and meaning at a variety of levels. These levels become especially apparent in the representation of knowledge about specialized constructs, whose meaning can be only partially represented as associations from more general linguistic structures. The analysis in Chapter 3 argued for a representation in which specialized knowledge interacts with general knowledge. The Ace hierarchy facilitates this interaction.

The three principles given at the beginning of Chapter 5 are particularly relevant in the representation of grammatical and extragrammatical constructions and of non-core grammatical constructions. Extragrammatical phrases, such as "by and large" and "all of a sudden" are treated as special patterns, just as they were in the pattern-concept pairs of PHRED and as they would ordinarily be handled in unification grammar. Non-core constructions, such as "The more <S>, the more <S>" and "<X>, let alone <X>" are also treated as patterns, but may fall in the linguistic hierarchy at a level below certain other syntactic patterns and above some more specific patterns. For example, "The more <S>, the more <S>" construct, illustrated by "The more Fred eats, the more repulsive he becomes" may be a special case of the construct "The <comp> <X>, the <comp> <X>", as in "The less money I make, the happier I am". It may also be considered a parent construction of specific phrases, such as "The more, the merrier". The construct "<X>, let alone <X>", as in "Bill won't even each fish, let alone sashimi" may be a subcategory of "<X>, <sub-conj> (X)", a general subordinating conjunction construct. The "let alone" construct is, however, more constrained than most subordinated constructs. Experience with systems such as PHRED and Ana suggested handling such specialized constructs with specific patterns; the Ace hierarchy allows such patterns to inherit from more general constructs.

The means in which specialized knowledge is encoded varies according to the particular construct. In the case of simple, non-productive constructs, such as "kick the bucket", the specialized interpretation and the literal meaning are actually associated with different linguistic patterns. This accounts both for the lack of syntactic flexibility of the idiom and the lack of any substantive contribution to its meaning by its individual components. For example, the "bucket" part of the "kick the bucket" pattern does not actually refer to any sense of the word "bucket", nor does it refer to anything. The "kick the bucket" knowledge can thus be represented as given below:

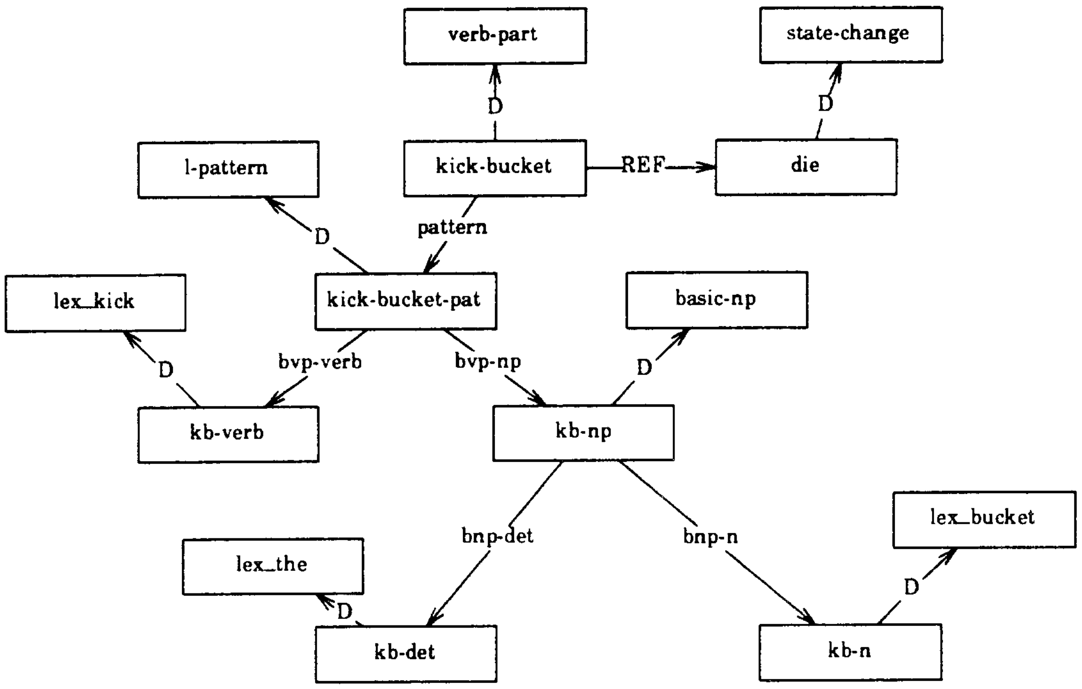

Figure 6-6.

"Kick the bucket" is thus a linguistic pattern whose only variation in structure is determined by the form of the verb "kick". The noun-phrase part has a specified structure, but the verb part of the verb phrase may be further specified, and obeys the constraints inherited from other *basic-vp* patterns. Thus "kicking the bucket" as a noun and "to have kicked the bucket" as an infinitive phrase may instantiate the same pattern, but "kicked a bucket" will not.

A motivated construct such as "bury the hatchet" corresponds to no particular linguistic pattern, as it may appear in phrases such as "The hatchet was buried at Appomattox" and "John and Mary buried their hatchet". This type of specialized construct is awkwardly handled in all current systems, including PHRAN and PHRED. In Ace, it corresponds to the following structure:

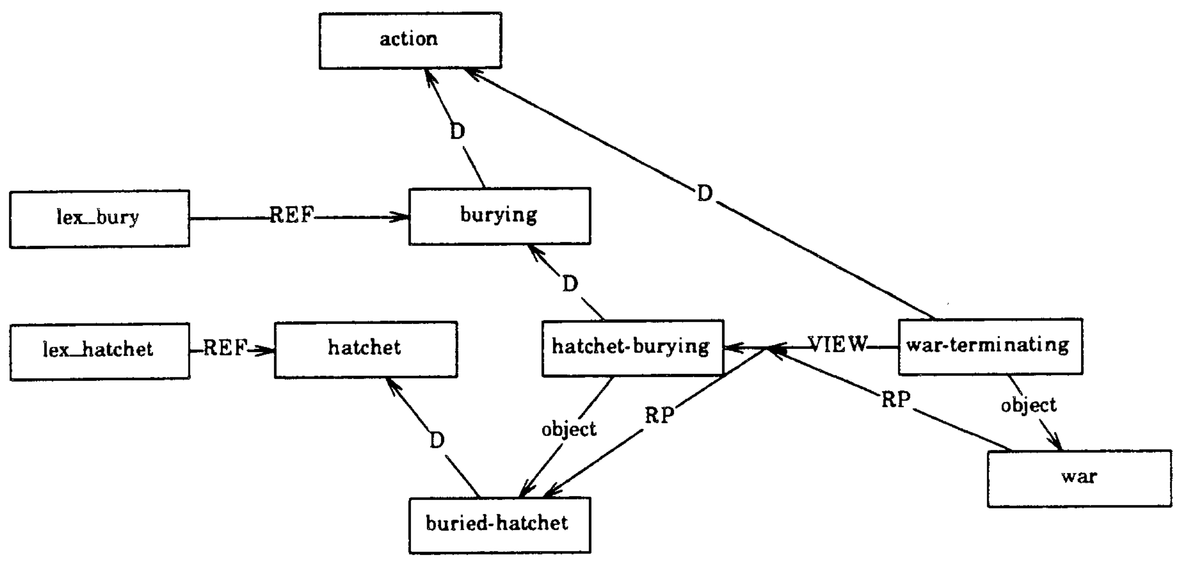

Figure 6-7.

The above representation handles the problem of representing the structure of "bury the hatchet" in a manner substantively different from PHRAN/PHRED. On the conceptual level, the concept of terminating a war is associated with the "burial" concept via a VIEW, the type of structured association generally used for such metaphorical relationships. However, there is no particular linguistic structure that can be used either as a trigger for this VIEW in analysis or as a means of determining the preferability of the use of the lexical terms "bury" and "hatchet" in generation. The solution embedded in the above representation is to treat the *hatchet-burying* as a more specific concept than *burying*. The object of this action, *buried-hatchet*, is a more specific concept than *hatchet*; it is the metaphorical hatchet that is buried at the end of a war.

The practical solution described above for the representation of flexible linguistic constructs takes advantage of the Proliferation of Conceptual Categories Principle presented earlier. It is a simple trick in implementing a system to create a special concept corresponding to a particular construct rather than to try to encode the knowledge about the construct as part of the linguistic knowledge of the system. The solution seems bold from a representational perspective because it goes against the grain of approaches which try to achieve a strong separation of linguistic and conceptual knowledge. The *hatchet-burying* concept exists essentially to satisfy a linguistic demand, supplying an indirect link between the lexical terms "bury" and "hatchet" and the concept of terminating a war. The effect of this organization is to link the specialized meaning of the "bury the hatchet" construct to the use of the lexical term "bury" and the lexical term "hatchet" where "hatchet" refers to the conceptual *object* of the *hatchet-burying*. This allows for the realization of the construct in expressions such as "The hatchet was buried at Appomattox" as well as in its more typical forms.

The richness of the "bury the hatchet" metaphor makes the description above a fairly simplistic one, but the same approach is manifest in a variety of more simple expressions. For example, in the sentence "John gave Mary a hug", the construct involves a similar relationship between a verb and its object. There is no single syntactic structure which can be identified with the specialized interpretation, yet intuitively the specialized meaning seems tied to the use of the verb "give" in conjunction with the object "hug". While the metaphorical connection between the *hugging* action and the *giving* action is dependent on the "acting upon is giving" metaphor, this "hugging is hug-giving" metaphor must be associated with the particular lexical items "give" and "hug". The following diagram illustrates how both objectives may be accomplished:

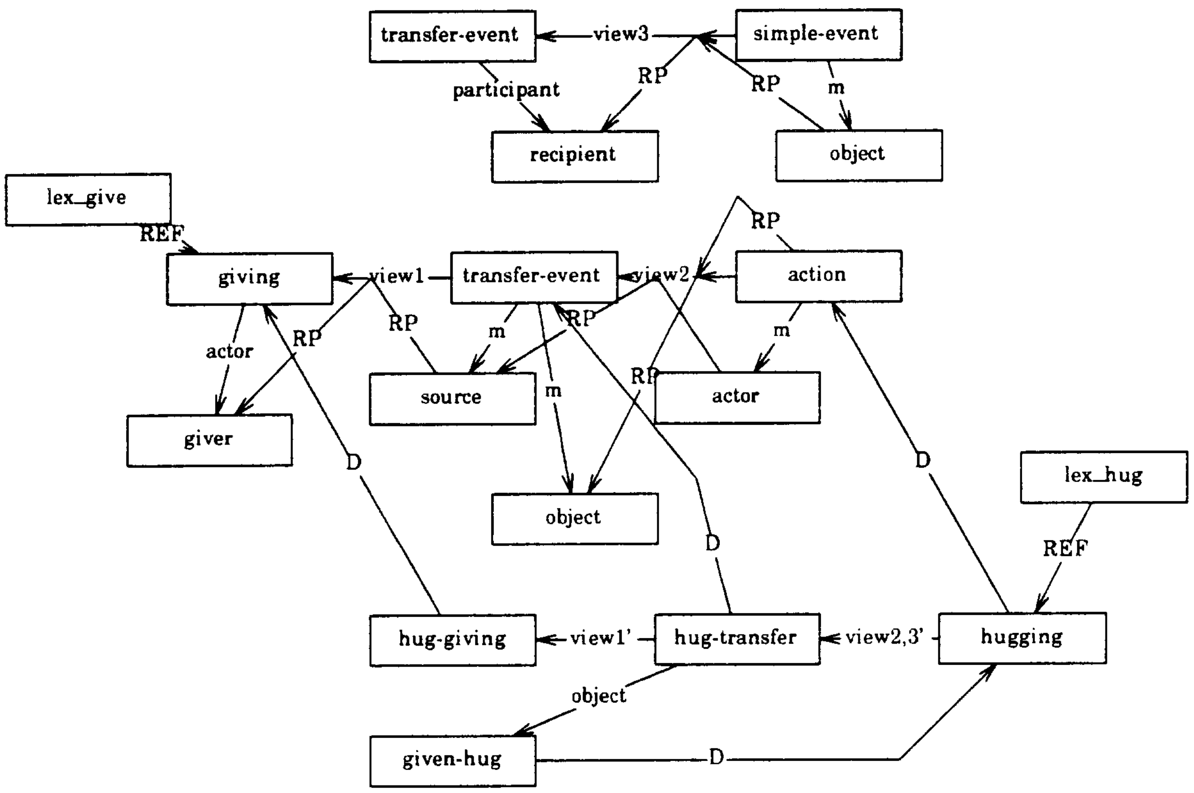

Figure 6-8.

In the above diagram the link between *hug-giving* and *hug-transfer* is labeled *view1'*, to indicate that it is DOMINATEd by the structured association between *giving* and *transfer-event*. This association, *View1*, relates the *source* of the *transfer-event* to the *giver* or *actor* of the *giving*. By inheritance and ROLE-PLAY, *view1'* associates the *giver* of the *hug-giving* action with the *source* of the *hug-transfer* event. The views *view2* and *view3* represent metaphorical associations that *action*s may be VIEWed as *transfer-event*s from the *actor* to the *object*. *View3* indicates the correspondence between the *object* of an action or event and the *recipient* of the transfer, and between the event and the *object* of the transfer. *View2* represents the relationship between *source* and *actor*. This structured association may correspond to a conceptualization of actions as transfers of energy from the actor to the object being acted upon, a common metaphor for specialized constructs such as "John took a fall", and "John got a punch from Bill". The Ace system allows for the representation of constructs such as these to take advantage of their common metaphorical associations.

The association between the *hug-transfer* event and the *hugging* action, labeled *view2,3'*, allows the inheritance of the knowledge that the *source* of the *hug-transfer* corresponds to the *actor* of the *hugging* action, that the *object* of the *hug-transfer* corresponds to the *hugging* action itself, and that the *recipient* of the *hug-transfer* corresponds to the *object* of the *hugging*. The *view2,3'* association is DOMINATEd by both *view2* and *view3* and thus inherits their ROLE-PLAYs.

It is apparent that the use of inheritance and structured associations makes the encoding of phrases such as "bury the hatchet" and "giving a hug" easier, but the representations here raise some legitimate questions about the cognitive reality of the Ace approach. For example, the Proliferation of Conceptual Categories Principle has inspired the creation of concepts such as *hug-giving* and *hatchet-burying*; this raises the question of whether these concepts are anything more than just a notational convenience.

In the case of *hatchet-burying*, there seems to be little evidence other than introspection for the existence of a concept of ending a war which is specifically invoked by the "bury the hatchet" expression. But certainly there must be *some* common structure invoked by a diversity of linguistic manifestations such as "The hatchet seems to have been buried" and "John and Mary buried their hatchet". The question, then, is whether this structure is conceptual or linguistic. The answer, in Ace, is that it doesn't really matter. The very nature of the Ace representation suggests an especially close association between certain conceptual structures and linguistic elements. Thus it is possible to have a concept such as *hatchet-burying* which has very little knowledge independently associated with it other than the knowledge that the lexical items "bury" and "hatchet" are used to describe it. It is a concept that seems to have no significance other than as a linguistic tool. On the other hand, there is no reason *per se* why there could not be a linguistic structure which played the same role as the *hatchet-burying* does here. The option to create a *hatchet-burying* concept was exercised because the parent class of conceptual structures, *actions*, seems to correspond well to the ways in which the "bury the hatchet" construct can appear.

The *hug-giving* concept may be justified for essentially the same reason as the *hatchet-burying* concept: There must be some knowledge structure representing the underlying construct, and it seems as reasonable to have it be a conceptual structure as to be a linguistic structure. In more restrictive expressions such as "kick the bucket", the underlying structure is unquestionably a linguistic entity. In more flexible cases such as "bury the hatchet" and "giving a hug", it makes more sense that it be a conceptual entity.

The existence of the *hug-transfer* concept, however, is more tentative. Superficially, it seems that the concept should exist, as expressions such as "Mary got a hug from John" and "Mary received a hug from John" seem to invoke the same VIEW as "John gave Mary a hug". However, "Mary took a hug from John" is questionable. If there is in fact a *hug-transfer* concept, it seems that one should be able to use this latter form to refer to it. One might argue that there is a constraint, inherited from the *actor* aspectual, that a *taker* must have some kind of agent-like properties with respect to the transfer, but this seems contradicted by examples such as "John took a punch in the mouth" and "Humpty took a fall".

There are two good answers to this problem, both of which assume that to some degree language is arbitrary in the constructs that it allows and does not. The first answer, which is behind the representation given here, is that concepts such as *hug-transfer* do exist, but that one normally does not use action verbs such as "give" and "take" to refer to them, and that the specific association between "give" and *hug-transfer* effectively countermands this generalization. This specific association thus *preempts* general linguistic knowledge in the case of "give a hug". This conclusion is supported by the fact that the sentence "Mary took a hug from John" belongs to the class of constructs that are comprehensible and even reasonable, but which are just not used. Thus it is acceptable to have a representation which indirectly associates *hug-transfer* with *taking-action*, so long as the representation does not ordinarily lead to the generation of the awkward construct. Since specific associations are preferred to indirect associations, the lexical term "give" is better than "take" in describing *hug-transfer*s.

The second answer is that the association between *hug-giving* and *hugging* exists, but does not go through the concept of *hug-transfer*. This would suggest a direct VIEW relation between *hug-giving* and *hugging*, which would inevitably be DOMINATEd by a general VIEW between *giving*s and *action*s, relating *actor* to *giver* and *object* to *recipient*. The advantages of this approach are twofold: First, processing efficiency may be increased by having more direct mappings between meaning and language; and, second, eliminating the *hug-transfer* concept precludes the use of incorrect constructs such as "taking a hug". However, the "giving a hug" expression would in this way be represented independently of "getting a hug" and "receiving a hug". The combination of structured associations to produce more direct associations can be called *short-circuiting* (*cf.* Wilensky, 1983), and generally results in processing advantages but leads to a less parsimonious and possibly less extensible representation.

A *macro-association*, as mentioned in Chapter 3, is a short circuit between structures otherwise indirectly associated. Macro-associations connect sequences of structured associations. These associations may be applied in order to make use of intermediate concepts in the sequence without considering all the associations which relate to these intermediate concepts. For example, it would seem wasteful to view *hugging* as a *hug-transfer* event and thus consider all the structured associations relating to *transfer-event*, such as the VIEW relating *transfer-event* to *taking*. Rather, *hug-transfer* seems an intermediate stage in the view between *hugging* and *hug-giving*. This type of short circuit may be illustrated by the following diagram:

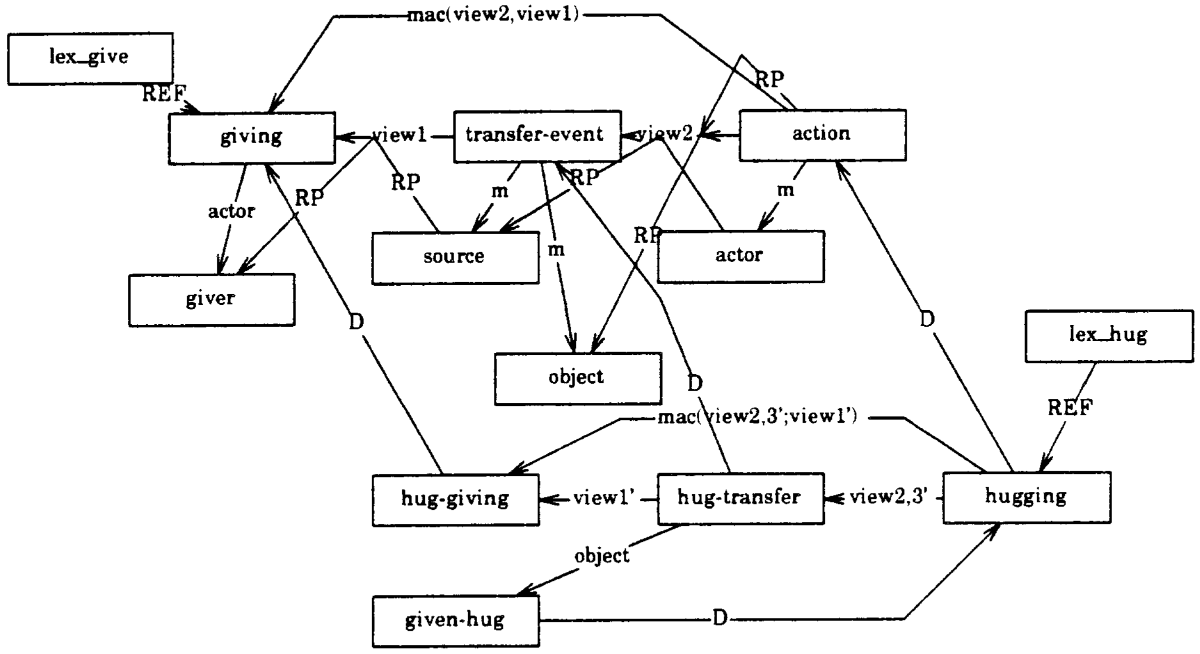

Figure 6-9.

The labels mac(view2, view1) and mac(view2,3'; view1') in the diagram above illustrate the macro-associations which link *action*s to *giving*s by connecting other structured associations. These macro-associations may be used to prefer particular sequences of structured associations in either analysis or generation; however, their application does not entirely bypass the individual VIEWs. The manner in which these macro-associations may be used in generation is discussed in the next two chapters.

The distinction was made earlier between highly constrained constructs such as "kick the bucket" and more flexible ones, such as "bury the hatchet". The representation given for the "bury the hatchet" expression was general enough to allow for phrases such as "burying a hatchet" and "the hatchet to be buried" to correspond to the specialized meaning. It would seem counter-intuitive to have no indication in the representation that "burying a hatchet" is somehow not as good a manifestation of the expression as "burying the hatchet". The preferability of "burying the hatchet" is probably not entirely attributable to its conceptual structure.

The problem of distinguishing "burying the hatchet" from "burying a hatchet" has several potential solutions. The most elegant would seem to be a general association which would connect constructs of a certain type with the definite article, so that "take a walk" and "give a hug" are actually distinguished from "bury the hatchet" and "take the cake". To represent the underlying motivation for the choice of article, however, seems a formidable task. A second possible solution would be to associate "bury the hatchet" with a linguistic structure similar to the "kick the bucket" pattern, which would serve as a default. This would prove somewhat inefficient, since the specific linguistic structure would still not serve to guide the choice of article in the passive or other unconventional use of the phrase. In this case, the sensible solution seems to be to attach to the *object* of the *hatchet-burying* concept an association with the noun phrase "the hatchet" in addition to the association with the lexical term "hatchet". This would suggest that under normal circumstances the definite article would be used, but if some other constraint were violated by this choice, the construct could still be realized.

6.5. Encoding New Knowledge in Ace

This thesis has posed extensibility and adaptability as critical issues in knowledge representation, but the discussion in this chapter and in the previous two chapters has only implicitly addressed the issue of adding new knowledge to Ace. I have made the argument that parsimony of representation is an indicator of extensibility, and that explicit referential knowledge and uniformity are indicators of adaptability. Many of the examples presented in these three chapters can be used to support this argument.

Most of the examples of events presented here have been related to the abstract concept of a *transfer-event*, and linguistic knowledge about referring to these events thus derives from more general knowledge. The *commercial-transaction* event, for example, can be decomposed into two *transfer-event*s, which in turn are joined by VIEW associations to actions such as *buying* and *selling*. While this introduces a fair amount of conceptual knowledge into the hierarchy, in keeping with the Proliferation of Conceptual Categories Principle, there is essentially no special linguistic knowledge required to realize all the forms in which descriptions of the *commercial-transaction* can appear. The dative form of the verbs, the prepositions used, and the constraints on what can be the subject, all derive from more general knowledge about *transfer-event*s, particularly the ROLE-PLAY relationships between elements of *transfer-event*s and elements of linguistic relations.

Similarly, the specialized knowledge about constructs such as "giving a hug" illustrates the ease with which knowledge about *transfer-event*s may be used to increase the linguistic knowledge of a system. By associating *hugging* with *hug-giving*, the Ace hierarchy represents the relationships realized in "John gave Mary a hug" and "Mary was given a hug by John" without any special linguistic patterns or relations. The addition of knowledge about "giving a kiss" likewise involves merely the encoding of two new VIEWs into the hierarchy, and, as will be shown later, enables KING to produce "Mary was given a kiss on the cheek by John" or "John kissed Mary on the cheek" without any special linguistic knowledge. Encoding knowledge about the concepts *kissing* and *punching*, along with lexical knowledge about "kiss" and "punch", enables KING to produce the following sentences:

John's being punched on the nose bothered Mary.
Mary's kiss on the cheek pleased John.
John wanted to be given a kiss on the cheek by Mary.

The Ace hierarchy was easily adapted to apply its knowledge about *transfer-event*s to the UNIX domain. Many concepts in the UNIX domain, for example, are viewed as *communication-transfer*s, or *transfer-event*s whose object is some type of message. These events are often referred to using verbs such as "tell", "give" and "get". The knowledge about *transfer-event*s allows these verbs to be used for *communication-transfer*s without any additional linguistic structures. Classifying certain UNIX-world concepts thus allowed the generator to produce:

> To get the name of the machine, type 'hostname'.
> Type 'write John' to send John a message.
> 'Ls' gives you the names of your files.

Multiple inheritance is an important aid in the encoding of new information into the Ace hierarchy. As an illustration, note that *object* is not an aspectual of *transfer-event* but of a higher-level concept in the system, *simple-event*. Thus, while *transfer-event*s have associated with them special knowledge about linguistic structures in which *source* and *recipient* may be realized, they do not have special knowledge about referring to their *object*s. Now, consider events in which the *object* is a command or desire. These are often referred to using verbs such as "order", "tell", "instruct", "want", "need", and "force". We may desire to have "John told Mary to go", "John asked Mary to go", and "John forced Mary to go" all make use of the same linguistic knowledge. This is accomplished by having the knowledge about the role of the desired event or state inherited from a different part of the hierarchy from knowledge about *transfer-event*s.

Consider the sentence "John ordered Mary to go". This sentence seems to describe an act of communication, but also reflects the compulsion of "needing Mary to go" or "wanting Mary to go". Such constructs seems to inherit some of their structure from means of describing communication, and some of their structure from means of describing compulsion. Acts of communication are subcategories of the abstract category *transfer-event*. This accounts for the linguistic structure of "John asked Mary a question" and "John told Mary a story". Needs and desires are categorized as *unrealized-predications*, or states which are unrealized at the time of an event. *Unrealized-predication*s are unrealized states of the *unrealized-subject*. The linguistic relation *verb-usub* relates the verb of a verb phrase to the constituent which refers to the *unrealized-subject* of the *unrealized-predication*. This relation associations "John" and "Mary" in "John wanted Mary to leave". The relation *verb-upred* associates the verb with the expression of the *unrealized-predicate*. This relates "John" and "to leave" in the above sentence. The hierarchy to which the *ordering* concept is associated is illustrated by the following figure:

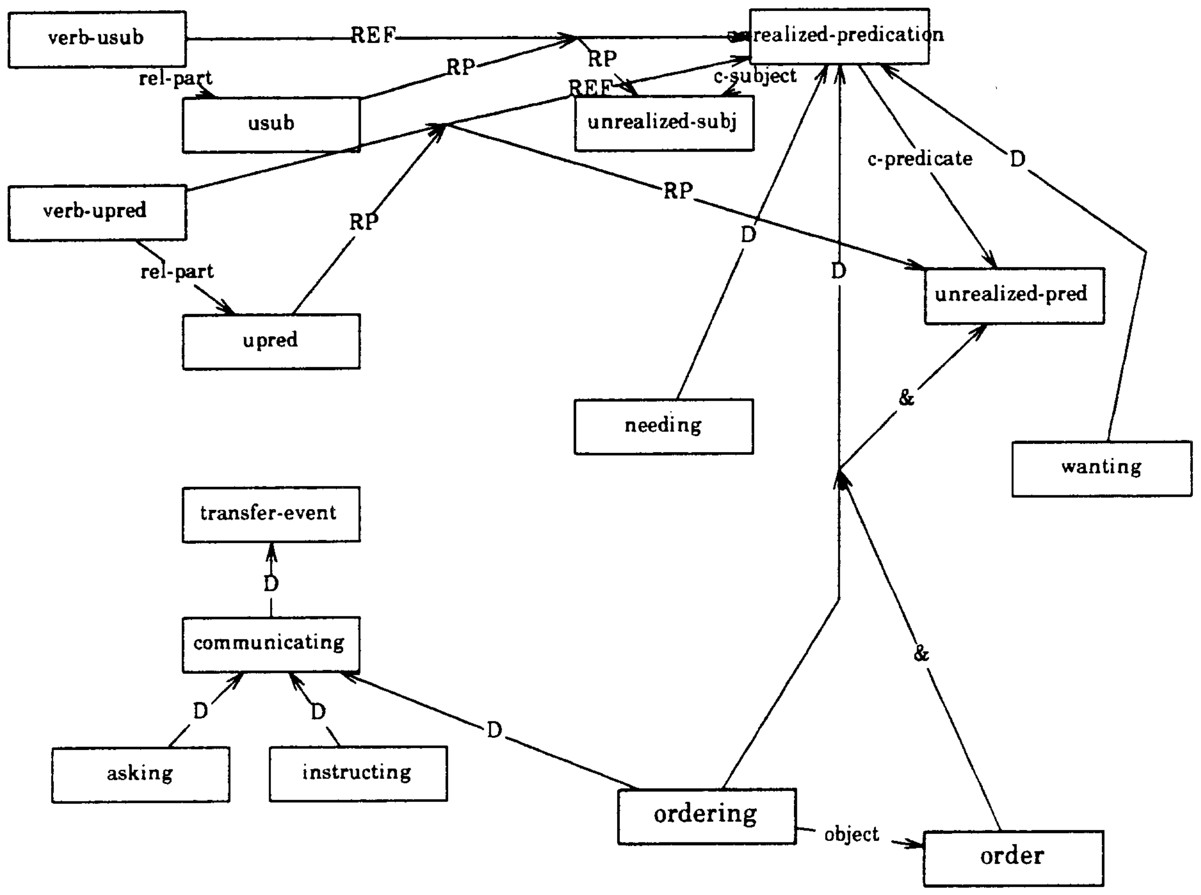

Figure 6-10.

In the above figure, the concept *ordering* is shown as belonging to both the *communicating* and *unrealized-predication* categories. By virtue of this knowledge, "John ordered Mary to leave" may be produced by combining knowledge about referring to *unrealized-predications* with knowledge about referring to *transfer-events*: The dative construct derives from knowledge about *transfer-events*, and the use of the infinitive phrase derives from knowledge about expressing *unrealized-predicates*, related to *verb-usub* and *verb-upred* but not shown above. "John told Mary to leave" may be used to describe an *ordering* event, without any additional knowledge about the verb phrase structures within which "tell" and "order" can be used.

The difference between ""John wanted to leave" and "John wanted Mary to leave" in Ace is not treated as a representational consideration. In one case, the subject of the *unrealized-predication* may be the same as the subject of the sentence. If this is true, a general processing rule in KING, to be considered in Chapter 7, dictates that each concept may play only one role in the linguistic relations realized. This rule presents complications in certain uses of pronouns, complications which still are difficult for KING. Nevertheless, it seems practical to handle EQUI- constructs such as these by using processing rules rather than representational distinctions.

Summary

This chapter has explored the use of a hierarchical, uniform knowledge representation in the encoding of associations between language and meaning. The examples presented are meant to illustrate how the Ace framework handles the representation of generalized and specialized relationships and of indirect associations between language and meaning. The structured association REF in Ace is used to represent structural relationships between linguistic and conceptual entities. By utilizing these associations at various levels of the Ace hierarchy, the representation provides a mechanism for making use of generalizations and indirect relationships in the encoding of specialized knowledge and thereby heightens the extensibility and adaptability of a knowledge base. The next two chapters will consider how this representation can be exploited in the generation of language.

7. Processing Aspects of Generation

The previous three chapters have presented the details of the Ace representation, designed to alleviate knowledge representation problems which limit the extensibility and adaptability of language processing systems. This chapter describes how this knowledge can be applied to the generation task. The processing framework described here is fundamental to the design of KING (Knowledge INtensive Generator), a system built to produce natural language output from a conceptual representation using a knowledge base in the Ace form.

In the spirit of PHRED, KING is built to share a declarative knowledge base with a language analyzer, and to produce linguistic output in real-time. KING is also geared toward facilitating the exploitation of new knowledge, to make adaptation and extension easier. While the problems with PHRED and other generation systems are primarily issues in knowledge representation, the design of the mechanism which organizes and applies the knowledge is an important consideration. I have suggested that knowledge-intensivity and incrementality are two important aspects of such a mechanism. Knowledge-intensivity is important because knowledge is easier to adapt than program. Incrementality is an asset because it helps to make the knowledge of the system more versatile, taking advantage of modularity of knowledge representation. The mechanism described here, and realized in KING, takes advantage of an incremental, knowledge-intensive means of exploiting the power of the Ace representation.

The process of constructing an utterance using this mechanism may be broken down into three basic phases, as described below:

Mapping is the process of retrieving and applying structured associations which relate the concept to be expressed into other conceptual structures and ultimately to linguistic structures. The result of the mapping process is a set of linguistic relations and constraints which should be realized in the structure produced. Mapping is primarily a data-driven component of the system, as the selection of structured associations for mapping depends mostly on the nature of the input concept.

Pattern selection is the task of accessing templates specifying linguistic patterns from the knowledge base of the system, and of selecting the pattern which best fits the input constraints and the structures derived from mapping. Pattern selection is the means by which knowledge structures derived from mapping may be combined. This process is more goal-directed than the mapping process, as it depends heavily on the linguistic entailments arrived at through mapping.

Restriction consists of applying a set of constraints to a selected pattern, thereby producing a pattern whose constituents are further specified using the constraints which have led to the selection of the pattern. The result of the restriction process is an instantiation of a linguistic pattern, representing a surface structure upon whose constituents the generation sequence is recursively invoked.

For example, consider the problem of producing a sentence such as "John sold Mary a book". The input to KING in this case is the conceptual representation of a particular *commercial-transaction* event. The mapping process results in the application of structured associations between *commercial-transaction* and *ct-merchandise-transfer* and between *ct-merchandise-transfer* and the *buying* or *selling* action, and identifies linguistic relations needed to express these concepts in a surface structure. These relations, obtained from REF associations with concepts DOMINATing *ct-merchandise-transfer* and *selling*, include *subject-predicate*, *verb-indir-relation*, and *verb-obj-relation*. Pattern selection results in the choice of a basic sentence pattern and a dative verb phrase, *i. e.*, the means for combining these relations in a sentence. Restriction results in the insertion of aspectuals of these relations in the appropriate places in the surface structure. For example, the *subject-predicate* relation used with the *Basic-S* pattern results in the restriction that the noun phrase of the pattern refer to the filler of the *subj* of the relation.

The sections which follow give an overview of each of the three phases of the generation process. Further details with respect to the implementation of each phase are presented in Chapter 8.

7.1. Mapping

The mapping phase of KING utilizes structured associations to produce new instances of knowledge structures from instances of other knowledge structures. In generation, this process can produce a variety of new structures from a conceptual input; for example, a new conceptual structure, a lexical category, a linguistic relation, or a linguistic constraint. These are derived from the mapping mechanism by traversing different types of structured associations. New conceptual structures may be produced by utilizing DOMINATE, MANIFEST, and VIEW associations. New linguistic structures are generally produced from REF associations.

In the *commercial-transaction* example, knowledge structures such as *ct-merchandise-transfer* and *selling* are accessed through MANIFEST and VIEW associations. That is, the mapping process uses the instantiated *commercial-transaction* concept to instantiate the *ct-merchandise-transfer* aspectual, using ROLE-PLAY relationships to determine the fillers of the roles of the *ct-merchandise-transfer*. ROLE-PLAYs similarly determine the roles of *selling* which are filled when the VIEW relation is applied between *ct-merchandise-transfer* and *selling*. This use of an aspectual of the complex *commercial-transaction* event to describe the event is due to the lack of any direct means of referring to the *commercial-transaction*.

REF associations are used to map into linguistic structures, including relations, constraints, and lexical categories. A lexical category, for example, the *lex_sell* category ("LEX = sell" in the feature notation), is a linguistic structure which is later used to produce a lexical item in KING. A constraint, for example, the *voice_active* constraint ("VOICE = active" in the feature notation), constrains either the selection of a linguistic pattern or the form of a lexical item. Also, relations, such as the *verb-obj-relation* and *verb-indir-relation* described in Chapters 5 and 6, are instantiated by mapping across REF structured associations.

When a VIEW association is applied in KING, the result is an instantiation of a new concept. The mapping process continues from the new concept, thereby initiating an indirect reference to the original concept. When a REF association is applied, the result is an instantiation of a linguistic structure. In this case, KING continues to attempt to apply REF associations from the same concept and then passes control to the pattern selection mechanism. No further VIEWs are used unless the process of building a linguistic pattern fails at this point. The mapping process thus applies VIEWs only until it reaches a point where it has derived enough referential knowledge for the building of the required pattern.

The discussion below considers the selection of structured associations in KING's mapping phase.

Choosing Structured Associations

The process of applying a structured association as a map is relatively straightforward: Given the instantiation of a concept and its aspectuals, a new concept is instantiated using the ROLE-PLAYs of the structured association to determine the fillers of the new aspectuals. The process of selecting *which* structured associations to apply, however, is a more difficult one, critical to the generation task. KING employs a strategy which replicates many of the heuristics embedded in other generation programs. This strategy may be summarized by the following principles:

Processing Principle 1. Favor "horizontal" associations over other associations.
The explicit referential associations recognized by KING are VIEW and REF. I refer to these as "horizontal" associations. Typically, these are explored before more "vertical" associations such as DOMINATE and MANIFEST. This is similar to the "choose the most specific match" rule obeyed by BABEL and PHRED: The verb "sell" is generally favored over the verb "give" for a *selling* concept because it is associated with *selling* via a REF, while "give" may only be reached by first following a DOMINATE link to *giving*. The justification for favoring VIEWs over DOMINATE and MANIFEST is that, when a direct means of expressing a concept is not found, the expression a metaphorically related concept is preferable in general to an expression of a concept which is too specific or too general.

Processing Principle 2. Favor "upward" associations over other vertical associations.
The "upward" associations in Ace are INSTANTIATE and DOMINATE. This principle means that in producing language one derives linguistic constructs using supercategories of the concept to be expressed, rather than attributes of the concept to be expressed. Thus indirect reference by supercategory is generally favored over reference by component.

Processing Principle 3. Favor associations which yield linguistically appropriate structures.
The generator prefers associations which produce linguistic structures, but it is also biased towards those which produce linguistic structures which are likely to be useful. REF associations are favored over all other structured associations, provided that the linguistic structures which they produce are usable. For example, actions are often referred to using verb phrases, but in some contexts they produce nouns or

modifiers. The generator should then be biased towards the appropriate structure. The phrases "John's selling the book", "John sold the book", and "the selling of the book by John", may all result from the same conceptual structure but depend on the structural constraints of the local context, including the linguistic constraints which propagate from linguistic structures already selected. The structured mappings which produce the possessive construct and the definite reference should not be used when constructing a complete sentence, and the *subject-predicate* relation should not be considered when constructing a noun or noun phrase.

The first two principles illustrate some of the processing aspects of the interaction of specialized and generalized knowledge in KING. The idea of *preemption*, favoring specialized knowledge over more general knowledge, is behind the first principle. Because structured associations in Ace may inherit their structure from more abstract associations, this specialized knowledge may depend on more general knowledge, but it may also override or *preempt* more general knowledge. The second principle specifies that where there is no specialized knowledge about referring to a particular concept, it is acceptable to apply knowledge about a more general concept. This principle would apply, for example, in referring to the *ct-tender-transfer* concept: The verb "pay" refers to a concept which DOMINATEs *ct-tender-transfer*, as "paying" is associated with the concept with paying for anything, while *ct-tender-transfer* as used here designates an event performed in exchange for *merchandise*.

The third principle keeps the process of applying maps from producing structures which will not be useful in constructing the required phrase. This applies mainly to structured associations which yield linguistic relations. The use of these relations may be subject to a variety of constraints. For example, the *verb-obj-relation* and *verb-indir-relation* relations may be realized only in *verb-phrase* constructs; if some other construct is being produced it is necessary to refer to recipient and conceptual *object* using other relations, as in "the sale of the book to Mary". This principle has the following corollary:

Corollary A. Avoid redundant and semantically inconsistent expressions. Structured associations may not be applied which involve more than one concept in the same role or involve the same concept in multiple roles. For example, "John kissed Mary's cheek" realizes the role of cheek as a conceptual *object*, but if "Mary" has been used in that role the generator should find another relation to express the cheek's role, as in "John kissed Mary *on the cheek*." This corollary also avoids building sentences such as "John was kissed the cheek" and "Mary kissed John's cheek on the cheek".

Like PHRED, KING is designed to incorporate as much as possible into *predisposition* mechanisms. The heuristics above predispose the generator towards applying certain structured associations and thus producing particular structures preferentially. These predispositions may influence the language produced. Even where the the heuristics are neutral with respect to some set of structured associations, the generator selects a structured association to apply, randomly if necessary, and continues to generate using that association. This is the case, for example, with the *commercial-*

transaction example. The generator will not produce a structure for "John sold Mary a book", another for "Mary bought a book from John", and evaluate the structural choices. Rather, it will map from *commercial-transaction* to *ct-merchandise-transfer* and then to either *buying* or *selling*, and will complete its work based on the result. In this way the biasing of the mapping process has a substantial effect on the output of the generator. This is similar to predisposition in PHRED and to the indelibility realized in MUMBLE, but is generalized to conceptual structures: A VIEW of a conceptual structure, once applied by KING, is not evaluated with respect to other VIEWs. If the generator applies a VIEW to obtain the *buying* concept from a *ct-merchandise-transfer*, it will not consider a *selling* VIEW.

Macro-associations in Mapping

Macro-associations have a special role in the process of selecting and using associations for mapping. The macro-association in KING allows a knowledge structure to affect mappings from subsequent knowledge structures. These associations effectively constitute short cuts that the generator can use in determining which mappings to apply, by connecting a series of mappings.

As described in Chapter 6, a macro-association in Ace specifies a sequence of structured associations which may be applied to produce a new structure from a given structure. This helps the generator to avoid considering related but unessential knowledge structures. For example, if *m1* represents the structured association between *commercial-transaction* and *ct-merchandise-transfer*, and *v1* represents the VIEW relation between *ct-merchandise-transfer* and *selling*, then *mac(m1,v1)* can be used to designate a macro-association between *commercial-transaction* and *selling*. This macro-association may be used to map to the *selling* concept without considering the mappings which would instantiate *ct-tender-transfer* and *buying*. This suggests the following rule:

> *Corollary B. Favor macro-associations over other structured associations, except where Principle 3 applies.*
> Macro-associations are considered before other structured associations in the mapping phase, assuming that no association directly produces a valid linguistic structure. Thus *mac(m1,v1)* above in the absence of other macro-associations would cause KING to refer to a *selling* concept unless some constraint were violated.

The two differences between applying a macro-association and applying a sequence of associations independently are (1) At any stage in the application of a macro-association, the next association in the sequence is considered before any other associations, and (2) If the application of an association in the sequence specified by a macro-association fails, the mapping process resumes where the macro-association was initiated, instead of at the last concept instantiated.

There are two important ways in which macro-associations differ from simple short cuts. First, the macro-associations may be biased according to their intermediate mappings. This stipulation is to allow, theoretically, for the preference of forms such as "John gave Mary a kiss in exchange for a favor" to "John kissed Mary in exchange for a favor". Since a macro-association corresponds to a sequence of mappings, rather than a single association, at any stage in the mapping process the macro-association may be used or abandoned based on context and subject to the application of constraints which affect the individual mappings. Such contextual biases are not, however, implemented in KING.

In the "giving a kiss" example, suppose *mac(v10,v11)* denotes a macro-association between *kissing* and *giving* through *transfer-event*. This macro-association generally results in the instantiation of the *giving* concept. In other cases, the choice of a macro-association is more likely to be influenced by constraints on the individual mappings, as is the case, for example, with "taking the news", which is subject to the constraint that the news must be negative. This constraint is not particular to the macro-association which relates the *reacting* concept to *transfer-event* and the lexical terms; it applies to most associations which link *transfer-event* to *taking* where the *taker* is not an underlying *actor*. Thus if *mac(v20,v21)* links *reacting* to *taking*, the application of *v21* between *transfer-event* and *taking* is subject to the usual constraints. If a constraint on this mapping fails, the generator continues mapping not from *transfer-event*, but from *reacting*.

The second major complication in the use of macro-associations is in their interaction with other associations and macro-associations. A macro-association may link a concept to other concepts by specifying a general sequence of associations. Suppose that *m0* is the relation between *complex-event* and *simple-event*, and *v0* is the VIEW association between *simple-event* and *action*. Then *mac(m0, v0)* is a macro-association which links *complex-event* to *action*. This macro-association biases the generator toward instantiating an *action* concept, but the choice between associations with *buying* and *selling* is not predetermined. The mechanism thus consistently favors the use of more specific structured associations where possible. The general macro-associations determine the *type* of structure to be produced, e. g., *action*, while the specific associations determine the particular structure, e. g., *buying* or *selling*.

The principles presented here are implemented in the mapping mechanism of KING, to be described in section 8.3. The same mechanism is used to apply REF associations as is used for VIEW associations. Each time such REF associations are used, the mapping process continues to apply REF associations from the last concept instantiated, keeping track of all the linguistic structures it has derived. The result of the mapping process is a set of constraints and linguistic relations which should be realized in the part of the utterance being produced. Mapping from the concept *(selling (object book))*, for example, produces an instantiated *verb-obj-relation* along with the lexical category *lex_sell*. Chapter 8 provides further examples of how these relations and constraints are derived and used.

The set of linguistic structures resulting from the mapping process is passed to the pattern selection phases, which is described in the next section.

7.2. Pattern Selection

The hierarchical organization of linguistic patterns makes a simple method of pattern selection in KING very effective. As in PHRED, potential patterns are returned by the generator in an ordered stream, biased by the constraints to be satisfied and the relations to be expressed by the pattern. The order in which patterns are retrieved and considered may influence the language produced. In KING, however, the bulk of the specialized linguistic knowledge is stored in the hierarchy of relations, while the patterns themselves are used to determine constituent order. There are thus far fewer patterns to be considered by the system. While neither PHRED nor KING has been extended to have a very large grammar, the linguistic coverage of KING is unquestionably better than PHRED's, although KING's knowledge base has roughly 50 patterns and the English knowledge base of PHRED has several hundred.

Pattern selection has a substantial top-down element, stemming from the fact that the applicability of a pattern depends heavily on the structure within which it is used. An example given earlier is that a complement using "that" must be followed by a basic sentence pattern. In this case, the pattern selection process is easy because only the basic sentence pattern need be considered. In contrast, the production of noun-phrases and verb-phrases is constrained less by where they appear and more by the concepts which are being expressed.

The pattern selection mechanism has three potential outcomes: If its input specifies a lexical category, it may produce a lexeme. Otherwise, it can produce a template, including a pattern which satisfies the necessary constraints. Otherwise, it will fail, generally passing control back to the mapping mechanism.

The following principles describe the process of selecting patterns:

Processing Principle 4. Consider only those patterns which fall in the most specific pattern category.
The input to the pattern selection phase may include pattern categories such as noun phrase or verb phrase. Such categories are always in the pattern which the generator is working from; for example, if KING is in the process of using an NP-VP pattern, the first constituent must be a noun phrase and the second a verb phrase. More specific categories may be given, such as question-sentence, postnominal-modifier, or basic-sentence. These categories may also be derived in the mapping process. The linguistic pattern being produced must necessarily belong to the most specific of these categories.

Processing Principle 5. Favor patterns which are associated with contextual structures.
This principle distinguishes contextual biases from strict constraints. An element of the context may be used to derive a pattern category, but the generator can if necessary select a pattern not in that category.

Processing Principle 6. Favor patterns which subsume as many relations as possible.
Among the candidates of equal status from Principles 4 and 5, patterns may be ordered according to the number of linguistic relations which they express. Patterns which express relations not derived from the mapping process are discarded.

Principles 4 and 5 above are relatively simple. The fourth principle dictates that only patterns which satisfy external constraints should be considered. The fifth dictates that contextual knowledge should be used where possible. Very few contextual biases are actually implemented in KING, but the mechanism is designed to allow for a greater contextual influence. One contextual phenomenon that has been encoded is that if part of the context in which KING operates is that the system is instructing its hearer, this biases the generator toward imperative sentences. If part of the situational context is the *be-polite* objective, the system may be biased towards using questions. The representation of objectives such as *instruct* and *be-polite*, as it stands, however, does not capture much of their meaning.

Principle 6 is difficult to implement. One problem is that it is not easy to determine what relations can ultimately be expressed in a pattern. Some noun phrase patterns, for example, include verb phrase constructs, and thus may express relations such as *verb-obj-relation* and *verb-indir-relation*. In this case it is important to have a means of determining which noun phrase pattern will allow the relations to be expressed, to facilitate the generation of phrases such as "selling Mary the book". This problem is further detailed in Chapter 8.

Another caveat to Principle 6 is that in certain cases it does not seem sensible to rely on the data-driven mapping process to provide all the relations necessary to realize a pattern. This seems true with do-support, the use of the auxiliary "do" in constructs such as "Did John go?" and "John did not go". The *aux-verb* relation with modal "do" should be instantiated if and only if the inverted question or negative pattern is selected and no other *aux-verb* relations have been instantiated. Otherwise, the generator might not be able to produce "John *did* not go" or might erroneously produce "John did not be kissed by Mary". To avoid this, the inverted question and negative pattern thus have associated with them a default *aux-verb* relation, specifying the auxiliary "do". The lack of a derived *aux-verb* relation thus does not prevent these patterns from being selected, but causes the default to be passed to the restriction phase.

As an example of the pattern selection process, consider the production of the verb phrase part of "John sold Mary a book." The generator will at this point have obtained a variety of constraints such as "FORM = finite" and "VOICE = active", and will have determined from the Basic-S pattern that it must produce a *verb-phrase* referring to a particular selling concept. The mapping process, furthermore, will have produced the relations *verb-indir-relation (iobj = mary1)* and *verb-obj-relation (obj = book1)*.

As suggested by Principle 4, the pattern selection mechanism first retrieves all templates with patterns in the *verb-phrase* category (about a dozen in the current knowledge base). It then orders these templates according to the number of derived relations which they express. The *basic-vp* of figure 5-6, for example, is considered before the *intrans-vp* pattern, because it expresses the *verb-obj-relation*. In this case, KING chooses the pattern which subsumes the *verb-indir-relation* and *verb-obj-relation*, the dative verb phrase pattern*:

* This pattern has been slightly simplified here to make the example clearer. As presented in Chapter 5, the indirect object part of the dative verb phrase should be treated as part of an embedded *indir-vp* constituent. This does not influence the example.

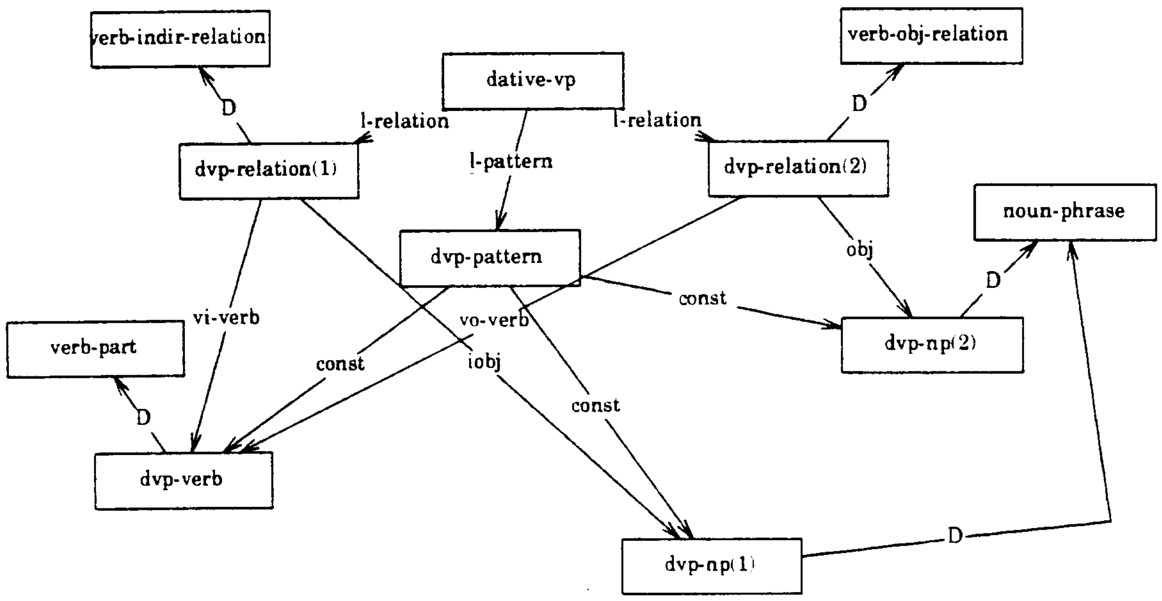

Figure 7-1.

Each time a pattern is selected by the generator, it is passed along to the restriction mechanism, which performs the bulk of KING's linguistic grunt work. This mechanism is described in the following section.

7.3. Restriction

Restriction is the process of applying constraints to a selected pattern, matching it with the relations it is to subsume and preparing the constituents of the pattern for completion. This role is often played by unification in systems such as those of Appelt (1983) and McKeown (1982). In PHRED the restriction process was divided into three components: *unification*, which matched the attributes of the input to the features of a pattern template, *elaboration*, which added constraints to individual constituents of the pattern, and *combination*, used to combine ordering patterns with flexible-order patterns. KING represents a step away from traditional unification, performing no syntactic unification at all. There are a variety of reasons for this elimination, the most significant of which is the goal of replacing an element of the generation process which can be both time-consuming and counter-intuitive: the use of explicit variables to represent associations between knowledge structures.

In unification-based systems, variables represent two basic kinds of associations: (1) the relationship between a linguistic attribute and a conceptual attribute, such as the "PREP-OBJ = <agent>" specification in the passive template in section 5.1, and (2) the relationship between a feature of one linguistic structure and that of another linguistic structure, such as the "TENSE = ^ TENSE" specification in the unification grammar example of Chapter 2. Both of these types of associations are represented by ROLE-PLAYs in the Ace hierarchy. In unification grammars, these variables serve two functions: (1) the matching of pattern attributes to input attributes, for example, to produce "PREP-OBJ = john1", and (2) the addition of features to individual constituents of the pattern, for example, to specify the tense of the verb of a pattern based on the constrained tense. Both functions are accomplished in KING by using REFs with ROLE-PLAYs instead of explicit variables. The first function is carried out by the application of REF associations in the mapping process. The second is carried out by the use of ROLE-PLAYs between patterns and pattern constituents in the restriction phase.

The restriction process in KING consists of stepping through the given relations and constraints. For each relation or constraint which has been produced through the mapping process, conceptual structures or constraints are added to the appropriate pattern constituent. For each relation, this means looking for a relation type which DOMINATEs the relation and is associated with the template containing the selected pattern. The appropriate conceptual element from each aspectual of the relation is then added to the corresponding constituent. For all linguistic structures derived through mapping, explicit ROLE-PLAYs are checked to add linguistic structures wherever necessary to pattern constituents. This may include constraints which are particular to the given relation. For example, the *passive-by-adjunct* relation has a constraint that the preposition of the adjunct is "by".

Restriction in KING compacts the unification, elaboration, and combination components of PHRED into a simpler mechanism. Unification is not applied at all: Simple checks are used to be sure that the selected pattern does not violate constraints; however, as in PHRED, the system depends on the pattern selection mechanism to ensure that constraints are satisfied. The elaboration process is essentially the same as that in PHRED, except that the hierarchical organization of linguistic structures allows the correspondences between pattern attributes and constituent attributes to be inherited, and also makes it easier to specify classes of linguistic constraints which correspond to certain constituents. For example, attributes such as person, number, tense, and form are used to specify constraints on verbs, and they may be grouped into a common category, *verb_constraint*. Associated with verb-phrase constructs, then, is a ROLE-PLAY which indicates that all *verb_constraints* are constraints on the *verb-part* constituent of the verb phrase construct. This knowledge is then inherited by all verb phrase constructs. This accounts for the omission of this information from the templates presented here. In other systems, the correspondence between each constraint of the verb and each constraint of the verb phrase requires an additional feature, such as "TENSE = ` TENSE".

Unification is unnecessary in KING because the knowledge contained in the templates of most linguistic representations is broken down into less complex, and more modular, structured associations. The nature of the knowledge representation is such that large-scale template-matching is unnecessary. Associations are applied based on the category classification of the concept to be expressed, and the output structure is built incrementally from the results. In unification-based systems, template-matching is necessary to perform such operations as the binding of conceptual attributes in a template. The "<person> <give> <person> <physob>" pattern in PHRED, for example, had associated with it a conceptual template with variables corresponding to each of the three main constituents, the two <person> constituents and the <physob>. A "RECIPIENT = (value 3)" binding explicitly indicated that the binding of RECIPIENT corresponded to the third consitutent of the pattern.

In KING, this binding is replaced by the mapping process, where ROLE-PLAYs effectively specify the relations among aspectuals which are equivalent to variable binding. A ROLE-PLAY between *iobj* and *recipient* takes the place of this specific binding and applies to all dative forms. Similarly, ROLE-PLAYs within the templates replace explicit variables used for elaboration: The correspondences between features of patterns and features of constituents, such as the TENSE feature discussed earlier, are indicated by the ROLE-PLAY relation. These correpondences also apply across a range of templates.

The result of the restriction of the dative verb phrase pattern given earlier might be the following:

```
First constituent:  V-P
     Concept    --    selling
     Constraints --   lex sell
                      voice active
                      form finite
                      person third
                      number singular
     Category   --    verb part

Second constituent:  NP (1)
     Concept    --    mary1
     Constraints --   case objective
     Category   --    noun phrase

Third constituent:  NP (2)
     Concept    --    book1
     Constraints --   case objective
     Category   --    noun phrase
```

The *lex sell* constraint on the verb part, produced by mapping across the REF association between the *selling* concept and the corresponding lexical category, is assigned to the first constituent by virtue of the knowledge that all *verb_constraints* on verb phrases are assigned to their verb parts. The *voice active* constraint derives in much the same manner. The *form finite* constraint is an inherent constraint on the *finite-verb-phrase* category, which is the subclass of verb phrase constructs with finite verbs. The *person* and *number* constraints are passed along from earlier stages. After each constituent is produced, the generator fills in the appropriate constraints on the dependent constituents. The case constraints on the noun phrases derive from constraints on the object and indirect object of the relations.

The restriction procedure, after it has completed the specification of each of the constituents of a pattern, reinvokes the generator successively on each constituent. As with PHRED, the order in which constituents are produced is generally left-to-right; however, it is possible to specify an order for a pattern or class of patterns. In English, this means that head nouns are produced before modifiers and subjects before auxiliaries in inverted constructs. This guarantees that agreement between constituents can be easily enforced.

The output of the restriction process, as shown above, consists of a set of constituents for which concept, constraints, and category are specified.

These serve as input to the generator for further refinement of the structure. The generation process thus proceeds by recursively invoking the mapping - pattern selection - restriction sequence, starting, in this case, with the verb part constituent. The control flow of the program through a complete example is traced in section 8.6.

Summary

This chapter has presented a high-level description of the processing aspects of generation as implemented in KING. The three-stage generation process is geared toward making use of the power and modularity of the Ace representation, as well as facilitating the interaction between abstract and specialized knowledge as introduced in Chapters 3-6. Chapter 8 considers the implementation and execution of the generator in greater detail.

8. Implementation of KING

KING is part of a new, evolving version of the UNIX Consultant system. Some of the details of the KING implementation have come about in a somewhat haphazard manner, and thus are separated here from the more theoretical issues discussed in the previous chapter. This discussion is provided to give a clearer picture of the internal workings of the generator.

8.1. Knowledge Manipulation Tools

The implementation of the Ace framework used by KING provides many of the same facilities as other knowledge representation languages such as FRL (Roberts and Goldstein, 1977), KL-ONE (Brachman et. al., 1979), and PEARL (Deering, Faletti and Wilensky, 1981). The basic set of knowledge manipulation tools was adapted from an implementation of a subset of PEARL, modified to incorporate the KODIAK structured associations and multiple inheritance, and extended to include the Ace VIEW and REF associations. The most relevant aspects of this implementation are the following:

Pseudonyms.
Pseudonyms are used by the generator to access attributes through ROLE-PLAY relations. In most representations such as those mentioned above, the filler of an attribute "slot" of a structure is accessed by referring to the name of the attribute. In the Ace implementation, these fillers often play a number of roles, and may be accessed by any of the roles which they play, either directly or indirectly. For each concept and attribute, Ace keeps a list of *pseudonyms*, or correspondences between names of attributes and names of their roles. For example, the *actor* attribute of the *selling* concept corresponds to the pseudonym *seller*, since the *seller* plays the role of *actor*. The *participant* aspectual might also be used to designate a *seller*, because *seller* indirectly plays the role of *participant.*

Inherited Properties.
As in most knowledge representations, attributes of concepts in Ace are inherited by DOMINATEd concepts. This includes ROLE-PLAY relations such as the correspondence between the tense of a sentence and the tense of a verb phrase, which is inherited by all sentence templates. The Ace implementation stores all these inherited relations on property lists associated with the names of concepts. Names of concepts which DOMINATE a given concept are also stored on these property lists, so all inherited attributes may be easily accessed.

Structured associations.
ROLE-PLAYs attached to all structured associations in Ace are implemented using pseudonyms, but are applied differently according to the association to which they are linked. For example, pseudonyms attached to DOMINATE links are *always* used when instantiating a new attribute. Thus when the *actor* of a *selling* action is instantiated, the *seller* role of the *selling* action is automatically filled. With horizontal associations, such as VIEW, pseudonyms are used only when

the horizontal association is explicitly applied. Thus the *seller* role is not automatically filled when the *merchant* of a *ct-merchandise-transfer* event is instantiated, but is filled when the VIEW between *ct-merchandise-transfer* and *selling* is applied. The *tense* role of a verb phrase likewise is not filled until both the *tense* of the sentence and the verb phrase component of the sentence are instantiated. The general rule, thus, is the following: *If a structured association is applied between A and B, and attribute A' of A is instantiated, then instantiate the corresponding attribute B' of B.*

These aspects of knowledge manipulation are exploited during each of the three main phases of the generation process. For example, in mapping, KING instantiates a new concept by applying a structured association. If this association has a ROLE-PLAY attached to it in which a role is played by an instantiated attribute of the mapped concept, it then must instantiate the corresponding role of the new concept. Since this role may be a pseudonym for other roles, it must also instantiate these other roles.

In the restriction process, new attributes are added to constituents of a pattern. These attributes may be added by applying correspondences, or ROLE-PLAYs, between elements of instantiated linguistic relations and the pattern being restricted. This correspondence may result in filling in a noun phrase component of a dative verb phrase pattern with the corresponding *iobj* element of the *verb-indir-relation*. The instantiation of this dative verb phrase with the attribute "TENSE = past" results in the application of the ROLE-PLAY between *tense* of *verb-phrase* and *tense* of *vp-verb* of *verb-phrase*, which results in the addition of the "TENSE = past" feature to the *vp-verb*.

The following sections will describe the Ace knowledge which is used in generation, and how this knowledge is applied during the mapping, pattern selection, and restriction phases in KING.

8.2. What KING starts from

The main input to KING is an instantiated Ace concept, with additional specifications of input constraints and of the pattern type to be produced. For example, the LISP function call *(KING hugging1 nil s)* would represent a typical request for KING. The first argument, *hugging1*, represents the concept to be expressed; the second designates the list of input constraints, and the third, *s*, the category of the pattern to be generated. The concept *hugging1* might have attributes attached to it, such as in the following diagram:

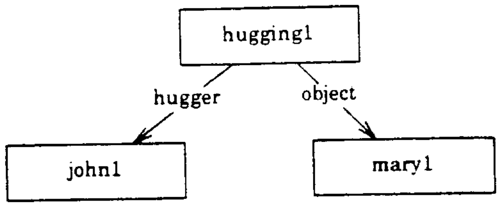

Figure 8-1.

This diagram represents basically what KING is generating from; however, *hugging1* is an instance of the *hugging* concept, and through this instantiation is connected to the network given in figure 6-8, as shown below:

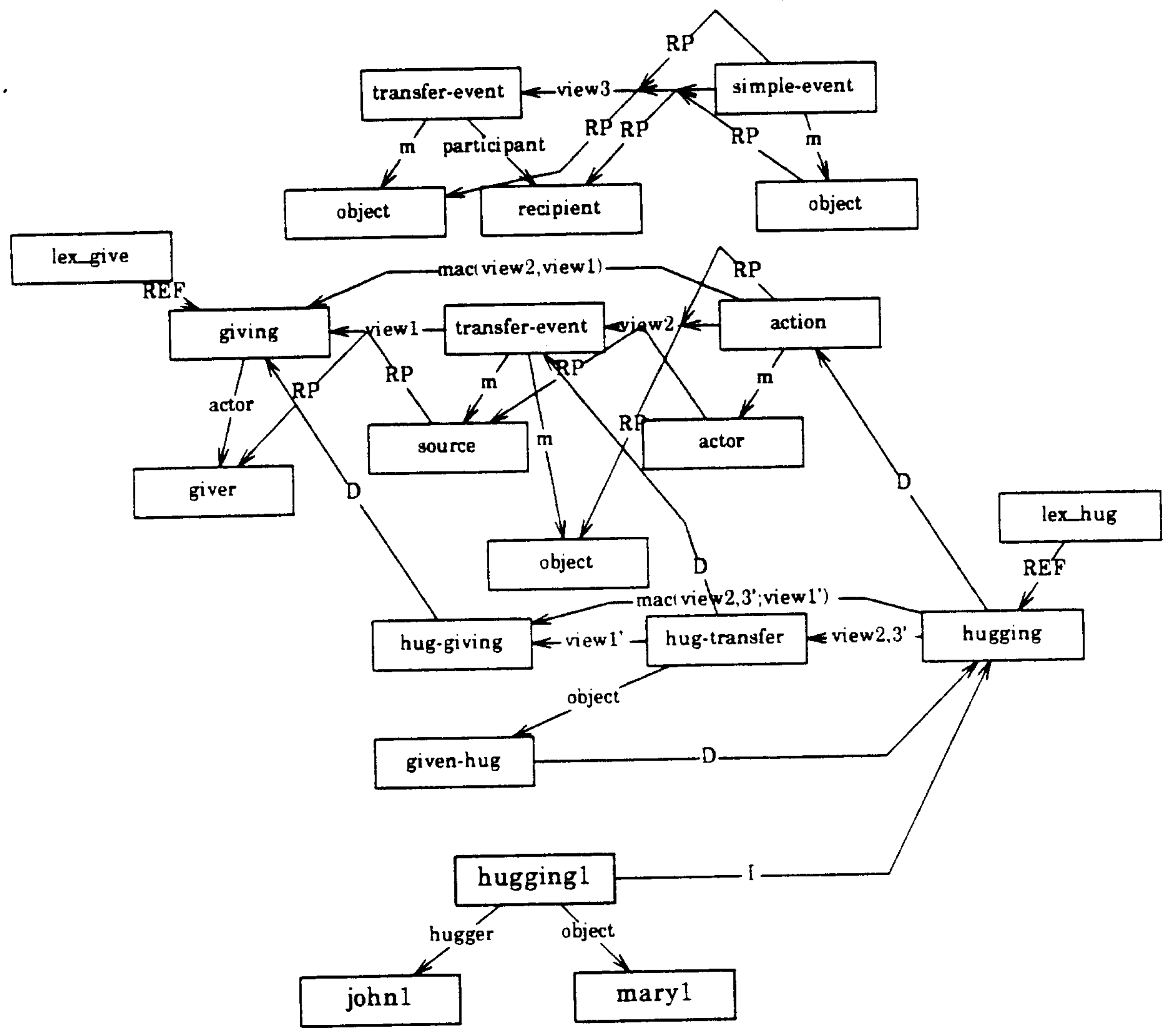

Figure 8-2.

The diagram above gives the essential fragment of the conceptual knowledge which is related to the *hugging* concept to be expressed. This example will be used in the next three sections to consider the knowledge manipulated during the generation process.

8.3. Mapping

The mapping process starts from the concept INSTANTIATEd by the concept being generated from, and continues generally upward in the network if successful maps are not found. At each concept considered, KING may attempt to traverse macro-associations and horizontal associations. The selection of which map to apply is carried out using the heuristics described in Chapter 7; that is, macro-associations and VIEW or REF associations are applied first. If the upward search through the network does not result in mapping to a new concept or the derivation of sufficient linguistic information to realize a surface structure, then MANIFEST associations are used. This can result in a metonymical reference, such as a reference to a complex object or event by referring to a part of the object or event, as in the *commercial-transaction* example.

When a structured association is selected, the mapping process applies each ROLE-PLAY relation attached to the association to instantiate roles of the new concept being created. This may result in the application of other ROLE-PLAYs as well to determine pseudonyms for the new attribute. If this process is applied successfully to produce a new concept, this new concept becomes the starting point for future maps. If the last map applied was an element of a macro-association, the first map considered with respect to this new concept will be the continuation of that macro-association.

The mapping process terminates when a set of linguistic relations applicable to the desired pattern category has been produced. The relations and constraints derived from mapping are then passed to the pattern selection mechanism.

Mapping into Conceptual Structures

In cases where objects and events are referred to indirectly, the mapping process produces new instances of conceptual structures as well as linguistic relations. In the "giving a hug" example, because of the preference of macro-associations, the first association applied is the macro-association between *hugging* and *hug-giving* at the bottom of figure 8-2. Were this macro-association not present, the generator at this point would map directly into the *predication* relation and produce the sentence "John hugged Mary". In this case, however, the first mapping applied will be the first part of the macro-association, *view2,3'*, which has attached to it the following relations:

(ROLE-PLAY *actor source*)
(ROLE-PLAY *action object*)
(ROLE-PLAY *object recipient*)

These ROLE-PLAY relations are inherited from *view2* and *view3* of figure 8-2, and thus do not have associated with them the most specific pseudonyms for *actor*. The mapping process thus obtains the *actor* of *hugging1* by looking at the pseudonyms for *hugging*. This yields the association (ROLE-PLAY *actor hugger*), which leads to *john1*.

ROLE-PLAYs which link absolute concepts to roles, such as (ROLE-PLAY *action object*) in this case, are applied only after ROLE-PLAYs are applied to their aspectuals. This allows the generator to enforce the stipulation that aspectuals may each be mapped only once. Thus, when the association (ROLE-PLAY *action object*) is applied to *hugging1*, *john1* and *mary1* already fill roles in the new *hug-transfer* concept. The *object* role is thus filled by the *hugging1* concept without its associated attributes.

The result of applying *view2,3'* to *hugging1* is thus the following structure:

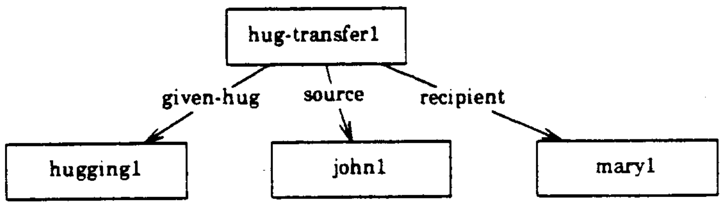

Figure 8-3.

The *hug-transfer1* concept above is produced by applying the first structured association in the macro-association *mac(view2,3'; view1')*, which is the *view2,3'* association. The next map tried, therefore, is the *view1'* VIEW, which has attached to it the correspondence (ROLE-PLAY *source giver*). The application of this map builds a new concept *hug-giving1*, and first applies this ROLE-PLAY to fill the *giver* role with *john1*. This exhausts the ROLE-PLAY relations attached to *view1*. However, the concept being produced is a *giving* action, and *giving* actions *are transfer-events* as well as being VIEWs of *transfer-events*. In such cases, where the structure being produced through the application of a VIEW is also DOMINATEd by the concept INSTANTIATEd by the original structure, property inheritance dictates that all attributes of the original structure should be preserved. In general, the mapping process implicitly maps all roles which are applicable to the new structure. The *hug-giving1* structure is thus the following:

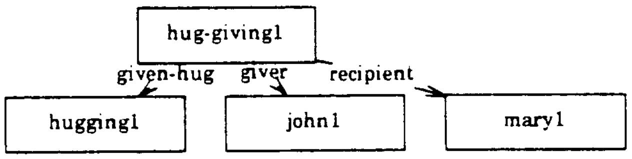

Figure 8-4.

Mapping into Linguistic Structures

When macro-associations have been exhausted and the mapping process has produced a conceptual structure from which REF links can be applied to instantiate linguistic structures, it applies all such relevant mappings to yield a set of applicable relations. The application of REF associations is exactly the same as that of VIEWs, except that no maps are attempted which originate from the resulting structures.

When the generation of the "giving a hug" example reaches the stage of producing a verb phrase, it is working from the following knowledge network:

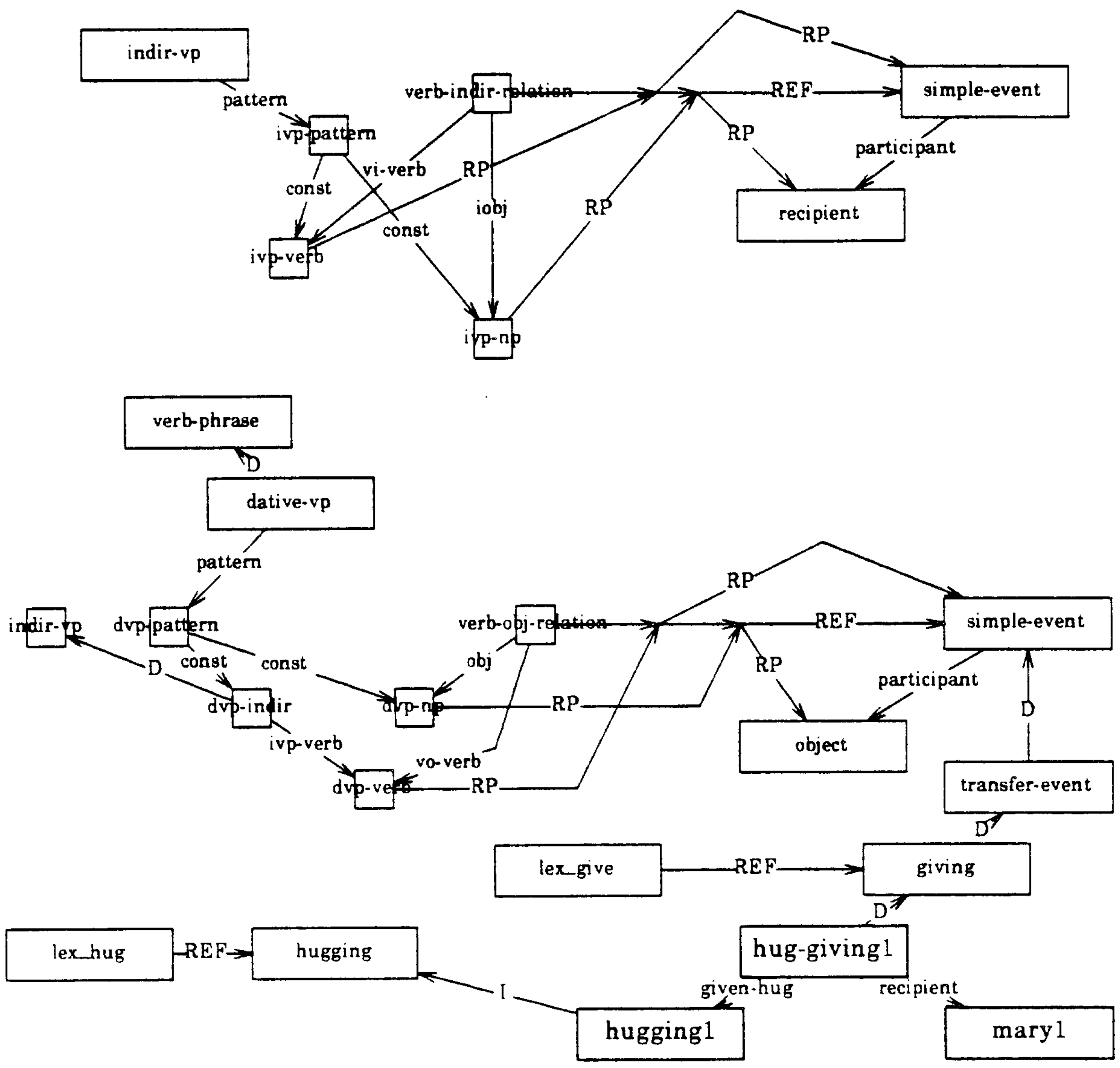

Figure 8-5.

The application of structured associations in the above network proceeds by applying REF links at successively higher levels. The first such REF association produces *lex_give*, the lexical category associated with *hug-giving*. Further up in the hierarchy, the *transfer-event* concept is reached. This results in the instantiation of the *verb-indir-relation* with indirect object referring to *mary1*, and verb corresponding to *hug-giving1*. Still further up the hierarchy, the REF association between *simple-event* and the *verb-obj-relation* establishes *hugging1* as the *obj* or direct object referent in this relation.

The output of the mapping phase at this point is the following set of constraints and relations*:

* The constraint *voice_active* is produced via a REF association from the *giver* role, as described in Chapter 6.

(lex_give)
(voice_active)
(verb-indir-relation (iobj mary1))
(verb-obj-relation (obj hugging1))

The mapping process thus produces a set of instantiated linguistic relations, lexical categories, and constraints, which can potentially be satisfied by a variety of surface structures. The pattern selection phase, considered below, determines how the linguistic knowledge derived from mapping can be combined into a grammatical utterance.

8.4. Pattern Selection

The two inputs to the pattern selection mechanism in KING are a set of linguistic structures and the category of the pattern to be produced. These linguistic structures may have been produced by the mapping process or passed along from earlier stages of generation. As described in Chapter 7, the main objective in pattern selection is to choose a pattern of the correct type which satisfies the necessary linguistic constraints and realizes as many linguistic relations as possible.

The basic scheme of KING's pattern selection mechanism is to retrieve all patterns of the correct type and apply a small set of rules to order them. Patterns which violate specified constraints are discarded, as are patterns which involve relations which have not been specified. Thus, in the production of a stative sentence, such as "John became ill", verb phrase patterns which involve transitive constructs are immediately eliminated.

The mapping process sometimes derives patterns, in which case these patterns are given preferential treatment to those retrieved by considering the most specific absolute type. One example in which this type of pattern selection takes place is in certain specialized constructs, where a pattern applies specifically to a given concept, such as in the "kick the bucket" expression. Such patterns are not indexed by the pattern type to which they belong, and thus can *only* be used if the mapping mechanism produces them.

For each template, such as *dative-up*, the Ace implementation maintains a list of DOMINATors, such as *verb-phrase*. For each template category, the knowledge manipulation functions maintain a list of templates containing patterns in that category. Processing Principle 4 of Chapter 7, which suggests considering only patterns which fall into the designated category, is thus enforced by retrieving a set of patterns DOMINATEd by that category.

The ordering of these patterns according to Principle 6, *i. e.*, determining which patterns subsume the most relations, is much more difficult. The problem with this test, as described in Chapter 7, is that it not easy to determine which relations a pattern can realize. For any given pattern, Ace provides easy access to the relations which are embodied in that pattern by treating these relations as features of the pattern. This does not include, however, the relations which may be embodied within the *constituents* of the pattern. A simple noun phrase or verb phrase pattern may be able to express a large number of relations because one of the constituents of the pattern is may be another complex noun phrase or verb phrase.

The question of how to realize as many relations as possible proved to be an implementation problem in KING, a problem which seems to stem from the disembodiment of grammatical relations from grammatical patterns. One solution to the problem would be to provide some means of starting from a pattern and looking ahead to determine what relations could be subsumed by each constituent of the pattern, thereby considering combinations of patterns instead of individual patterns at each stage in the generation process. The problem with this solution is that it introduces a great deal of complexity into an otherwise simple mechanism, requiring the generator to perform a substantial search through its grammar during the pattern selection phase and seeming to defeat the purpose of incrementality in KING.

The compromise solution adopted instead was to make a special case of chain rules, or patterns with only one constituent, and perform this lookahead only for these patterns. For example, the gerund phrase pattern corresponds to the chain rule "NP* -> P-VP", where NP* is a modifiable noun phrase and P-VP is a verb phrase with the verb part in progressive form, as in "John's being kissed by Mary". For this type of rule, KING considers verb phrase patterns along with noun phrase patterns, effectively treating gerund phrases as both verb phrases and noun phrases as described in Chapter 5. The internal structure of the gerund phrase derives from its verb phrase structure, while its external linguistic behavior and the progressive form constraint on the verb part are knowledge about a particular NP* pattern.

When the pattern selection stage is reached and the pattern type is a simple category such as verb, determiner, or noun, KING abandons its normal pattern selection algorithm and instead uses simple morphological knowledge to construct its output word. This is in contrast to the PHRED approach which painstakingly attacked morphology with the same mechanism as syntax. Word formation is accomplished by using a discrimination net to index simple morphological rules according to sets of constraints such as "TENSE = past" and "PERSON = third", storing exceptions along with each irregular verb. A lexical category such as *lex_give* has as one of its properties a *root* attribute, to which these rules may be applied to produce an output word. In the case of *lex_give*, the past tense form is an exception, "gave".

The pattern selection process in KING is relatively simple because most of the specialized knowledge of the generator is in linguistic relations rather than patterns. The *verb-indir-relation* and *verb-obj-relation* derived in this example are used to select the *dative-vp* pattern, as described in section 7.2. The process of using these relations to generate from the selected pattern is performed in the restriction phase, described in the next section.

8.5. Restriction

The input to the restriction process is the output of both the mapping and pattern selection phases. The restriction process in KING is implemented as a loop which considers each element of linguistic structure returned by the mapping process with respect to the pattern selected.

If the element being considered is a linguistic relation, KING uses the name of the relation to find the pseudonym for the relation as it applies to the particular pattern, and from this pseudonym retrieves the set of correspondences or ROLE-PLAYs between pattern elements and relation elements. In the case of the dative verb phrase and direct object as shown in figure 8-5, for example, this set is the following:

(ROLE-PLAY *obj dvp-np*)
(ROLE-PLAY *vo-verb dvp-verb*)

Using these ROLE-PLAY relations, KING assigns the structure playing the role of *obj* in the *verb-obj-relation* to the *dvp-np* role of the *dvp-pattern.*

A linguistic relation, like any knowledge structure in Ace, may have constraints imposed on its roles as well as having fillers for these roles. In this case, these constraints are attached to the corresponding elements of the selected pattern in the same manner as the concepts are added. Thus each constituent of the pattern may have attached to it a pattern category, a token referred to, and a set of linguistic constraints, which will serve as input to the generator in the generation of this constituent.

The application of relations to a pattern is one means of *elaborating,* or filling out, pattern constituents. Another means of elaborating constituents is enabled when the element of linguistic structure being considered applies entirely to a particular constituent of the pattern. In this case ROLE-PLAYs are used to determine the constituent to which the structure applies. If the structure is a constraint which conflicts with a constraint on the pattern itself, the pattern must be rejected. Otherwise, the structure is attached to any pattern constituent retrieved in ROLE-PLAY relations with the type of the constraint or structure as a pseudonym. This retrieval for the *dative-vp* yields the following relation, inherited from the *verb-phrase* structure:

(ROLE-PLAY *tense* (*tense* (of *vp-verb*)))

The term (*tense* (of *vp-verb*))) is used to designate the player of the *tense* role of the player of the *vp-verb* role. The *vp-verb* pseudonym is applied to the *dative-VP*, obtaining the role *dvp-verb.* The *tense* input constraint, "TENSE = past", is then added as an elaboration to the *dvp-verb* of the selected pattern.

Further aspects of the operation of the mapping - pattern selection - restriction mechanism are demonstrated in an example in the next section.

8.6. An Annotated Trace of KING

The following is a trace of KING producing the sentence "John was given a kiss on the cheek by Mary":

```
*******Running KING*******
```

KING starts from the Ace representation of a *kissing* event, and first applies a sequence of VIEWs as in the "giving a hug" example described in this chapter. For each of these VIEWs, the mapping process uses pseudonyms to determine the appropriate roles in the concept being produced.

```
-------------------------------------------------------------------

****Now generating from:

(kissing (kisser mary1) (kissee john1) (surface cheek1))

               Input structures:
nil

               Desired structure:  s

-------------------------------------------------------------------

*****Applying  map

               Map is  :

((action transfer-event) (actor source) (object recipient) (*all* object))
```

The designator **all** in the map to *transfer-event* indicates that the structure being mapped is not an aspectual but an absolute concept; thus the *kissing* concept itself is used to fill the *given-kiss* role:

```
               After mapping, struct is  :

(kiss-transfer  (source mary1)
                  (recipient john1)
                  (given-kiss (kissing (surface cheek1))))

-------------------------------------------------------------------

*****Applying  map

               Map is  :

((transfer-event giving) (source giver))

               After mapping, struct is  :

(kiss-giving (kiss-giver mary1) (recipient john1) (given-kiss (kissing (surface cheek1))))

-------------------------------------------------------------------

*****Applying  map


               Map is  :

((c-relation pred) (nuc subj) ((*asp* nuc) pred))
```

```
            After mapping, struct is  :

    (pred (c-subj john1)
         (c-pred
          (recipient
           (of (kiss-giving (kiss-giver mary1) (given-kiss (kissing (surface cheek1))))))))
```

In the map to *pred* (referred to as *predication* in the text), *nuc* (for *nuclear-term*) is a pseudonym for all roles of the concept which may be mapped into the *subj* role. For this example, the generator has been instructed to favor *john1*.

The term *(*asp* nuc)* is used to indicate that the pseudonym of the element playing the *nuc* role is mapped into the *c-pred* role, rather than mapping the ROLE-PLAYer itself. This is because *predications*, as presented in section 6.3, are considered to be references to aspectuals, rather than to absolute concepts.

This series of mappings produces the *pred* structure, which is linked by REF associations to linguistic relations. The mapping process next applies these REF associations:

```
------------------------------------------------------------------

Adding linguistic structures (mapping):

Structures at completion:

((sp (subj john1)
     (pred
      (recipient
       (of (kiss-giving (kiss-giver mary1) (given-kiss (kissing (surface cheek1))))))))
 (sv (subj john1)))
```

The *sp*, or *subject-predicate*, relation is the linguistic relation corresponding to the *predication* concept. The *sp* relation is realized in sentential forms. If KING had been called upon to refer to the same *predication* with a noun phrase, this stage of the mapping process would have produced the *posn* relation, corresponding to phrases such as "John's being given a kiss". In this case, the *sp* relation is used to retrieve the basic sentence pattern, *B-s*:

```
------------------------------------------------------------------

*****Filling out pattern (restriction):

(B-s
 ((bs-np nil nil np)
  (bs-vp nil ((person (matches s-np)) (number (matches s-np))) vp)))
```

Each constituent in the pattern above is indicated by an aspectual name, a referent token, a list of linguistic structures, and a pattern category.

The restriction process may fill in some of this information, and then passes the constituents individually back to the generator. The first constituent passed here is the reference to *john1*:

```
------------------------------------------------------------------

****Now generating from:

john1

              Input structures:

((case nominative))

              Desired structure:  np

------------------------------------------------------------------

Adding linguistic structures (mapping):

(person (name John))

Structures at completion:

((dn (n (person (name John)))) (name (name John)) (animate) (case nominative))
```

In the case of referring to *persons*, the mapping process adds a preference for the noun phrase pattern which employs a name. Otherwise the generator might use the *dn* relation above (for *determiner-noun*) to produce "the man", "the person", or "the man named John". The *dn*, *name*, and *animate* structures above are derived from mapping, while the *case nominative* constraint is passed along from the constraints on the basic sentence pattern.

The production of the noun phrase for *john1* results in the filling of the linguistic category *nm* (for "name") with the name *John*:

```
------------------------------------------------------------------

*****Filling out pattern (restriction):

(N-np ((N-np-nm nil nil nm)))
```

```
------------------------------------------------------------------

****Now generating from:

John

                Input structures:
nil

                Desired structure:  nm

------------------------------------------------------------------


------------------------------------------------------------------

****WORD FOUND**** :        "John"
```

Having completed the noun phrase part of the sentence, KING next attempts the verb phrase constituent referring to the *recipient* role of *john1* in the *kiss-giving* action:

```
****Now generating from:

(recipient (of (kiss-giving (kiss-giver mary1) (given-kiss (kissing (surface cheek1))))))

                Input structures:
nil

                Desired structure:  vp


------------------------------------------------------------------

Adding linguistic structures (mapping):

Structures at completion:

((vpas))
```

The structure *vpas* produced by the mapping process expands to the "VOICE = passive" constraint. This constraint is associated with *object* and *recipient* roles, so that references to these roles lead to passive forms. The earlier choice of *john1* as subject is thus what causes this to be a passive sentence.

When producing a reference to an aspectual, the mapping process automatically maps next from the concept in which the role is being played. In this case it is the *kiss-giving* concept:

```
-------------------------------------------------------------------

Adding linguistic structures (mapping):

(kiss-giving (kiss-giver mary1) (given-kiss (kissing (surface cheek1))))

Structures at completion:

((vo (obj (kissing (surface cheek1))))
 (pvby-adj (adj mary1))
 (lex_give)
 (vpas))
```

As described in the text, the mapping from the *simple-event* concept which DOMINATEs *kiss-giving* fills the role of *obj* in the *vo* relation (short for *verb-obj-relation*) with the player of the *object* role in the *simple-event*, which in this case corresponds to the player of the *given-kiss* role. Mapping from the *action* concept produces the *pvby-adj* relation (corresponding to *passive-by-adjunct* in the text) where the role of *adj* or *adjunct* is filled by the *actor* or *kiss-giver*. The lexical category *lex_give* is produced from the direct REF association with *kiss-giving*. The pattern selected to express these linguistic structures is the *adjunct-verb-phrase*, *A-vp*:

```
-------------------------------------------------------------------

*****Filling out pattern (restriction):

(A-vp ((avp-vp nil nil vp) (avp-adj nil nil adj)))
```

Filling out this pattern places *mary1* in the adjunct position. The other linguistic structures, using ROLE-PLAY relations, are used to fill attributes of the *avp-vp* constituent. The generator next tries to generate a form which expresses the remaining constituents. This results in the selection of a transitive verb phrase structure:

```
-------------------------------------------------------------------

****Now generating from:

nil

                Input structures:
((vpas)
 (voice passive)
 (lex_give)
 (person third)
 (number singular)
 (vo (obj (kissing (surface cheek1)))))

                Desired structure:  vp
```

```
------------------------------------------------------------------

*****Filling out pattern (restriction):

(B-vp ((bvp-verb nil nil v-p) (bvp-np nil nil np)))
```

Application of the *vo* relation to the above pattern results in the elaboration of the noun phrase constituent by the concept *maryl*. Elaboration using ROLE-PLAYs specifies the constraints on the verb part aspectual *bvp-verb*, which is generated next:

```
------------------------------------------------------------------

****Now generating from:

nil

               Input structures:

((lex_give) (vpas) (voice passive) (person third) (number singular))

               Desired structure:  v-p
```

The structure *vpas* is DOMINATEd in the Ace linguistic hierarchy both by *constraint* and *linguistic-relation*. As a constraint, it expands to "VOICE = passive". As a relation, it is a form of the *helper-verb* relation with the constraint that the "helper" is the auxiliary "be". The next pattern filled is the *verb-part* pattern which realizes this relation, the compound verb pattern:

```
------------------------------------------------------------------

*****Filling out pattern (restriction):

(C-v-p ((cvp-h nil nil h) (cvp-v-p nil nil v-p)))
```

The C-v-p pattern consists of two constituents, the first a *helping-verb* (abbreviated by *h*) and the second a *verb-part* (abbreviated by *v-p*). The elaboration phase of the restriction process applies the constraint *lex_be* on the helper constituent, as well as the agreement constraints. The helper is then generated:

```
-----------------------------------------------------------------

****Now generating from:

nil

            Input structures:
((lex_be) (person third) (number singular))

            Desired structure:  h

-----------------------------------------------------------------

****WORD FOUND**** :        "was"


-----------------------------------------------------------------

****Now generating from:

nil

            Input structures:

((form perfective) (lex_give))

            Desired structure:  v-p

-----------------------------------------------------------------

*****Filling out pattern (restriction):

(B-v-p ((B-v-p-v nil nil v)))


-----------------------------------------------------------------

****WORD FOUND**** :        "given"
```

KING has now completed the noun phrase and complete verb, corresponding to "John was given ... " The next constituent is the noun phrase referring to the *kissing* action. The mapping process applied to this concept produces a lexical category and two linguistic relations, a *determiner-noun* relation and the *vbo* relation. Like the *pvby-adj* relation, *vbo* is a linguistic relation that may be realized either in verb phrases or noun phrases, but the player of the *adj* role is the object of the preposition "on":

```
-----------------------------------------------------------------

****Now generating from:

(kissing (surface cheek1))

               Input structures:
nil

               Desired structure:  np

-----------------------------------------------------------------

Adding linguistic structures from:

(kissing (surface cheek1))

Structures at completion:

((idn (n (kissing (surface cheek1))))
 (vbo (adj cheek1) (v-p (kissing)))
 (lex_kiss))
```

The *determiner-noun* relation *idn*, corresponding to indefinite noun phrases, is produced by the generator as the default when referring to actions. The *vbo* relation here falls into the class of linguistic structures which can be realized only in postmodifiers, thus the pattern selected next is the postmodified noun phrase, *X-np*:

```
-----------------------------------------------------------------

*****Filling out pattern (restriction):

(X-np ((xnp-np nil nil np) (xnp-pmod nil nil pmod)))
```

The *idn* structure results in the selection of a determiner-noun pattern and restricts this pattern to an indefinite determiner:

```
-----------------------------------------------------------------

*****Filling out pattern (restriction):

(B-np ((bnp-det nil nil det) (bnp-np* nil nil np*)))
```

```
------------------------------------------------------------------

****Now generating from:

nil

              Input structures:
((ref indef))

              Desired structure:  det

------------------------------------------------------------------

****WORD FOUND**** :        "a"


------------------------------------------------------------------

*****Filling out pattern (restriction):

(B-np* ((bnp*-n nil nil n)))

------------------------------------------------------------------

****Now generating from:

nil

              Input structures:
((lex_kiss))

              Desired structure:  n

------------------------------------------------------------------

****WORD FOUND**** :        "kiss"
```

KING has now produced the structure corresponding to "John was given a kiss..." The next constituent produced is the postmodifier part of the noun-phrase, corresponding to the concept *cheek1*:

```
------------------------------------------------------------------

****Now generating from:

cheek1

              Input structures:
((vbo (adj cheek1) (v-p (kissing))) (prep_on))

              Desired structure:  pmod
```

```
------------------------------------------------------------------

Adding linguistic structures from:

(cheek (part-of john1))

Structures at completion:

((dn (n (cheek (part-of john1))))
 (inanimate)
 (lex_cheek)
 (prep_on))
```

Since the *vbo* relation can be realized only by a prepositional phrase, the postmodifier pattern selected is the prepositional phrase postmodifier, to which restriction adds the constraint that the preposition must be "on" and the noun phrase corresponds to the expanded *cheek1* concept:

```
------------------------------------------------------------------

*****Filling out pattern (restriction):

(P-pmod ((ppmod-pp nil nil pp)))

------------------------------------------------------------------

****Now generating from:

nil

               Input structures:
((prep_on) (vbo (adj cheek1) (v-p (kissing))) (dn (n (cheek (part-of john1)))))

               Desired structure: pp

------------------------------------------------------------------

*****Filling out pattern (restriction):

(B-pp ((bpp-prep nil nil prep) (bpp-np nil nil np)))

------------------------------------------------------------------

****Now generating from:

nil

               Input structures:
((lex_on))

               Desired structure: prep
```

```
------------------------------------------------------------------

****WORD FOUND**** :        "on"


------------------------------------------------------------------

****Now generating from:

nil

                Input structures:

((dn (n (cheek (part-of john1)))))

                Desired structure:  np

------------------------------------------------------------------

*****Filling out pattern (restriction):

(B-np ((bnp-det nil nil det) (bnp-np* nil nil np*)))


------------------------------------------------------------------

****Now generating from:

nil

                Input structures:
nil

                Desired structure:  det

------------------------------------------------------------------

****WORD FOUND**** :        "the"
```

KING still has no real knowledge to select the appropriate determiner here, and thus will choose a default, either "a" or "the". It then applies mapping to derive the lexical category *lex_cheek* and produces the corresponding noun:

```
----------------------------------------------------------------

****Now generating from:

(cheek (part-of john1))

                Input structures:
nil

                Desired structure:  np*

----------------------------------------------------------------

Adding linguistic structures from:

(cheek (part-of john1))

Structures at completion:

((dn (n (cheek (part-of john1)))) (inanimate) (lex_cheek))

----------------------------------------------------------------

*****Filling out pattern (restriction):

(B-np* ((bnp*-n nil nil n)))


----------------------------------------------------------------

****Now generating from:

nil

                Input structures:
((lex_cheek))

                Desired structure:  n

----------------------------------------------------------------

****WORD FOUND**** :        "cheek"
```

The final phrase produced is the prepositional phrase corresponding to the *by-adjunct*. At this point KING has completed the noun phrase "a kiss on the cheek" and is generating the final constituent of the *adjunct-verb-phrase* pattern:

```
------------------------------------------------------------------

****Now generating from:

maryl

               Input structures:

((pvby-adj (adj maryl)) (prep_by))

               Desired structure:  adj

------------------------------------------------------------------

Adding linguistic structures from:

(person (name Mary))

Structures at completion:

((dn (n (person (name Mary))))
 (name (name Mary))
 (animate)
 (pvby-adj (adj maryl))
 (prep_by))
```

The production of the *by-adjunct* proceeds in exactly the same way as the generation of the "on" prepositional phrase, except for a different constraint on the preposition itself:

```
------------------------------------------------------------------

*****Filling out pattern (restriction):

(B-adj ((B-adj-pp nil nil pp)))

------------------------------------------------------------------

****Now generating from:

nil

               Input structures:
((prep_by)
 (pvby-adj (adj maryl))
 (name (name Mary))
 (dn (n (person (name Mary)))))

               Desired structure:  pp
```

```
-----------------------------------------------------------------

*****Filling out pattern (restriction):

(B-pp ((B-pp-prep nil nil prep) (B-pp-np nil nil np)))

-----------------------------------------------------------------

****Now generating from:

nil

                Input structures:
((lex_by))

                Desired structure:  prep

-----------------------------------------------------------------

****WORD FOUND**** :        "by"
```

The descriptor for *mary1*, after expansion, is produced in the same way as that for *john1*. This results in the completion of the surface structure, and "walking" through the surface structure results in the final utterance.

```
-----------------------------------------------------------------

****Now generating from:

mary1

                Input structures:
((dn (n (person (name Mary)))) (prep_by))

                Desired structure:  np

-----------------------------------------------------------------

Adding linguistic structures from:

(person (name Mary))

Structures at completion:

((name (name Mary)) (animate) (dn (n (person (name Mary)))) (prep_by))
```

```
-----------------------------------------------------------------

*****Filling out pattern (restriction):

(N-np ((N-np-nm nil nil nm)))

-----------------------------------------------------------------

****Now generating from:

Mary

               Input structures:
nil

               Desired structure:  nm


-----------------------------------------------------------------

****WORD FOUND**** :        "Mary"

*****************************************************
                    OUTPUT
*****************************************************

John was given a kiss on the cheek by Mary.

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

8.7. Analysis of KING

At the time of this writing, KING is a fully implemented tactical language generator with the ability to produce utterances which form a useful and exemplary subset of natural English. The system is incorporated in a new version of the UNIX Consultant, which is still under development. Written in Franz Lisp and compiled, the code of the generator, exclusive of indexing and knowledge representation tools, is slightly more than 20K, about one-fifth the size of the analogous code of PHRED. The Ace implementation takes up another 15K. The knowledge base includes about 50 basic linguistic patterns, 75 linguistic relations, several hundred Ace assertions describing these structures, 150 concepts, and 200 "horizontal" structured associations. The typical running time of the program is about two seconds for a sentence of 10-15 words.

8.7.1. Successes

In a variety of ways KING has proven to be a strong qualitative success. The most obvious successes are in alleviating the difficulties in making use of new knowledge in the system.

With respect to the knowledge acquisition bottleneck, the Ace framework has proven to be a great help. Because the representation of the system allows new information to be easily linked to existing knowledge, extending the power of the system is far easier than it was in PHRED. For example, the addition of a new concept related to the *transfer-event* enables the use of the dative construct. The encoding of metaphorical generalizations in the UNIX consultant, such as the relationship between access and possession realized in "You need write permission", is accomplished by using structured associations. New grammatical patterns often require only the specification of the linguistic relations in the pattern.

KING gets more power per knowledge cell because the hierarchical representation eases the problem of redundancy. Furthermore, the use of indirect mappings from meaning to language provides, for a given concept, a broader range of potential utterances than direct mappings.

KING is smaller than PHRED mostly because it suffers less from the knowledge adaptation tie-up. Much of the code in PHRED, as in most knowledge-based systems, was written to make up for rough spots in the representation and to handle linguistic phenomena which did not fit well with the representation. As a result, PHRED had sections of code which were distinctly devoted to verbs, noun-phrases, and pronouns. Code explosion tends to follow the separation of phenomena which might be treated uniformly. In principle, KING is small because it is *knowledge intensive*, because the power of its knowledge base obviates the need for specialized code.

One of the major technical problems to be overcome in language generation systems is the consideration of many possible linguistic choices which are not applicable or have little chance of success. By incorporating a significant data-driven component, KING avoids many such choices. The process of selecting linguistic structures, as described in sections 7.2 and 8.4, is largely dependent on the results of the data-driven mapping mechanism. The proliferation of categories in the representation also takes much of the work out of the process of selecting structures and instead puts it into combining and synthesizing structures, or restriction. Unification-based generators tend to perform selection within the unification mechanism; others apply queries or other matching methods. The pattern selection process in KING is much simpler because it takes advantage of the relations derived from mapping. Thus it is the *incremental* nature of the generation mechanism that eliminates many of the frivolous possibilities.

The advantages described above are basically software successes, due mostly to the cognitive approach. A less tangible, but more important, success of the work presented here is a reshaping of the generation task and the identification and redefinition of problems within this new framework.

The structured associations used by KING to produce language from meaning prove to be a powerful linguistic tool, providing a means of taking advantage of generalizations and enabling indirection in language. They do this by transforming much of the generation process into a search problem where the search space includes not only a range of linguistic constructs but a multitude of related conceptual structures as well. This mars the traditional what to say/how to say it distinction, as the search is constrained not only by what is to be expressed but also by the context and by the available linguistic tools, including metaphorical or metonymical relationships.

Since the "how to say it" component of generation seems to have a "what to say" element to it, one must consider whether to apply a planning model such as Appelt's or a rhetorical structure model such as Mann's or McKeown's to this phase as well. Certain aspects of text planning and rhetorical structure have their influence at all levels of language production, but there seems to be a difference between the construction of text plans and the *execution* of the plans. The distinction between the strategic and tactical aspects of generation still holds, but strategy cannot be isolated from tactics. In Appelt's hierarchical model, the interaction between levels of language planning consists effectively of communication between various planning components. In the KING model, the interaction is accomplished by having intentional strategies, like contextual information, bias the search for tactics.

Another result of the KING implementation is the identification of important aspects of the generation process which may be accomplished by general knowledge-manipulation strategies. The initial implementation of a parser-generator using the Ace representation (cf. Jacobs and Rau, 1984) took advantage of three general knowledge manipulation mechanisms which were used for both analysis and generation. Because of the basic differences between analysis and production, generation within this model relied heavily on the manner in which maps were applied, while analysis depended more on the process of combining the results of the mapping. The mechanism which performed this combination, originally used for both analysis and generation in the original system, was replaced by the restriction mechanism described here. Both analysis and generation depend heavily on the selection and use of structured associations in Ace, and thus accessing and applying these structured associations is a critical knowledge manipulation tool.

8.7.2. Limitations

Fortunately, the development of the Ace representation and the KING generator has achieved success by virtue of technical improvements in the representation of knowledge about language and in tactical language generation. It has proven to be a useful tool as part of a natural language help facility. Unfortunately, all the work that has been done in language production has still barely scratched the surface of the generation problem.

Probably the greatest of the unsolved problems in generation is the problem of context. The work presented here describes a generator equipped to produce grammatical output as a communicative tool. Like most such systems, it has certain knowledge about the role of context in generation, which can lead to favoring certain linguistic constructions over others. The broad problem of context in language, however, may be viewed as the interaction of beliefs, intentions, situational knowledge, and discourse knowledge. The use of context in language generation requires a general mechanism for dealing with this interaction as well as a representation scheme for encoding the contextual knowledge and a means of simulating the relative influences of various contextual elements. All these effectively constitute a model of human memory, which is unlikely to exist for some time. The treatment of contextual information as biases on the application of structured associations, however, is an area whose exploration may produce some short term success.

The main representation tool used by KING which is not available in other generators is the structured association, and the use of this tool is largely responsible for KING's success in knowledge-intensive, incremental generation. This technique, however, has the negative effect of introducing the complexity of controlling the application of these associations. To a degree, the search problem that has been partially solved by eliminating the consideration of candidate linguistic structures has been transferred to the search problem in selecting structured associations. This complexity is a potential problem; however, relatively simple methods of controlling the search, as described in section 7.1, produce very good results.

One technical problem that arises from the Ace representation is the start-up cost of building a knowledge base. The examples of Ace representation presented here have been primarily pieces of knowledge about events and verb phrases, because the verb phrase structure is most essential to sentence-level generation. The encoding of each area of an Ace network requires a substantial initial effort. Once this initial activation is overcome, it becomes much easier to add related knowledge to the network. But because specific knowledge in the system depends on more abstract knowledge, the correct organization of the abstract knowledge is especially important. This means that the benefit of the Ace representation in adding new knowledge to the system comes only after the most fundamental knowledge is encoded.

Summary

This chapter has presented some of the implementation details of KING, a simple, knowledge-intensive generation mechanism. The program utilizes the processing principles of Chapter 7 to exploit the Ace representation. The use of ROLE-PLAY relations is especially important to the generator in allowing structured associations to apply to the task of mapping from general or specific knowledge structures. The incrementality of the generator makes the pattern selection process difficult from an implementation standpoint, but allows the mechanism to make maximal use of the knowledge derived from mapping across structured associations. The restriction process is performed smoothly without unification by making use of explicit and implicit ROLE-PLAY relations.

The KING implementation seems to support the idea that a representation such as Ace enables the use of a simple mechanism to achieve substantial generation capabilities. The knowledge base of the generator is still relatively small, but the ease with which it has been adapted suggests a strong potential for expanding the capabilities of the system.

9. Summary and Conclusion

The basis for this thesis is the cognitive and practical motivation for the development of knowledge representation tools for language processing. This foundation has been realized in the Ace representation framework, which utilizes methods in linguistic and conceptual representation to synthesize an associative, uniform, and hierarchical representation applicable to the language generation task. The KING generator is a small, knowledge-intensive mechanism using an Ace knowledge base to provide a system within which the processing entailments of the representation can be explored. The practical and theoretical results of this work have two areas of impact: (1) the increased understanding of representational and processing aspects of language generation, and (2) the suggestion of areas in which the work can be further explored.

The first area above supplies some immediate positive feedback, in the perception of improvements in a system. The second area, however, is perhaps most important in this type of work. This research has concentrated on the development of tools, and the power of a tool is best tested by continued use. It is safe to admit that most aspects of the language generation task have *not* been solved and to assert that many can be productively explored. Thus the work presented here is geared toward speeding the evolution of language generation systems by presenting a framework within which further research can be conducted.

9.1. Summary

The knowledge-based approach described in this thesis suggests that the problem of generating language be attacked from a broad perspective in which linguistic phenomena are considered with respect to issues in knowledge representation. The practical problem addressed is that of building extensible and adaptable natural language systems. While this problem is related to redundancies in knowledge representation, it is not parsimony of representation directly that alleviates them. Parsimony is an indicator that a knowledge representation provides the ability to exploit generalizations, a facility which also makes a knowledge base easier to adapt and extend.

An examination of *high-frequency, low-content* verbs such as "give" and "take" suggests that the generalizations behind specialized linguistic knowledge do not generally apply throughout a particular class of constructs. There are, however, manifest structural similarities in the linguistic constructions which involve these verbs. The use of such similarities depends on the ability to integrate general and specific knowledge into a representational framework for language processing.

The generalizations mentioned above are often metaphorical conceptual relationships which lead to particular linguistic constructions. For example, expressions such as "give a punch" and "take a punch" are motivated by the conceptual *action as transfer* view. That such conceptual relationships seem to allow particular forms of reference suggests a representation of explicit referential and metaphorical relationships among concepts. These relationships, called *VIEW*s, may be used in generation to refer indirectly to concepts by generating references to other concepts.

The interaction of generalized and specialized knowledge suggests a *hierarchical* representation for both linguistic and conceptual knowledge. The use of conceptual relationships supports the idea of representing *explicit referential knowledge* in the representation. An additional representational objective is *uniformity,* which makes it easier to handle the interactions among different types of knowledge.

Hierarchy, explicit reference, and uniformity are realized in the Ace representation system. Ace adopts the linguistic foundations of feature systems such as unification grammar, but builds from these foundations a hierarchical linguistic representation, using a common framework for linguistic and conceptual knowledge. Essential to this framework is the notion of a *structured association*, a relation among knowledge structures which has associated with it structure relating other knowledge structures. The structured association is used to organize linguistic and conceptual knowledge and to connect linguistic knowledge to conceptual knowledge.

The application of the Ace representation to the problem of exploiting generalizations is illustrated by the verbs "buy" and "sell", which refer to the *commercial transaction* event in Ace. Structured associations relate the concepts *buying* and *selling* to components of the commercial transaction event. These structured associations are represented as special cases of the associations between *taking* and *giving* and transfer events. Thus the correspondences between linguistic and conceptual roles in "John sold Mary a book" and "John gave Mary a book" are represented at an abstract level. This abstraction is exploited in the representation of other related verbs and concepts. This leads both to a more sparse encoding of knowledge and to more effective means of adding new knowledge to the Ace network.

The use of Ace in facilitating indirect reference is presented using the examples of "giving a hug" and "bury the hatchet". Both of these are examples of specialized constructs which refer to events indirectly by referring to metaphorical views of the events. A chain of structured associations encodes such indirect relationships and allows the representation of such flexible constructs. Fixed or unmotivated expressions, such as "kick the bucket", and handled easily without indirection. The Ace framework thus provides a uniform set of tools for representing a variety of knowledge about language.

The Ace representation is used by KING (Knowledge INtensive Generator) to produce language from a conceptual representation. KING is a small, simple mechanism built to apply the power of the underlying knowledge using a combination of data-driven and top-down methods. A tactical language generator along the lines of PHRED, KING demonstrates how some of the more complex or time-intensive aspects of generation are simplified because of a more suitable organization of knowledge. The generator produces the output of a new version of the UNIX Consultant system, which is still being developed. It has been tested independently with positive results in the production of a range of linguistic constructs.

9.2. Directions for Further Research

This work has concentrated on some representational and tactical aspects of language generation. Planning, focus, context, rhetorical structure, and other matters are essential to the generation task. This thesis has not addressed any of the theories which concentrate on these problems. Progress in these areas, however, depends heavily on the development and improvement of representational and processing tools. I intend that the tools presented here will be useful in the continuation of research on planning, focus, context, rhetorical structure, and other critical topics in generation. One of the goals of the Ace tools is to ease the interface between the strategic and tactical aspects of generation; thus extending the use of these tools to the strategic level should be productive.

Two areas for future related research that have particular promise are the study of context and of knowledge acquisition. Both pose extremely difficult theoretical and practical problems. The problem of context sensitivity is fundamental for generators which are to be used with knowledge about different users, domains, and situations. The problem of acquisition is critical because it is hard to believe that true robustness can be achieved without automated methods of adding knowledge to a system.

One reason for promise in the problem of context is the hierarchical organization of structured associations in Ace. The associations which link linguistic and conceptual structures themselves belong to conceptual categories, and thus may be accessed through a variety of organizational nodes. Because the encoding of this associative knowledge thereby leaves room for the explicit and implicit encoding of knowledge about the application of these associations, there is room for the use of contextual information to control which associations are applied in generation.

A second reason why the framework proposed here may improve context-sensitive aspects of the generation task is the decreased granularity of knowledge in the system. One of the problems with using contextual information effectively in language is that language processing systems often apply large pieces of knowledge, which may have complex contextual constraints. The emphasis on the incremental use of smaller chunks of knowledge simplifies the constraints on each chunk and emphasizes the process of incrementally building new knowledge structures. As the use of context is a process which involves the interaction of many types of knowledge structures, a model in which this interaction is stressed seems well suited to the problem of context. The KING implementation provided a system in which certain simple aspects of context could be tested, and showed some positive signs that structured associations could allow for the encoding of contextual information.

The knowledge acquisition problem may be made easier because the Ace framework emphasizes the exploitation of abstract knowledge in a system. The addition of new knowledge often involves adding a new linguistic category, determining where in the hierarchical organization the new category fits, and associating this category with a conceptual structure. The process of adding knowledge "by hand" is made easier because it is easier to add knowledge that is related to existing knowledge; the process of acquiring knowledge automatically should be similarly facilitated.

One of the most interesting aspects of developing and applying KING has been the light which linguistic demands place on the organization of conceptual knowledge. Ace and KING were designed with the goal of attacking issues in knowledge representation which influence the generation process, and the implementation of the systems punctuated more strongly than expected the role of conceptual representation and organization in language processing. The consideration of a variety of linguistic phenomena, such as relative clauses, EQUI-, and WH- movement, all provoked questions in conceptual representation. The way in which problems such as these were handled made heavy use of multiple inheritance and ROLE-PLAY as implemented in Ace. The ability to handle these constructs, like the ability to produce many specialized linguistic constructs, was achieved using relatively little additional knowledge of linguistic structure. The relations between concepts and language, and the relationships among concepts, proved to be most critical in the ability to generate these linguistic structures. The automated acquisition of linguistic knowledge, therefore, may be approached by considering the effect of conceptual organization on linguistic capabilities.

9.3. Conclusion

A knowledge-based approach to the task of natural language generation highlights theoretical issues in natural language processing which are behind practical problems in building generation systems. The generation task is rooted in the problem of representing knowledge about language; thus, many of the difficulties of the task are alleviated by addressing the theoretical problems at the level of knowledge representation, and applying the power of the knowledge to the task of generation.

The Ace representation is a reshaping and reworking of the knowledge representation tools required in language processing. The tools address theoretical and cognitive issues at the same time as they are tuned to the task of building adaptive and extensible systems. The KING generator presents a simple demonstration of the power of the representation applied to the generation problem.

The bulk of the areas of application of the Ace framework are open for further exploration. The immediate positive results presented here are the improved handling of phenomena which were difficult for previous systems, the ease with which a sampling of linguistic and conceptual knowledge is encoded in Ace, and the applicability of a simple, efficient, generation mechanism making use of this knowledge.

Appendix A: An Annotated Ace Grammar

The most common linguistic patterns used by the KING generator are presented here, with some analysis, to show how basic linguistic constructs are handled. The choice of what subset of English to include was made primarily to provide a reasonably robust language facility, but also to ensure that the generator could demonstrate the capacity to handle a range of linguistic constructs.

Naturally, there will be mistakes and omissions in this knowledge base. The scrutiny of the patterns and discussion here should nevertheless provide insight into the workings of KING and the development of a knowledge base in the Ace representation.

Only the pattern templates are presented here. These patterns often inherit properties of high-level templates; for example, the ROLE-PLAYs which represent correspondences between attributes as discussed in Chapter 8. The knowledge about constraints on certain constituents, which may also be inherited from such high-level templates, is also generally omitted. In most cases these constraints will be apparent.

The sentence templates used by KING are presented first. Below is the basic sentence template discussed in Chapter 5.

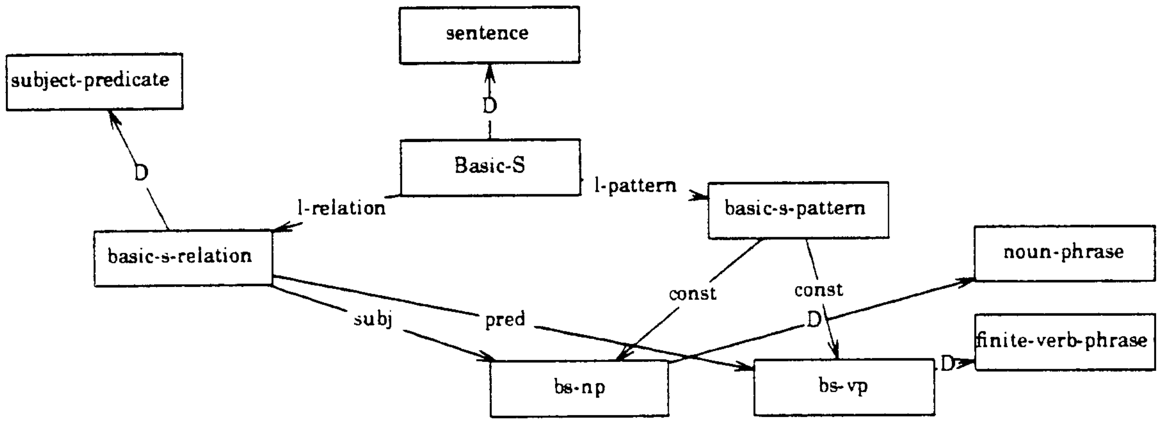

Figure A-1.

The imperative sentence pattern, returned by the pattern selection mechanism when a command or instruction is desired, is simply a verb phrase whose verb is in the tenseless form:

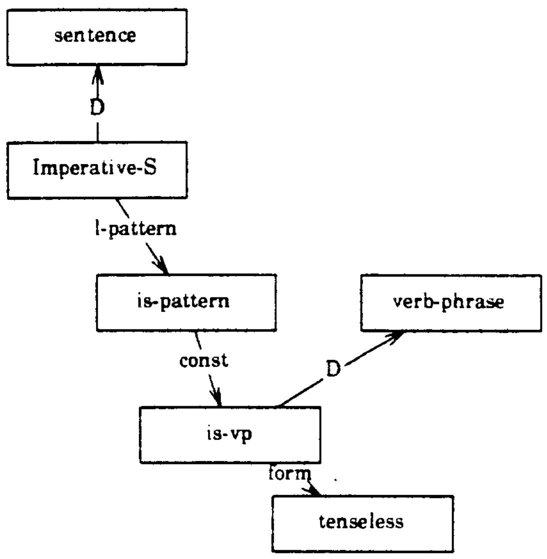

Figure A-2.

The handling of questions is a more complex problem. Wh-questions are handled using two different templates, one which is equivalent to the basic sentence pattern and the other handling Wh- forms where the Wh- term is part of the predicate. The following is a general wh-question template, handling those forms where the Wh- term is not the subject of the sentence. The relation *verb-term* DOMINATEs all relations which link a verb to other constituents which might appear in a verb phrase with the verb, including *verb-adjunct*, *verb-object*, etc. The Wh- constituent must thus be part of one of these relations. The *wh* constraint is assigned within the Wh- constituent, which determines the pronoun, such as "who", or determiner, such as "which", ultimately used.

Do-support, mentioned in Chapter 7, is a rare example of a relation, in this case the *helper-verb* relation, whose aspectual, *helper*, is filled by default if the restriction process fails to fill it. The pattern below, therefore, is selected on the basis of two relations, *verb-term* and *subject-predicate*, which because of the nature of the mapping mechanism means whenever the question term has not been mapped into a *subj* role. If the question term is in the *subj* role, the Basic-S pattern is used, with the *wh* constraint assigned to the NP term.

There are undoubtedly many potential linguistic arguments about this treatment. PHRED's knowledge base, and most others to which I have been exposed, explicitly represented the correspondence between the Wh- term and a "gapped" or "silent" component of the VP. This was better for examples where an adjunct is not "pied piped", e. g. to produce the unpreferable "Whom was Mary hit by?" instead of "By whom was Mary hit?" The "gap" method may also be easier in analysis, because it allows the same analysis mechanism to be used once the "silent" constituent has been inserted. The method espoused here, in both analysis and generation, instead takes advantage of the non-redundancy stipulation. In generation, this means that the relation associating the Wh- term and verb is not expressed in the verb phrase because it is expressed elsewhere; in analysis, it means that this relation must be determined on the basis of which relations were *not* expressed elsewhere. The main problem with this approach is that it requires grammatical patterns which are not ordinarily used except in "gapped" constructs; for example, the verb phrase "give Mary" is grammatical, but may only be used in cases such as "What did John give Mary?" and "The book which John gave Mary".

The inverted Wh- question template is given below:

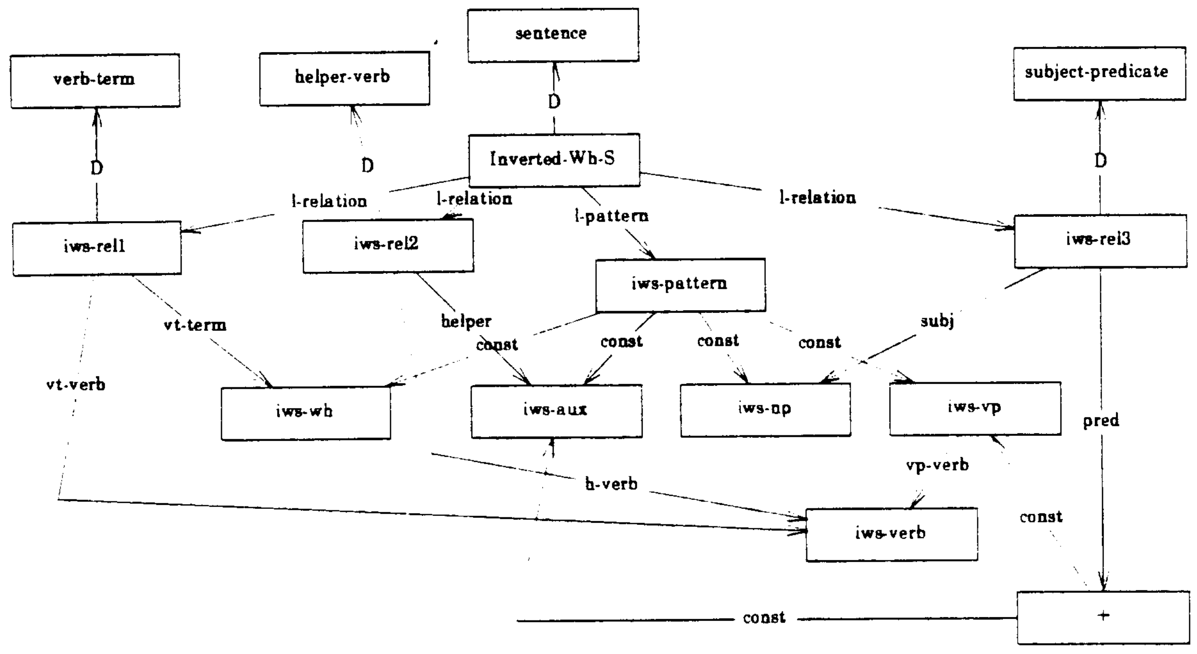

Figure A-3.

The inverted sentence construct, used for all yes-or-no questions, is similar to the inverted Wh- construct, but is spared the complication of the extra relation. The operator "+" in both templates is used to ensure proper agreement of subject and auxiliary.

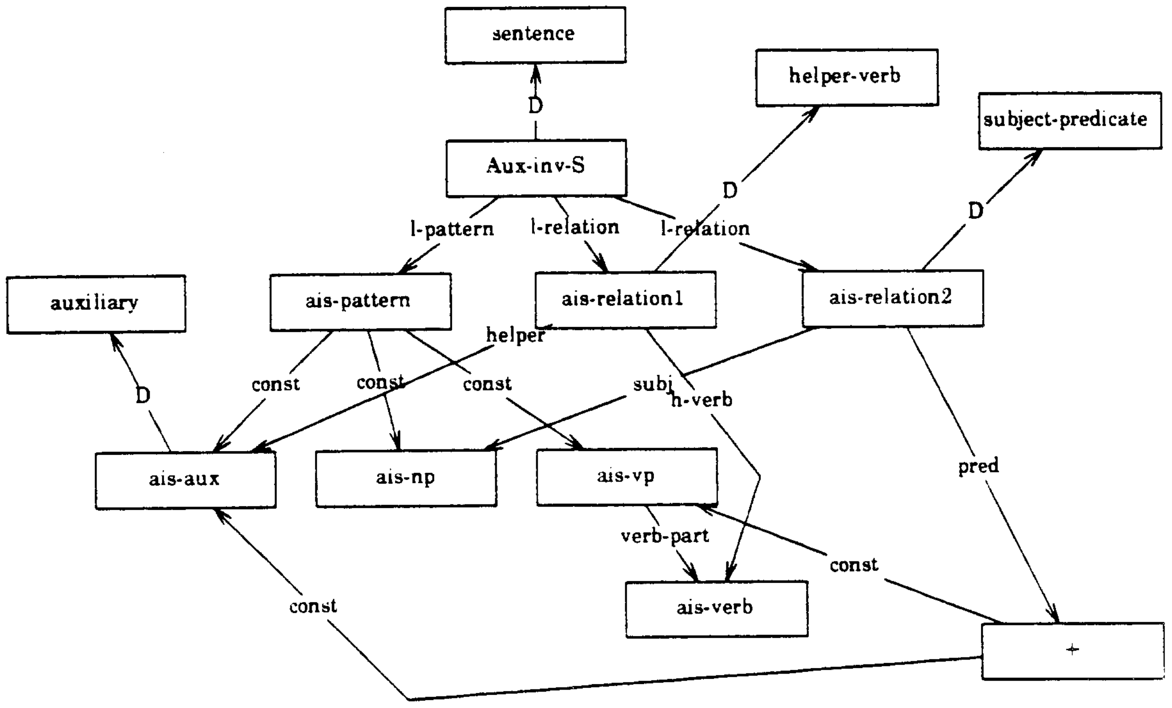

Figure A-4.

Subordinated sentences here are treated as two sentences joined by a subordinating conjunction. This includes constructs such as "Delete the file after you have printed it" and "What message was produced when you executed the command?"

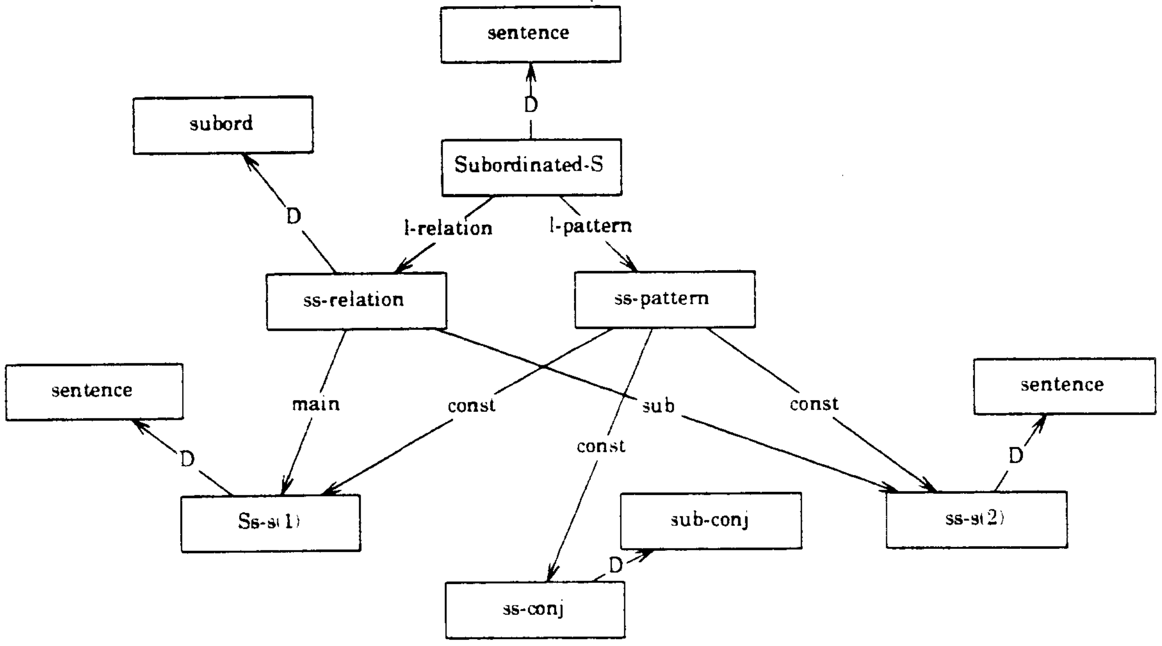

Figure A-5.

The next patterns are the relative clause templates. These are very similar to the Wh- and inverted Wh- sentences, except that the relative clause with the relative pronoun in the subject position does not appear in inverted form:

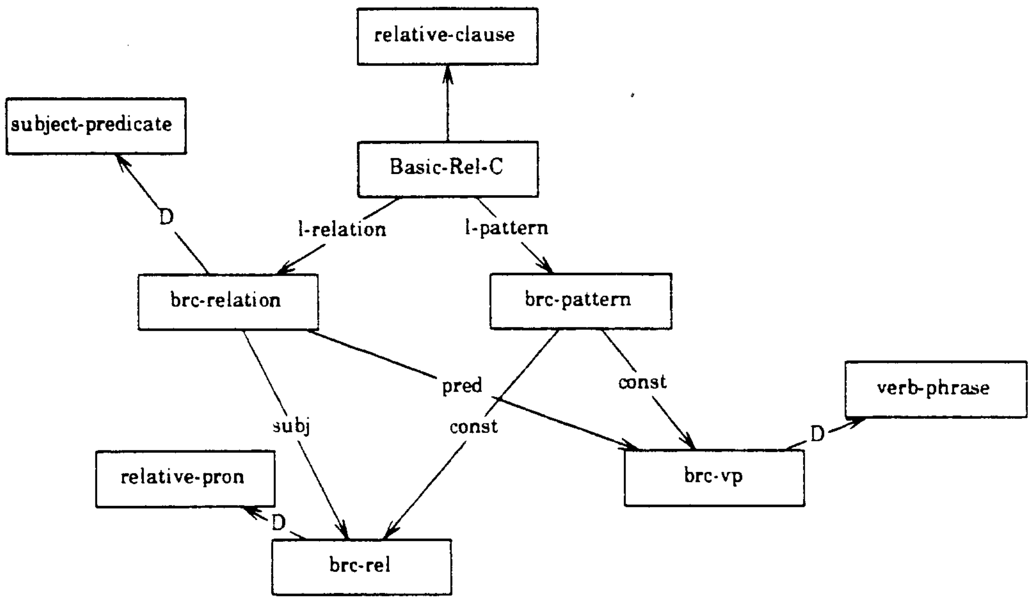

Figure A-6.

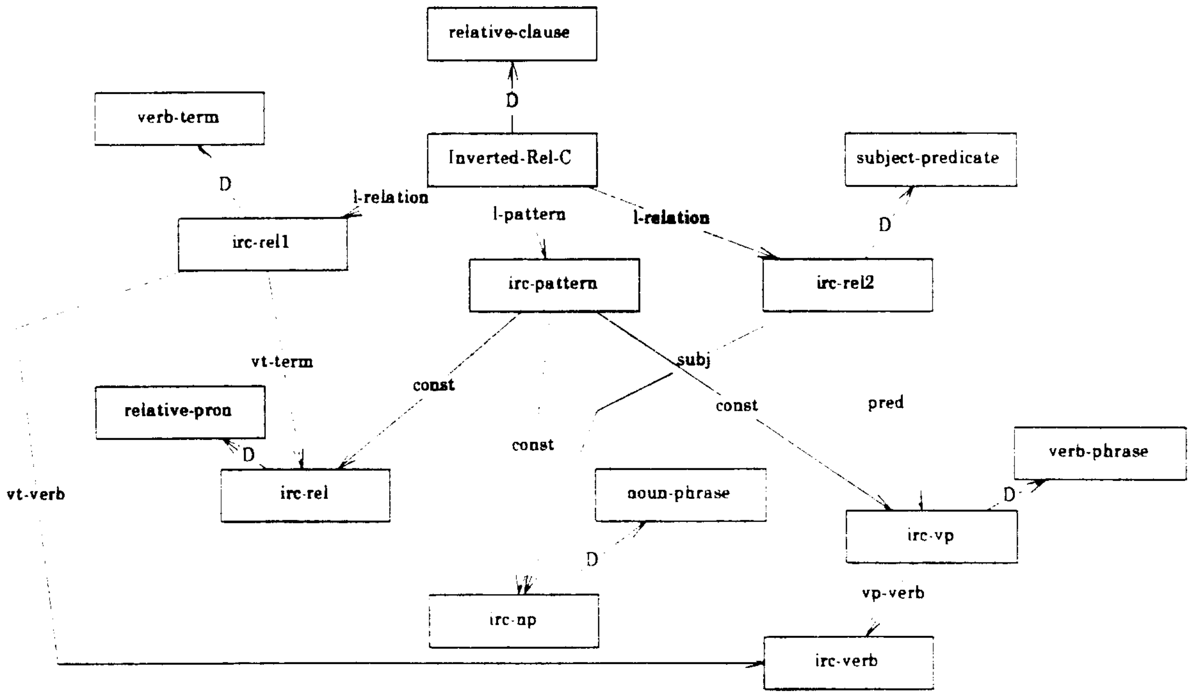

Figure A-7.

The verb phrase templates represent the widest range of constructs handled by KING. The presentation in Chapter 5 discussed how the verb phrase patterns, in conjunction with the hierarchy of grammatical relations, could be used to encode parsimoniously the ability to produce language such as "Selling John the book bothered Mary," and "John wanted to be sold the book by Mary."

As the discussion in Chapter 5 pointed out, a wide range of linguistic attachments to verbs are treated within the class of adjuncts in the Ace representation in order to minimize the amount of syntactic knowledge. Thus passive *by-adjuncts* require no special syntactic patterns. Because the objects and indirect objects of verbs require special ordering, there are patterns for this purpose. Adjuncts are handled by the following knowledge structure, introduce in Chapter 5:

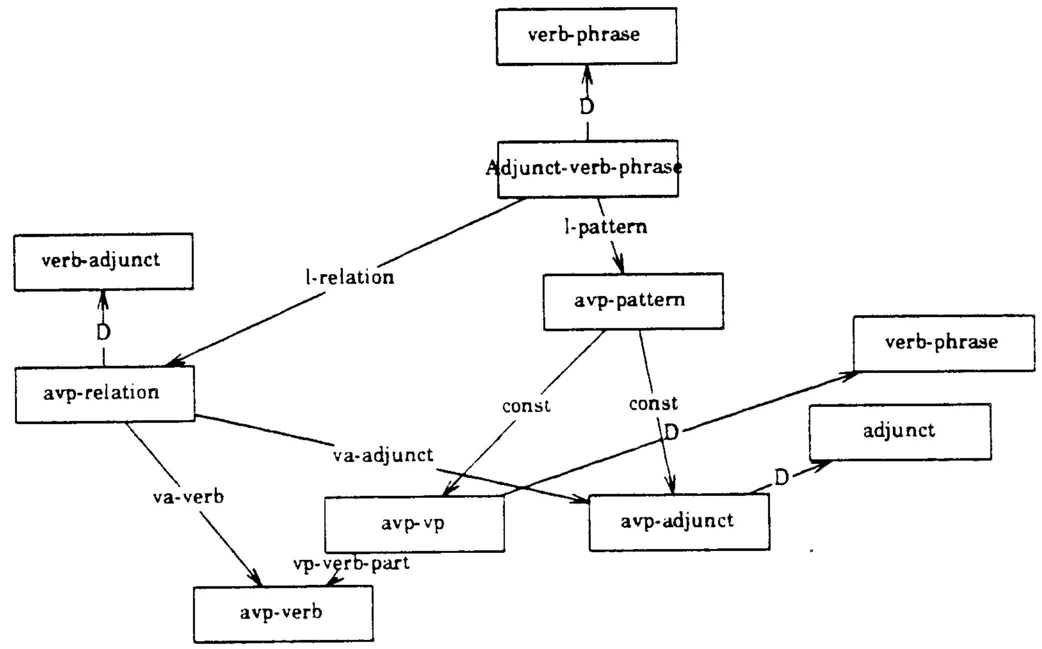

Figure A-8.

The following are the patterns which handle orderings within verb phrases, exclusive of adjuncts. These have been discussed and diagrammed in Chapter 5:

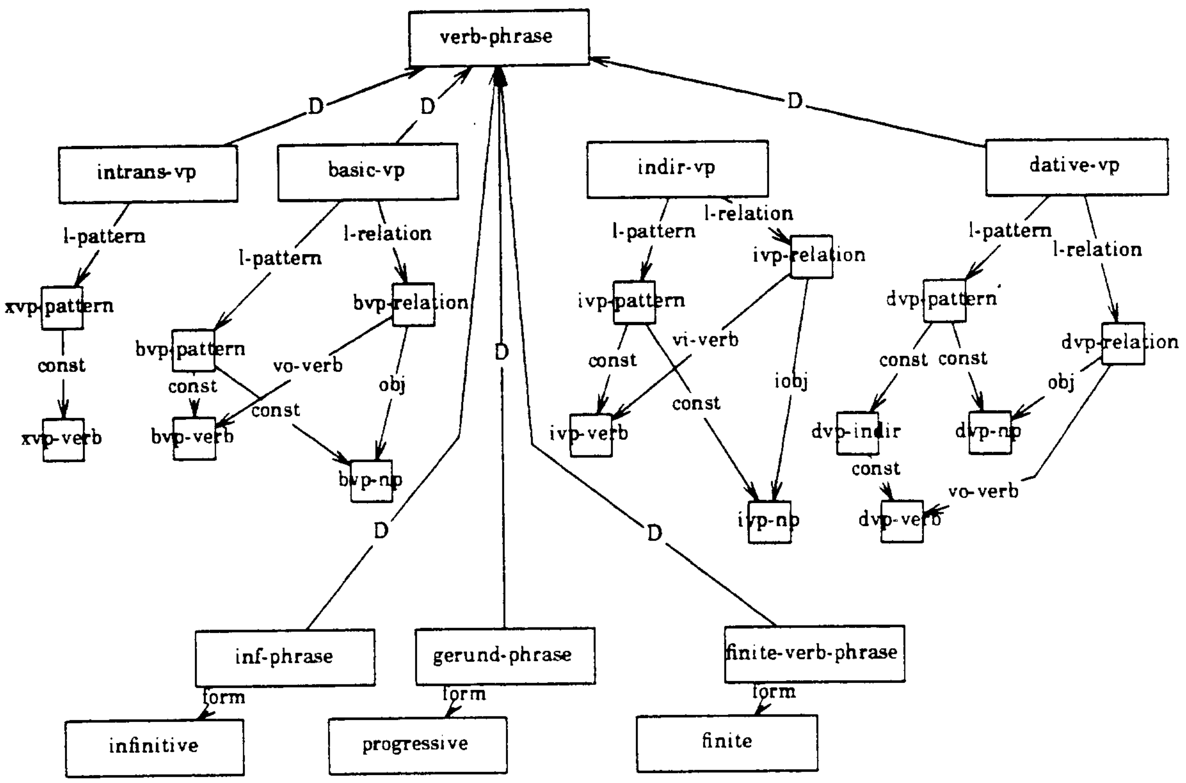

Figure A-9.

Below are given the verb phrase patterns which describe states, rather than actions. The first of these is the predicate modifier, which is used for "John was sick", "Mary looked sick", and "The file became lost":

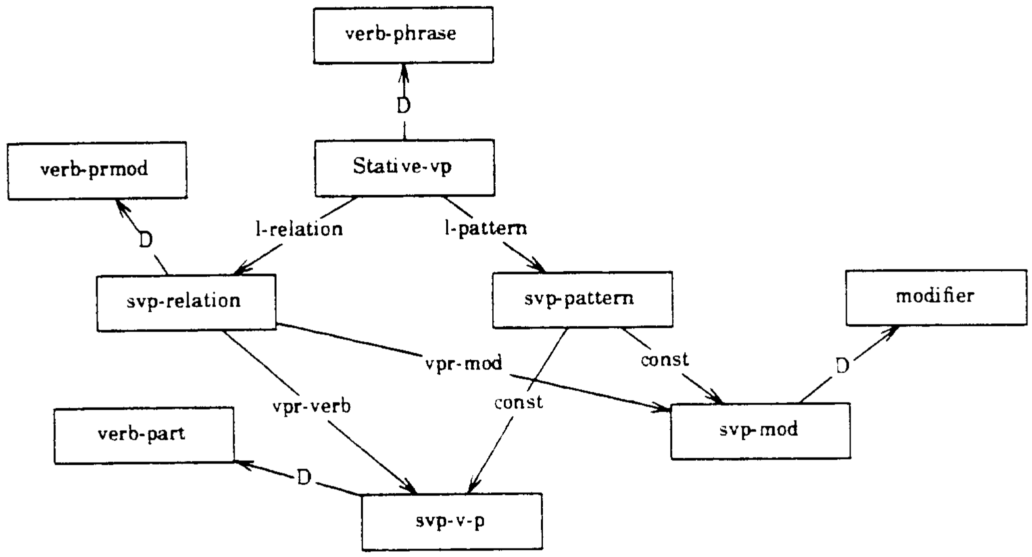

Figure A-10.

The *verb-prmod* relation is used to indicate relations between verbs and modifiers which correspond to a state of the subject of the verb for stative verbs and the object of the verb for transitive verbs. The cases of "Bill made John sick" or "Make the file publicly readable" are handled by the following pattern:

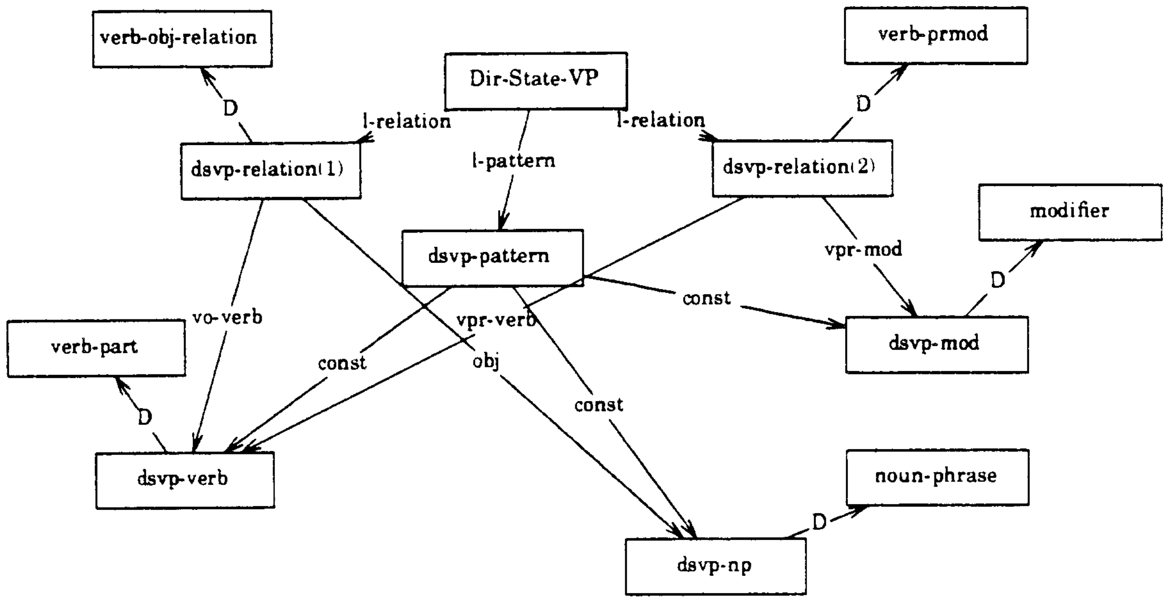

Figure A-11.

A class of verb phrase patterns incorporates other verb phrase patterns. This is the case with constructs used with EQUI verbs such as "want" as in "John wanted Mary to go", and "John promised Mary to take out the garbage". Some of the other applications of these patterns have been considered in Chapter 6. The same pattern may be used for both of these sentences, because the *subject-predicate* relations are determined during the mapping process from the conceptual structure to be expressed. While these *subject-predicate* relations are not directly satisfied by the sentence structure, they are available to guide constraints, such as reflexivization.

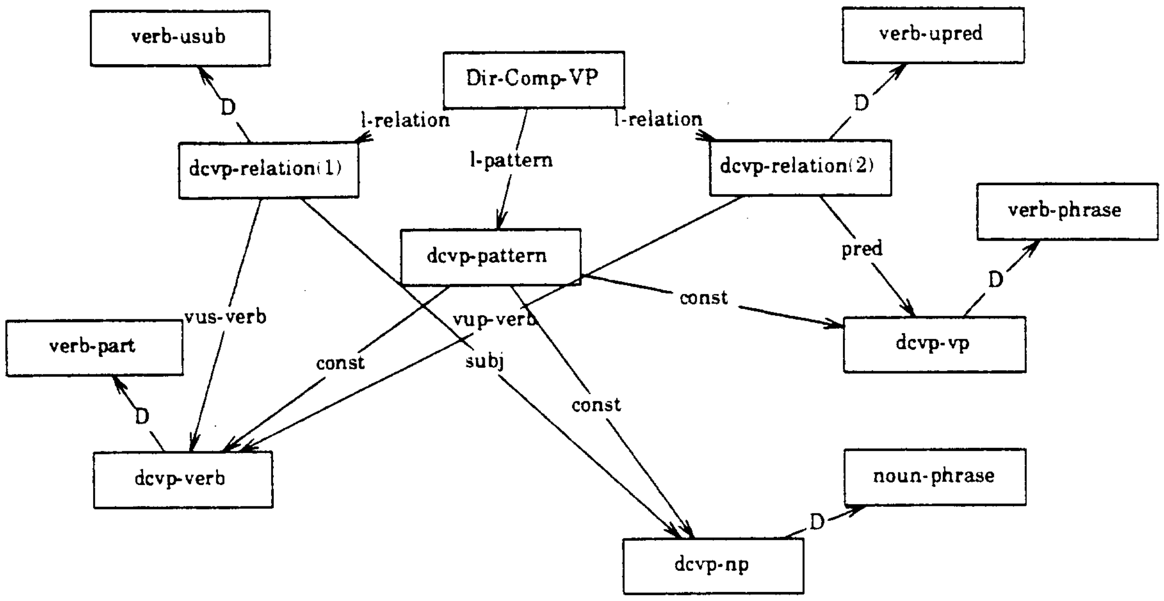

Figure A-12.

Verb phrases can appear alone as complements in verb phrases, such as in "John wanted to go" and "John had to sell Mary the book".

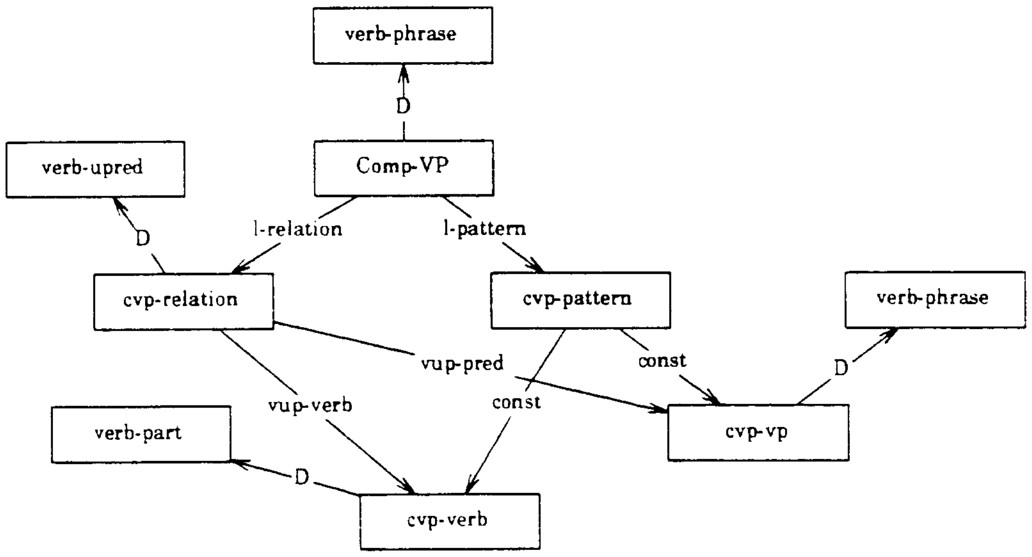

Figure A-13.

KING works from a small repertoire of noun-phrase patterns, relying on specialized knowledge about referring to objects to select noun phrases. For example, there is a structured association between the concept of a *concrete-object* and the *determiner-noun* relation which causes the selection of the Basic-NP pattern below:

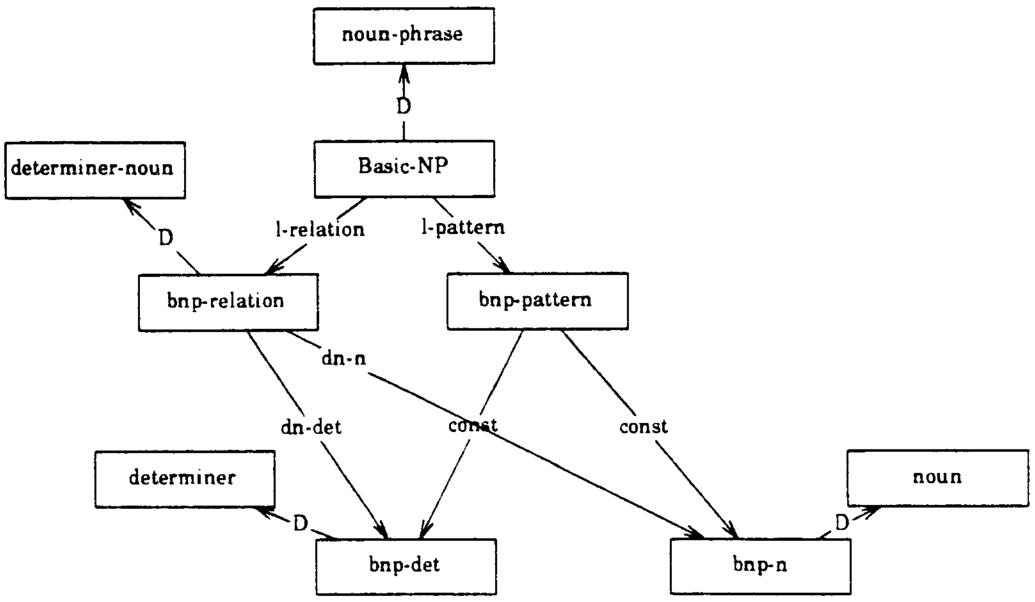

Figure A-14.

Objects which have names are generally referred to by name. The *name* relation is produced via mapping from such objects, causing the Name-NP pattern to be used:

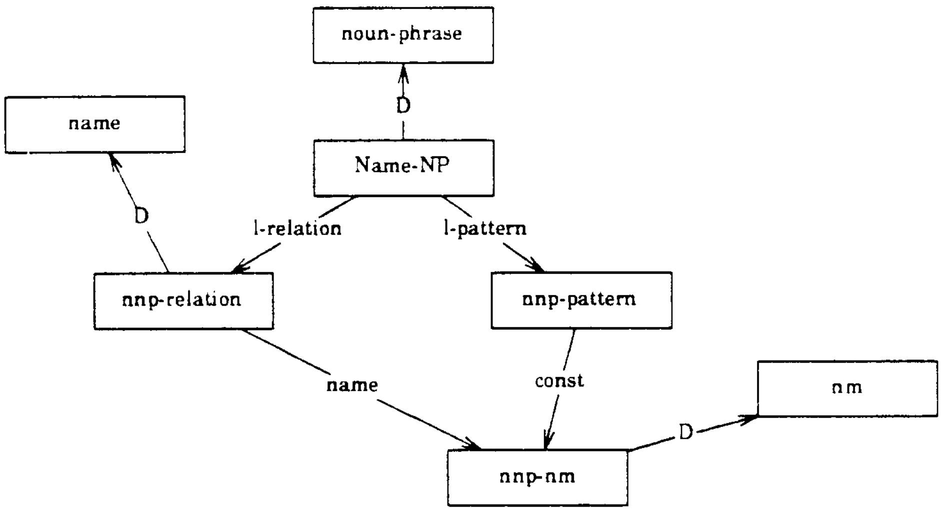

Figure A-15.

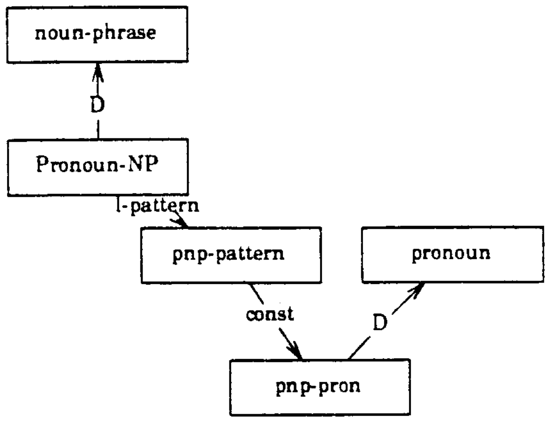

Figure A-16.

There are several ordinary noun-phrase patterns without determiners, used for referring to collectives, abstract concepts, generics, and groups. In these cases, the entire noun phrase is made up of the NP* component, to which constraints such as number are attached.

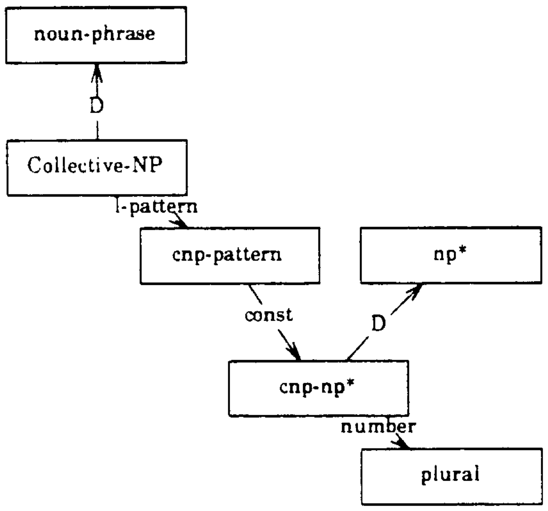

Figure A-17.

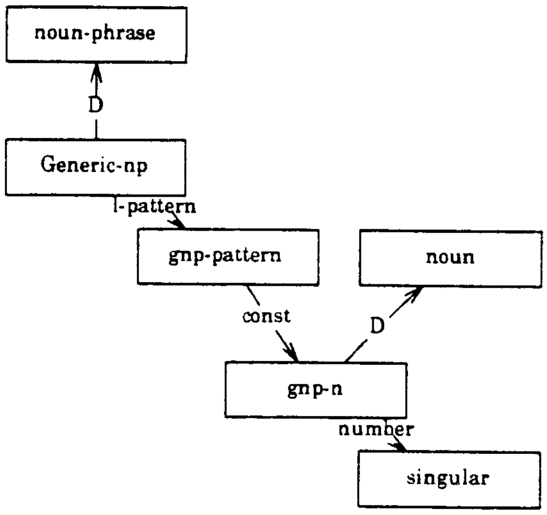

Figure A-18.

Certain noun-phrase patterns use verb phrase constructs. The gerund phrase and infinitive phrase are handled by the knowledge that these verb phrase constructs also belong to the class of noun phrase templates. The following knowledge accounts for the sentential complement:

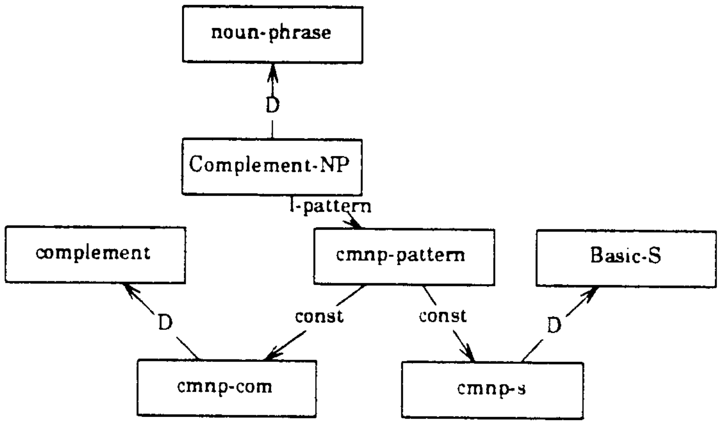

Figure A-19.

Handling modifiers is a difficult problem. In this knowledge base, pre- and post-modifiers are linked to their modified nouns by the *modifier-noun* relation, which does not predetermine the order in which they appear. The type of modifier determines whether mapping produces a *postmodifier-noun* relation. This generally occurs when the modifier is a predicate, as in "The man sold the book by Mary sold it to John". KING fails to be able to deal elegantly with constructs such as "the salvaged tire" because it will favor a Postmodified-NP pattern. This, like the "John gave Mary for her enjoyment...." sentence discussed earlier in this appendix, is an example where the length of a constituent makes it preferable to put it in a position other than where it would ordinarily belong. The influence of the complexity and length of a constituent on the choice of linguistic structure is a phenomenon which is not covered in KING.

Postmodifiers may appear after nouns or noun-phrases, depending on the restrictiveness of the modifier. The following pattern is used for non-restrictive postmodifiers and for constructs such as "who from Mary's home town":

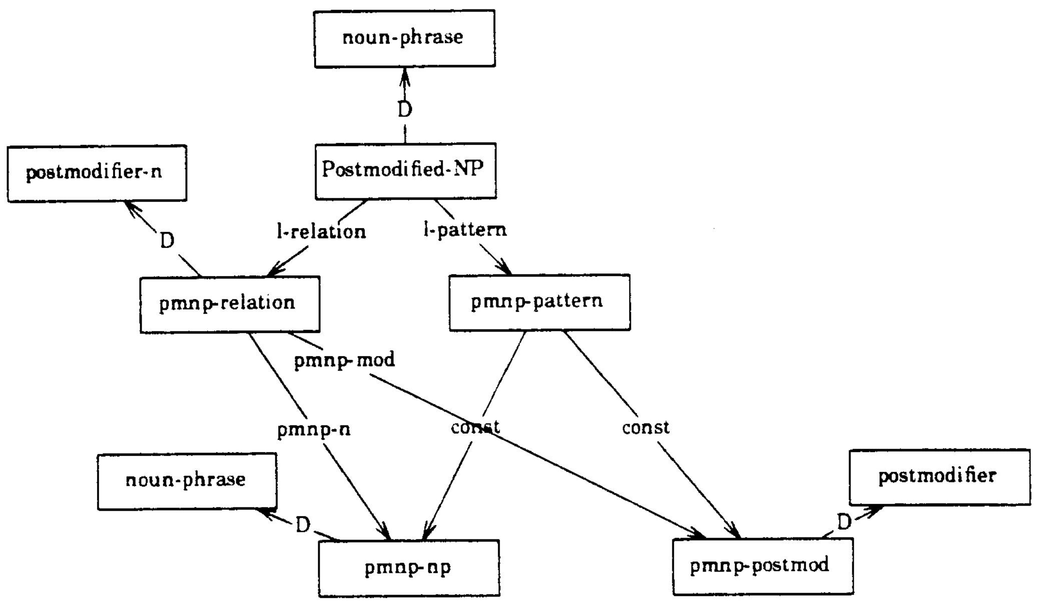

Figure A-20.

The following patterns are used for constructing NP* elements:

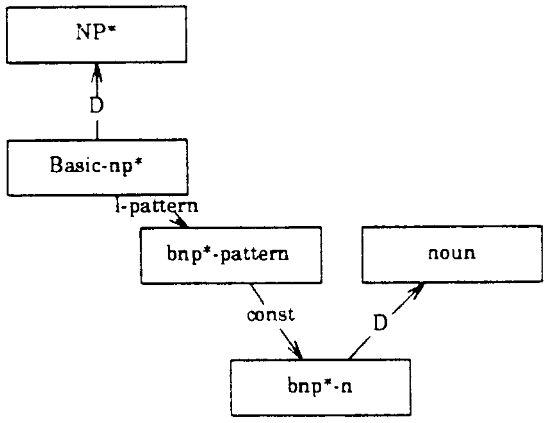

Figure A-21.

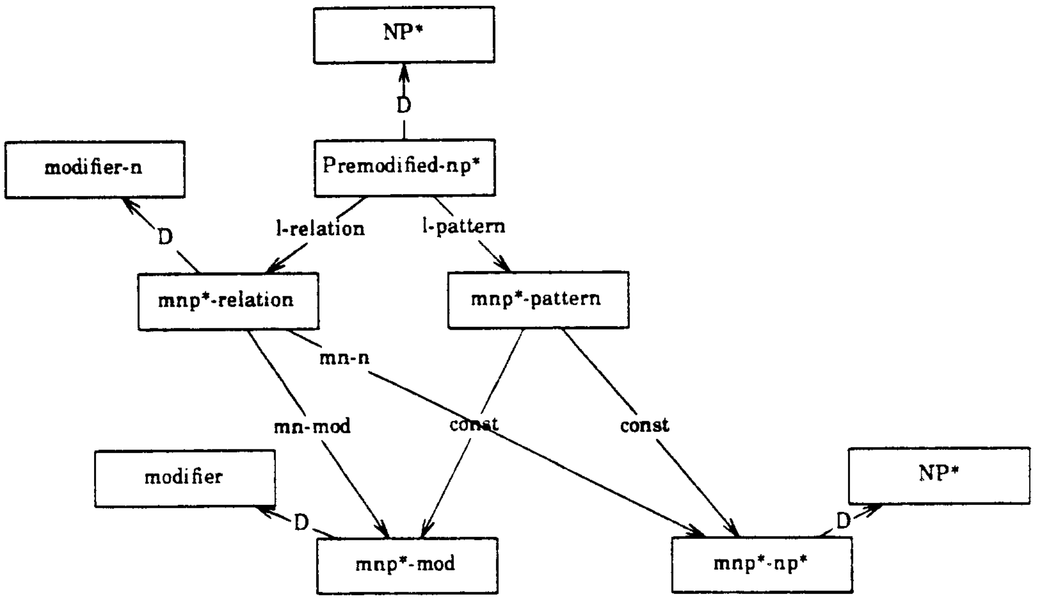

Figure A-22.

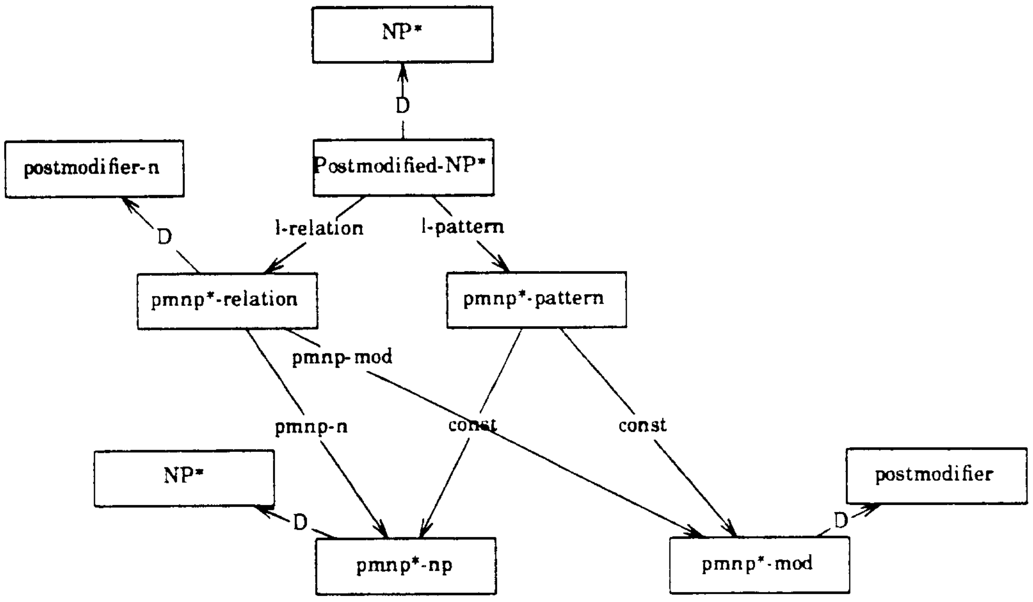

Figure A-23.

These are the normal premodifier constructs:

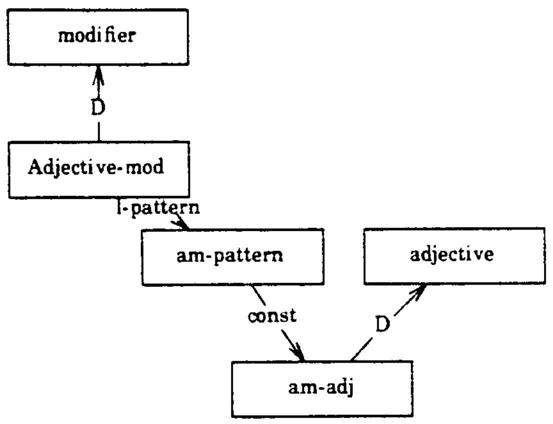

Figure A-24.

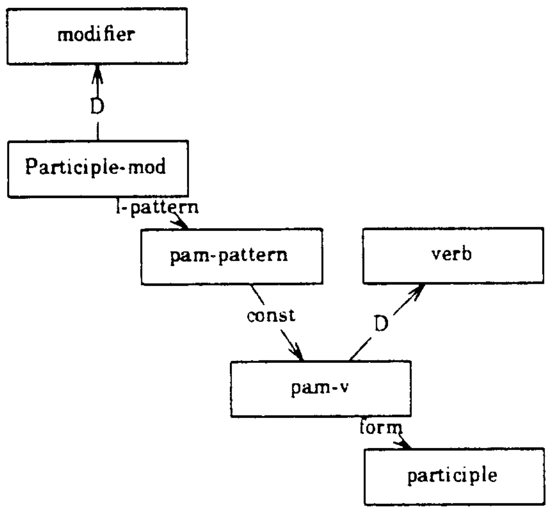

Figure A-25.

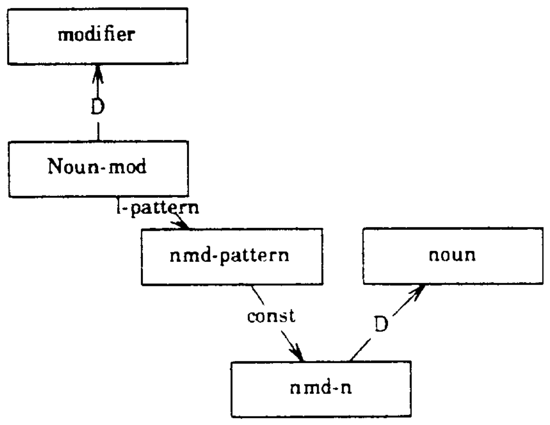

Figure A-26.

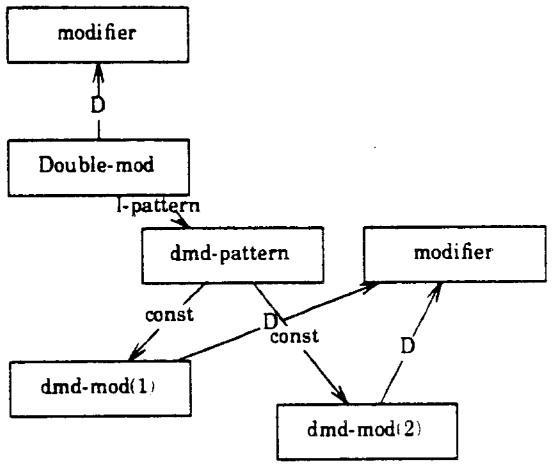

Figure A-27.

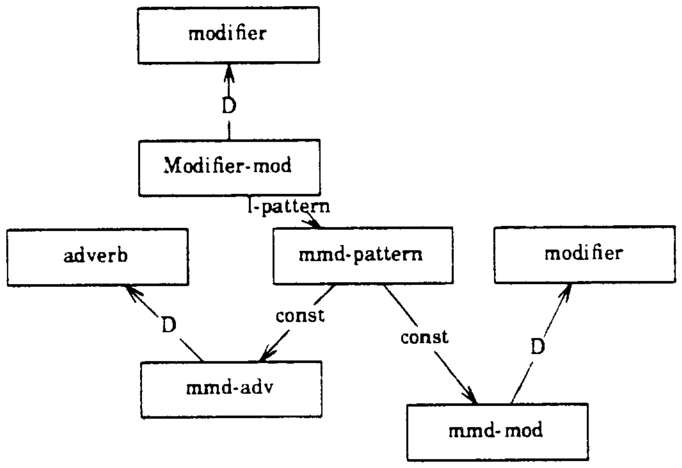

Figure A-28.

These are the postmodifier constructs, including relative clauses, prepositional phrases, and passive postmodifiers. These are used, for example, in "John knew the man who loved Mary", "What is the name of the file deleted from the queue?" and "the terminals on line":

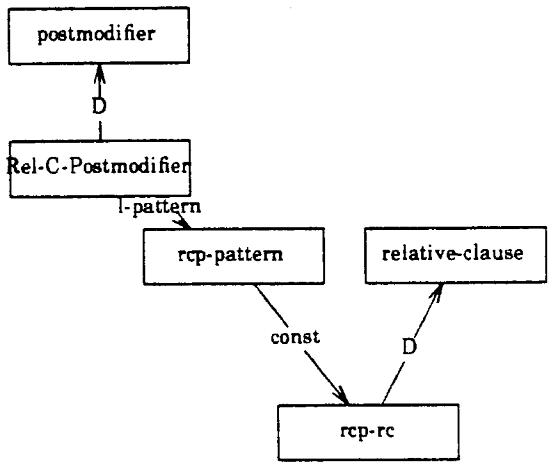

Figure A-29.

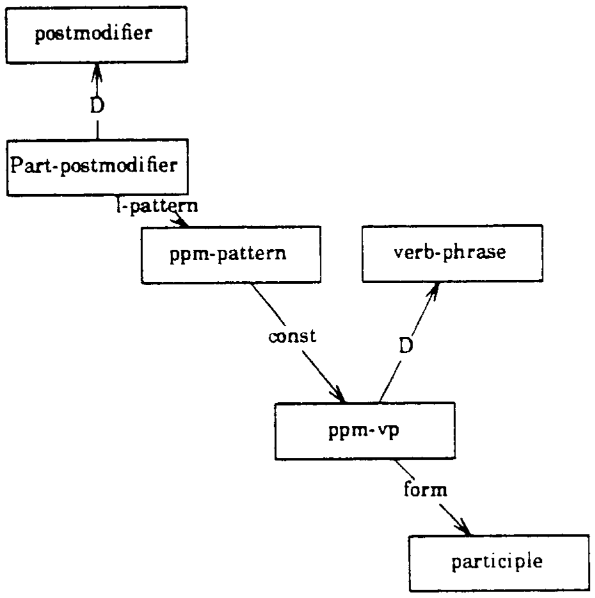

Figure A-30.

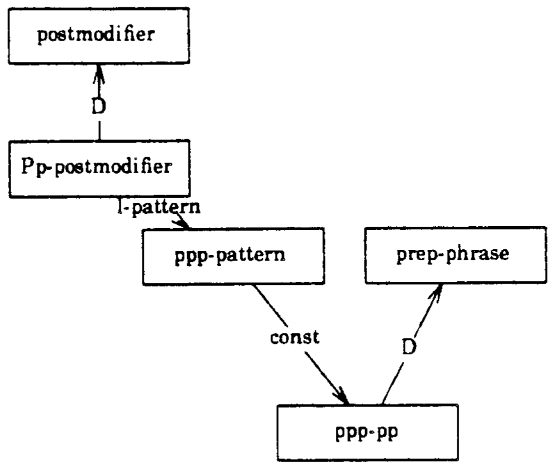

Figure A-31.

This is the prep phrase template:

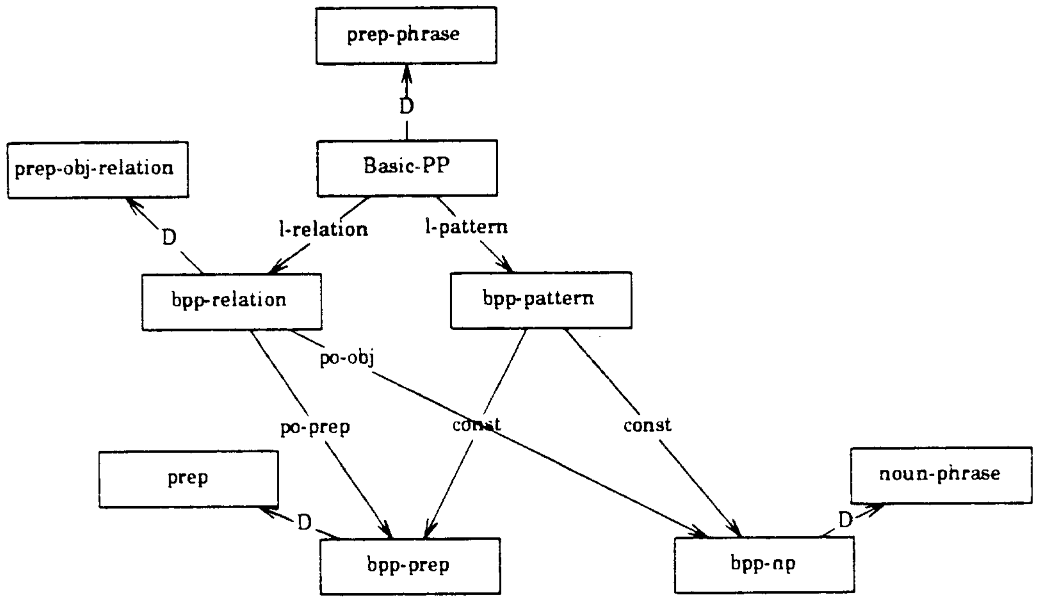

Figure A-32.

These are the adjunct patterns:

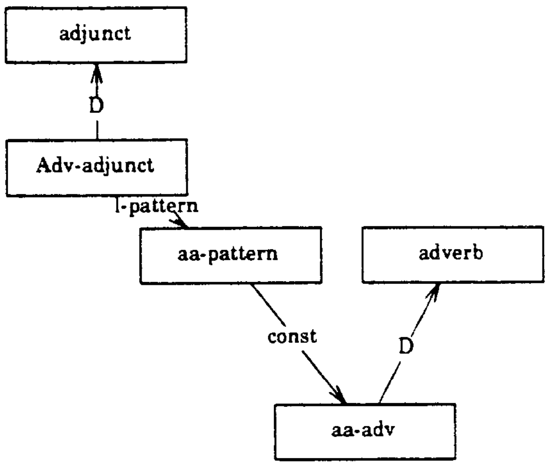

Figure A-33.

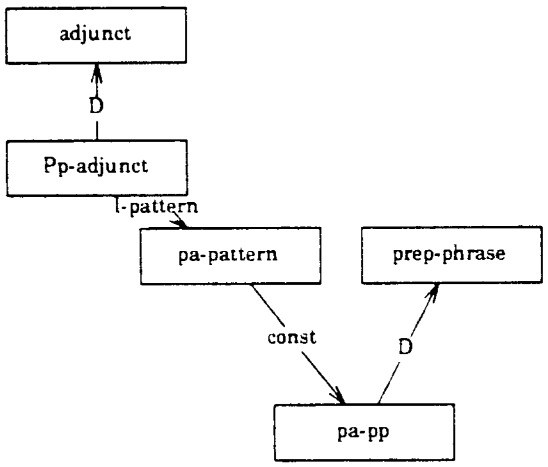

Figure A-34.

Here are the verb-part patterns:

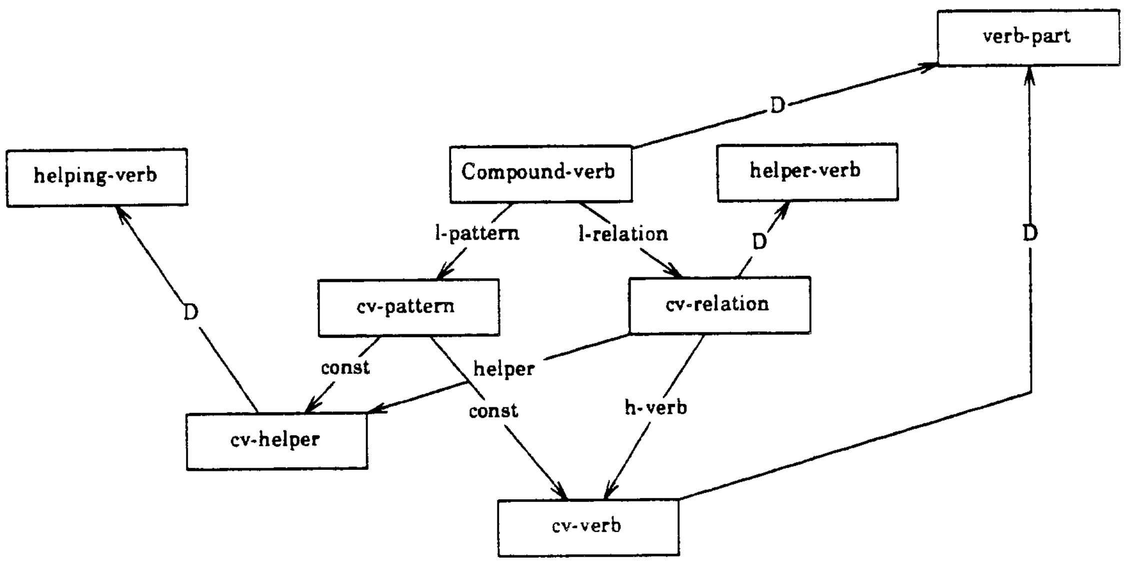

Figure A-35.

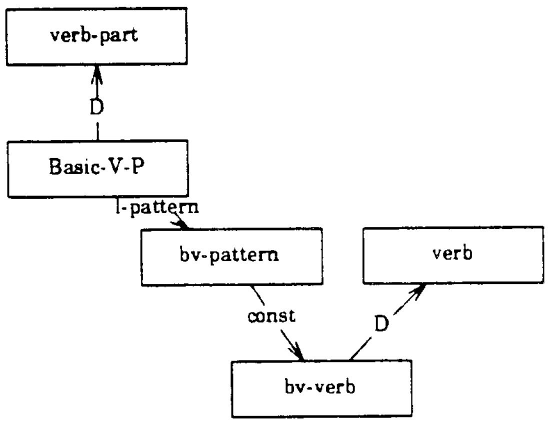

Figure A-36.

Appendix B: Some Knowledge Representation Examples

The examples of knowledge representation in the text were presented to illustrate particular aspects of the Ace framework. This section is provided to give a clearer picture of parts of an Ace network. The examples will be discussed with respect to their influence on processing.

The following is a piece of Ace knowledge about *write-permission* in UNIX:

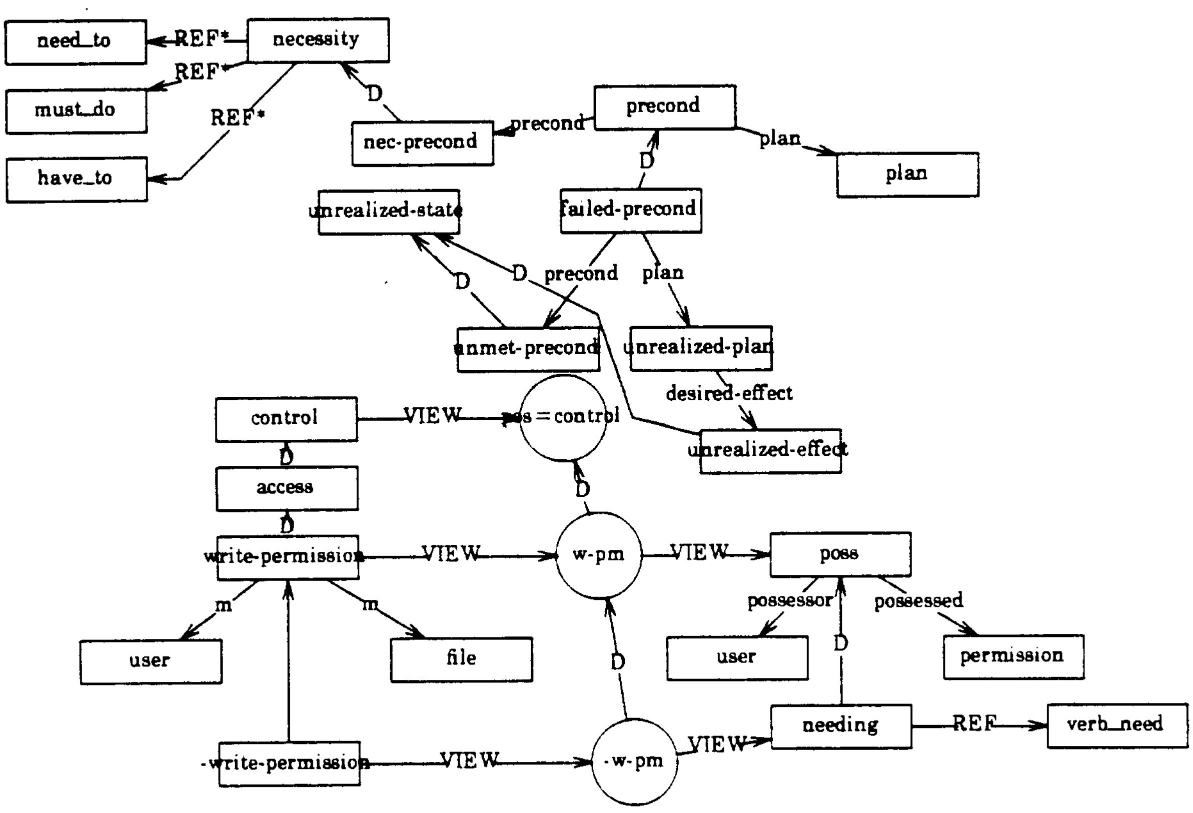

Figure B-1.

This diagram is a faithful rendition of knowledge used to generate sentences such as "You need write permission on the parent directory." There is no link between the knowledge about *write-permission* and the knowledge about *preconds*. The reason for their positioning above is to emphasize the fact that "You need write permission..." describes a state which is an instance of both *write-permission* and *unmet-precond.*

The links labeled "REF*" indicate that some sequence of associations leads from one structure to the other. Thus the *necessity* concept is joined to the *need_to* linguistic construct via chain of structured associations, some of which are not represented here. The *necessity* category DOMINATEs entities which are obligatory; this is associated with the concept of an *obligation*, which represents an entity which is obligatory for the *obligator*, as illustrated in Chapter 4. Thus "You must have write permission" refers indirectly to the necessary state by referring to an *obligation* on the part of the user.

The nodes *have_to*, *need_to*, and *must_do* are linguistic relations which dictate both the choice of verb and the constraints on the structure in which the verb appears. "You have to have write permission" and "The write permission has to be set" may be generated from this knowledge, depending on what maps into *obligator*. The verb "need" as in "You need to have write permission" behaves oddly, as the negative form could be "You needn't have write permission". "Need" and "dare" are two examples of auxiliaries which behave as auxiliaries only in special circumstances, and otherwise are preempted by their equi- forms.

Circles in the diagram are used for clarity in illustrating the hierarchy of structured associations linking concepts of *access* and *possession* to linguistic entities. The *pos=control* VIEW represents the common metaphorical relationship between possession and control. This metaphorical association can be applied to the concept of *write-permission*, which allows the user access to a file. The circles labeled "w-pm" and "-w-pm" represent implicit structured associations. The verb "have" can thus be used to describe this access. "Have" refers to a general state of possession: "You have write permission", "You must have write permission", and "You don't have write permission" are all possible forms. "Need" refers to a more particular state of possession--a possession state which is a *necessity* and an *unrealized-state*, as is the case in "You need write permission". The verb "lack" and a somewhat archaic form of "want" can also be used to refer to unrealized possession states, but "need" is still more specific.

The use of the VIEW here allows "need", "require", "have", and "lack" to be used to refer to states of *write-permission* without duplicating any structural knowledge about the use of the verbs. In the example given, "need" is the favored choice, because it is associated with the specific concept of not having something that is necessary.

When KING is called to inform the user that write permission is needed, it is passed a concept which is an instance of *write-permission* as well as playing the role of *unmet-precond* in the user's plan. Since the *plan* component of the plan is implicit in the user's statement-- "I tried typing 'rm foo' "--the response need only describe the failed precondition. KING thus generates the response, "You need write permission on the parent directory."

The input to KING consists of a network of instantiated concepts, which the generator traverses in producing the utterance. In the case of the response, "Use the 'rm' command", part of this network is the following:

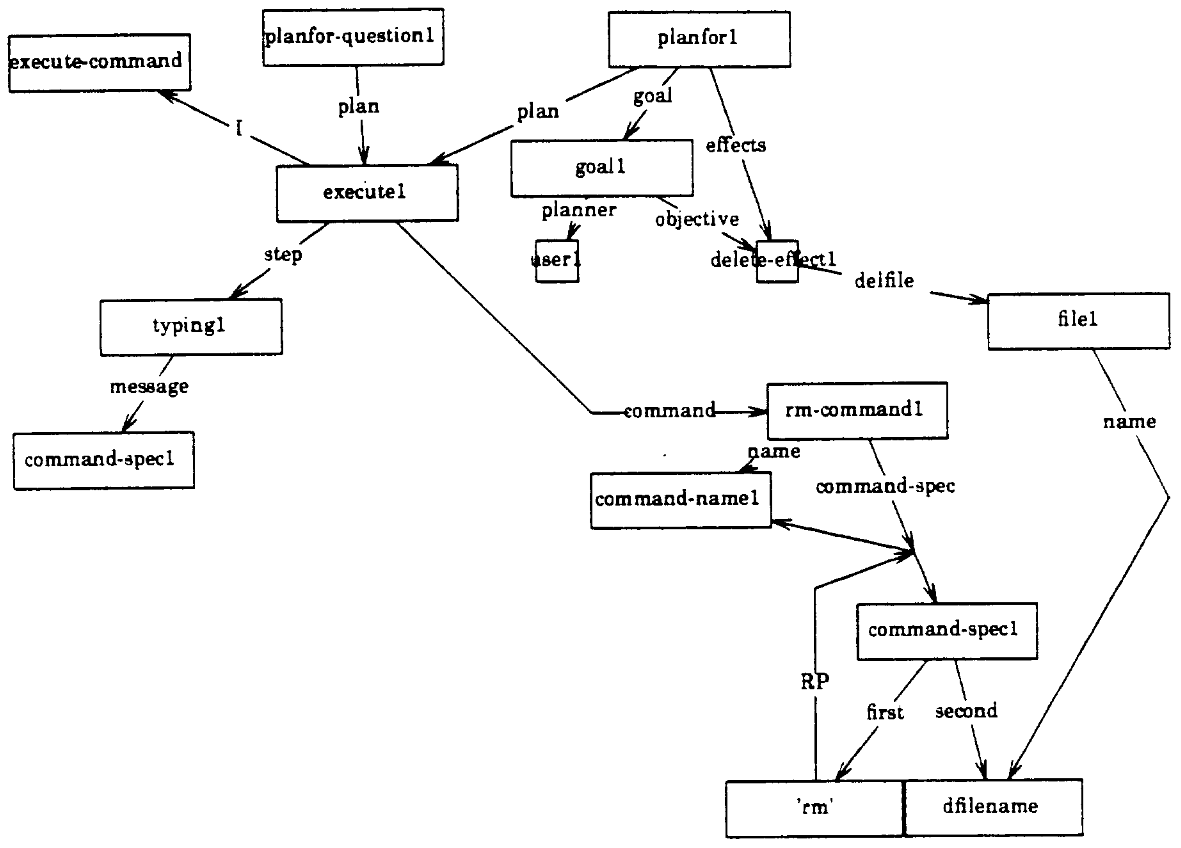

Figure B-2.

In response to the question "How do I delete a file?", the UNIX Consultant passes to KING a pointer to the *execute1* part of the network. This node is an instance of the *execute* concept, which represents the notion of using a UNIX command. A structured association between *rm-command* and the *name* relation, a subcategory of a more general structured association used to refer to named concepts.

When the response "Use the 'rm' command" is followed by the question, "How can I do that?", UC interprets the question as asking for details in executing the plan it has described. It passes to KING a pointer to the *step* association between *execute1* and *typing1*, because the system identifies *steps* as causal enablers, and "how" questions are requests for enablers.

One of the advantages of the KODIAK representation which is demonstrated in the Ace hierarchy is the ability to define concepts and relationships in terms of more abstract concepts and relationships. The commercial transaction example given in Chapter 4 demonstrates how knowledge about *transfer-events* is used in the representation of concepts such as *buying* and *selling*. The discussion in Chapter 5 mentions that the Ace representation poses a high start-up cost, because of the difficulty in initially arranging abstract concepts in the hierarchy. This hierarchy of abstract concepts is not illustrated in the text. The following diagram represents some of the abstract relationships in the system:

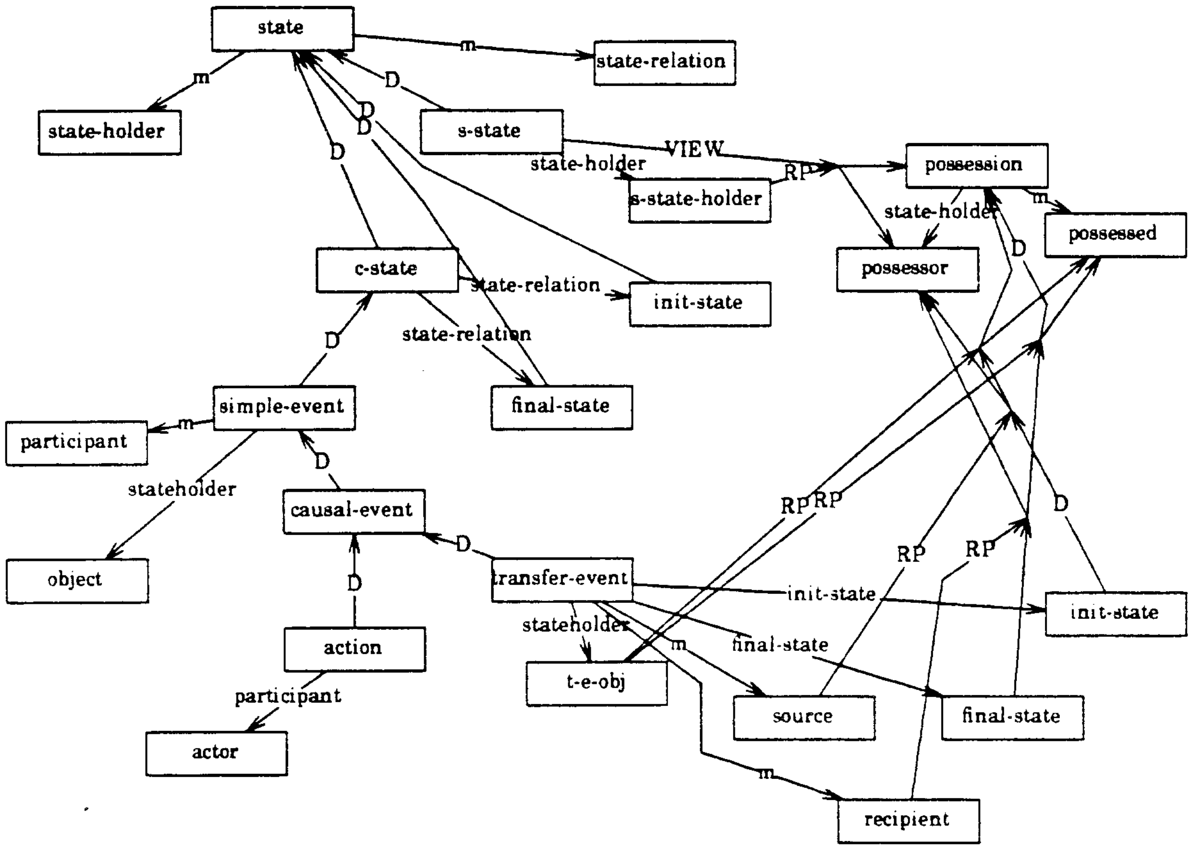

Figure B-3.

The concept of a *transfer-event* above is represented as a change in state of an object, where the initial state is *possession* on the part of the *source* and the final state is *possession* on the part of the *recipient*. The verbs "give" and "take" tend to be used to refer to transfer events. The motivation for metaphorical uses of "give" and "take" often comes from the abstract VIEW association between *s-state* and *possession* in the diagram above. This VIEW represents the close association between states of being and states of possession. For example, "I *am* ill" can mean "I *have* a headache" and "I *am* inspired" can mean "I *have* an idea". KING does not directly use the abstract relationship between being and possession, but makes use of structured associations which are motivated by this relationship. Constructions tend to be consistent about whether they use "being" constructs or "having" constructs, as in "That *gave* me a headache", and "You've just *given* me an idea", but "Pomegranites *make* me ill". The verb "make", meaning "cause to be" is used for constructs which may also use "be"; the verb "give", meaning "cause to have" is used for constructs which may also use "have". In languages where "I *have* hunger" is used to mean "I *am* hungry", "It *gives* me hunger" is used to mean "It *makes* me hungry". Such consistency applies to the UNIX world as well. "Chmod" can be used to *make* a file publicly writable, but *gives* one write permission on the file. Because permission is treated as a state of possession, constructs ordinarily used to describe possession are used to describe write permission. Such constructs cannot be used with writability, which is viewed as a state of *being* of a file.

This text has focussed on the *transfer-event* as a tool for exploiting conceptual relationships in language processing; however, a great deal of territory remains to be explored in terms of stative verbs, locative relationships, and other related phenomena. The above diagram may be related to linguistic structures as in the following schema:

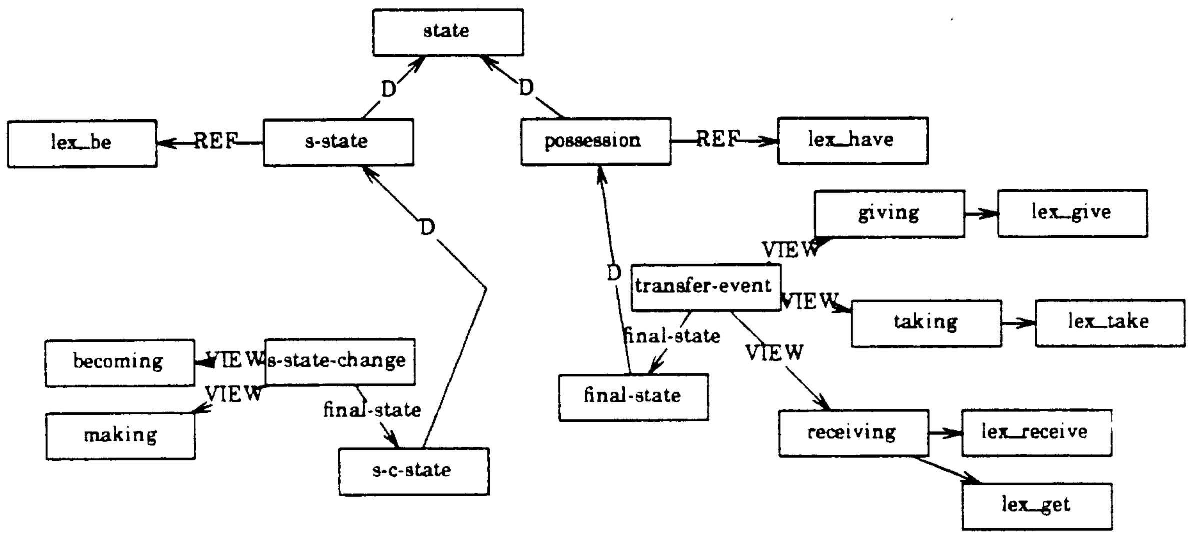

Figure B-4.

The relationships between *transfer-events* and the *giving* and *taking* actions as above have been discussed earlier. These *transfer-events* may also be referred to via the concept of *receiving*, which falls into a category whose properties are still being explored, known as *happen-to*. While *actions* designate an *actor* who must play an active role, *happen-tos* are events in which the designated role-player plays an inactive role. This idea can be applied to other types of *c-states*, or state changes, as well. For example, *becoming* above is a VIEW of *s-state-change* as a *happen-to*. VIEWed as an *action*, where the *actor* need not be part of the *s-state-change*, the *event* maps into *making*. Thus "John became ill" is to "Mary made John ill" as "John received the book" is to "Mary gave John the book".

The examples presented here illustrate some further details of the Ace representation and its incorporation of motivation and indirect reference into language processing. All of these diagrams, however, seem to open up as many questions as they answer. This is indicative of the many areas in which this approach must still be explored and developed.

References

Anderson, J.R., and Bower, G. H. 1973. *Human Associative Memory.* Washington, D. C.: Winston.

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

Appelt, D. 1983. Telegram: A grammar formalism for language planning. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics,* Cambridge, Massachusetts.

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

Austin, J. L. 1962. *How To Do Things With Words.* New York: Oxford University Press.

Bobrow, D. and Winograd, T. 1977. An overview of KRL, a knowledge representation language. *Cognitive Science 1* (1).

Brachman, R. 1979. On the Epistemological Status of Semantic Networks. In N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers.* New York: Academic Press.

Brachman, R., et. al. 1979. Research in Natural Language Understanding. Bolt Beranek and Newman Research Report #4274.

Brachman, R., Fikes, R., and Levesque, H. 1983. KRYPTON: Integrating terminology and assertion. In *Proceedings of the National Conference on Artificial Intelligence,* Washington, D. C.

Bruce, B. 1975. Belief Systems and Language Understanding. Bolt Beranek and Newman Research Report #2973.

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

Chafe, W. L. 1984. Integration and Involvement in Speaking, Writing, and Oral Literature. In D. Tannen (ed.), *Oral and Written Language.* Norwood, N. J.: Ablex.

Chester, D. 1976. The translation of formal proofs into English. *Artificial Intelligence 7 (3).*

Clippinger, J. H. 1974. *A Discourse Speaking Program as a Preliminary Theory of Discourse Behavior and a Limited Theory of Psychoanalytic Discourse.* Ph. D. thesis, University of Pennsylvania.

Cohen, P. R. 1978. On Knowing What to Say: Planning Speech Acts. University of Toronto, Technical Report #118.

Cohen, P. R. 1984. The pragmatics of referring and the modality of communication. *Computational Linguistics 10* (2).

Cohen, P. R. and Perrault, C. R. 1979. Elements of a plan-based theory of speech acts. *Cognitive Science 3*.

Davey, A. 1979. *Discourse Production*. Edinburgh: Edinburgh University Press.

Deering, M., Faletti, J., and Wilensky, R. 1981. PEARL: An efficient language for Artificial Intelligence programming. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia.

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

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

Friedman, J. 1969. Directed random generation of sentences. *Communications of the ACM 12 (6)*.

Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science 7*, pp. 155-170.

Goldman, N. M. 1974. *Computer Generation of Natural Language from a Deep Conceptual Base*. Ph. D. Thesis, Stanford University.

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

Grosz, B. J. 1977. The representation and use of focus in dialogue understanding. Stanford Research Institute Technical Note No. 151.

Grosz, B. J. 1979. Utterance and objective: Issues in natural language communication. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*,

Halliday, M. A. K. 1967. Notes on transitivity and theme in English. *Journal of Linguistics 3*.

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

Hobbs, J. 1978. Coherence and Coreference. SRI Technical Note No. 168.

Hobbs, J., and Evans, D. 1980. Conversation as planned behavior. *Cognitive Science 4,* pp. 349-377.

Jacobs, P. 1983. Generation in a natural language interface. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, Germany.

Jacobs, P., and Rau, L. 1984. Ace: associating language with meaning. In *Proceedings of the European Conference on Artificial Intelligence*, Pisa, Italy.

Jacobs, P. 1985. PHRED: A generator for natural language interfaces. University of California, Berkeley, Computer Science Division Technical Report #UCB/CSD 85/198.

Kaplan, R. M. and Bresnan, J. (eds.) 1983. *The Mental Representation of Grammatical Relations*. Cambridge: MIT Press.

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

Kay, M. 1984. Functional unification grammar: a formalism for machine translation. in *Proceedings of the Tenth International Conference on Computational Linguistics*, Stanford, California.

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

Klein, S. 1975. Meta-compiling text grammars as a model for human behavior. in *Proceedings of TINLAP-1*, Cambridge, Massachusetts.

Kripke, S. 1977. Speaker Reference and Semantic Reference. in French, et. al. (eds.) *Contemporary Perspectives in the Philosophy of Language*. Minneapolis: University of Minnesota Press.

Kukich, K. 1983. *Knowledge-Based Report Generation: A Knowledge-Engineering Approach to Natural Language Report Generation.* Ph. D. thesis, Univ. of Pittsburgh.

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

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

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

Lakoff, G. 1984. There-constructions: a case study in grammatical construction theory. University of California, Linguistics Working Paper.

Langacker, R. 1982. *Foundations of Cognitive Grammar.* Linguistics Department, University of California, San Diego.

Maida, A. 1984. Processing entailments and accessing facts in a uniform frame system. in *Proceedings of the National Conference on Artificial Intelligence.* Austin, Texas.

Mann, W. 1982. The Anatomy of a Systemic Choice. University of Southern California, ISI Technical Report #ISI/RR-82-104.

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

Mann, W. C. and Moore, J. A. 1980. Computer as Author -- Results and Prospects. USC Information Sciences Institute, Report #RR-79-82.

Mann, W. C. and Moore, J. A. 1981. Computer generation of multiparagraph English text. *American Journal of Computational Linguistics 7 (1).*

McCoy, K. 1982. Automatic enhancement of a data base knowledge representation used for natural language generation. Master's thesis, University of Pennsylvania.

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

McKeown, K. 1982. *Generating natural language text in response to questions about database structure.* Ph. D. thesis, University of Pennsylvania.

Meehan, J. R. 1977. TALE-SPIN, an interactive program that writes stories. in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.*

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

Moore, R. 1980. Reasoning about Knowledge and Action. SRI International AI Center Technical Report #191.

Quillian, M. R. 1966. *Semantic Memory.* Cambridge: Bolt Beranek and Newman.

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

Roberts, R. B. and Goldstein, I. P. 1977. The FRL Manual. MIT AI Lab Rebort #AIM-408.

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

Schank, R. C. and Abelson, R. P. 1977. *Scripts, Plans, Goals, and Understanding*. Halsted, New Jersey: Lawrence Erlbaum Associations.

Searle, J. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge: Cambridge University Press.

Searle, J. 1979a. Indirect speech acts. in *Expression and Meaning: Studies in the Theory of Speech Acts*, Cambridge: Cambridge University Press.

Searle, J. 1979b. A taxonomy of illocutionary acts. in *Expression and Meaning: Studies in the Theory of Speech Acts*, Cambridge: Cambridge University Press.

Shapiro, S. C. 1975. Generation as parsing from a network into a linear string. *American Journal of Computational Linguistics*, Fiche 33.

Shapiro, S. C. 1979. Generalized augmented transition network grammar for generation from semantic networks. In *Proceedings of the Seventeenth Meeting of the Association for Computational Linguistics*.

Sidner, C. L. 1979. *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse*. Ph. D. thesis, MIT.

Simmons, R. and Slocum, J. 1972. Generating English discourse from semantic networks. *Communications of the ACM 15 (10)*.

Sondheimer, N., Weischedel, R., and Bobrow, R., 1984. Semantic interpretation using KL-ONE. In *Proceedings of the Tenth International Conference on Computational Linguistics*, Palo Alto.

Swartout, W. R. 1981. *Producing Explanations and Justifications of Expert Consulting Programs*. Ph. D. thesis, MIT.

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

Wilensky, R. 1983. *Planning and Understanding: A Computational Approach to Human Reasoning*. Reading, Mass.: Addison-Wesley.

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

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

Wilensky, R., Arens, Y., and Chin, D. 1984. Talking to UNIX in English: An overview of UC. *Communications of the Association for Computing Machinery*, June.

Winograd, T. 1972. *Understanding Natural Language.* New York: Academic Press.

Winograd, T. 1983 *Language as a Cognitive Process, Vol. I: Syntax.* Reading, Mass.: Addison-Wesley.

Woods, W. A. 1970. Transition network grammars for natural language analysis. *CACM 13:10*, pp. 591-606.

Yazdani, M. 1982. How to write a story. In *Proceedings of the European Conference on Artificial Intelligence.*

Yngve, V. H. A. 1962. Random generation of English sentences. In *The 1961 Conference on Machine Translation and Applied Natural Language Analysis.* London: Her Majesty's Stationary Office.